INOVANCE

Advancing industrial technology,
for a better world



H5U and Easy Series Programmable Logic Controller

Instruction Guide

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

Data code 19012250A20

Legal and Disclaimer

Copyright Notice

Copyright © 2019 Shenzhen Inovance Technology Co., Ltd. This document is the intellectual property of Shenzhen Inovance Technology Co., Ltd.

Without written permission from the company, no part of this document may be excerpted, reproduced, modified, or translated, nor may it be distributed in any form.

Unauthorized use will be subject to legal action.

Trademark Notice

INOVANCE is a registered trademark of Shenzhen Inovance Technology Co., Ltd. and its affiliates. All other trademarks or registered trademarks mentioned in this document are the property of their respective owners. Unauthorized use of these trademarks by third parties for their own purposes constitutes infringement.

Disclaimer

Product specifications and technologies are subject to continuous improvement, which may lead to discrepancies between this document and the actual product. In such cases, the actual product shall prevail.

This document may be modified due to product upgrades without prior notice.

Disposal

Storage, use, and disposal of this product (including accessories) must comply with local laws and regulations.

Personnel Requirements

The product/system described in this document is intended for operation only by qualified professionals. Their operation must comply with the relevant instructions in this document, particularly the safety and warning notices.

Such qualified professionals must possess relevant training and experience to identify risks associated with this product/system and mitigate potential hazards.

Proper Use of the Product

Proper transportation, storage, assembly, installation, commissioning, operation, and maintenance are prerequisites for the safe and normal operation of the product. Ensure compliance with environmental requirements and follow all instructions in this document.

Preface

Introduction

The H5U series small-sized high-performance programmable logic controllers (PLCs) carry compact structure and 16 inputs and 14 outputs.

The Easy series small-sized and medium-range PLCs are available in 14 models, covering the demands of automation equipment requiring small footprint, multi-axis motion control, accurate temperature control, and easy networking.

This guide describes the basic instructions and complex instructions used for programming applications, as well as examples of these instructions.

Related Manuals

Document Name	Data Code	Description
H5U Series Programmable Logic Controller User Guide	19011517	Describes the product information, mechanical installation, communication connection specifications, features, and usage methods of the PLC.
Easy Series Programmable Logic Controller User Guide	PS00006444	Describes the installation and wiring of the product, including product information, mechanical installation, and electrical installation.
Easy321 and Easy511 Programmable Logic Controller User Guide	PS00017976	Describes the full-scenario task information of the Easy321 and Easy511 products, including product overview, installation, wiring, configuration and commissioning, special features, troubleshooting, and maintenance.
Easy321S Programmable Logic Controller User Guide	PS00021275	Describes the full-scenario task information of the Easy321S products, including product overview, installation, wiring, configuration and commissioning, special features, troubleshooting, maintenance, and appendix.
H5U and Easy Series Programmable Logic Controller Programming Guide	19012249	Describes the basic knowledge of PLC programming, quick start guidance, communication, motion control, and the use of high-speed counters.
H5U and Easy Series Programmable Logic Controller Instruction Guide (this guide)	19012250	Describes the basic instructions and complex instructions used for programming applications, as well as examples of these instructions.

Document Acquisition

This guide is not delivered with the product. You can obtain the PDF version in the following way:

- Do keyword search under Service and Support at http://www.inovance.com.
- Scan the QR code on the product with your smart phone.

- Scan the QR code below to install My Inovance app, where you can search for and download user guides.



Contents

2 New Functions 20

3 Instruction Overview 26

3.1 Instruction Structure 26

3.1.1 LD Instruction Structure 26

3.1.2 LiteST Instruction Structure 27

3.1.3 Lists of Soft Elements and Variables 27

3.2 Soft Elements 31

3.2.1 Bit Soft Elements 31

3.2.2 Word Soft Elements 31

3.2.3 Special Soft Elements 33

3.3 Variables 34

3.3.1 Overview 34

3.3.2 Defining a Variable 34

3.3.3 Defining an Array 36

3.3.4 Defining a Structure 37

3.3.5 Defining an Enumerated Variable 38

3.3.6 Defining an IP Variable 39

3.3.7 Defining a String 40

3.3.8 Defining a Specific Union 40

3.3.9 Using a Variable 41

3.4 Graphic Block Instructions 42

3.4.1 Instruction Structure 42

3.4.2 Instruction-Level Programming 43

3.4.3 Label Number Function 43

3.5 Function Block and Function 46

3.5.1 Function Block 46

3.5.2 Function 53

3.5.3 Authorizing a Function Block 56

3.5.4 Setting the Initial Value of Function Block 57

3.5.5 Encrypting a Function Block or Function 60

4 Instructions Cheat Sheet 62

5 Instruction Description 86

5.1 Program Logic Instructions 86

5.1.1 Contact Instructions 86

5.1.1.1 List of Instructions 86

5.1.1.2 LD 86

5.1.1.3 LDI 87

5.1.1.4 LDP 88
5.1.1.5 LDF 88
5.1.1.6 AND 89
5.1.1.7 ANDI 90
5.1.1.8 ANDP 91
5.1.1.9 ANDF 91
5.1.1.10 OR 92
5.1.1.11 ORI 93
5.1.1.12 ORP 94
5.1.1.13 ORF 95
5.1.1.14 MEP 96
5.1.1.15 MEF 96
5.1.2 Output Control Instructions 97
5.1.2.1 List of Instructions 97
5.1.2.2 OUT 98
5.1.2.3 SET 98
5.1.2.4 RST 99
5.1.2.5 ZSET 100
5.1.2.6 ZRST 101
5.1.2.7 PLS 103
5.1.2.8 PLF 104
5.1.2.9 ALT 105
5.1.2.10 R_TRIG 106
5.1.2.11 F_TRIG 107
5.1.3 Energy Flow Control Instructions 109
5.1.3.1 INV 109
5.1.4 Immediate Input (Contact) Instructions 109
5.1.4.1 List of Instructions 109
5.1.4.2 ILD 110
5.1.4.3 ILDI 111
5.1.4.4 IAND 112
5.1.4.5 IANDI 112
5.1.4.6 IOR 113
5.1.4.7 IORI 114
5.1.5 Immediate Output Instructions 115
5.1.5.1 List of Instructions 115
5.1.5.2 IOUT 116

5.1.5.3 ISET 117
5.1.5.4 IRST 117
5.1.5.5 IZSET 118
5.1.5.6 IZRST 119
5.1.6 Immediate Input and Output Instructions 120
5.1.6.1 List of Instructions 120
5.1.6.2 IRDBIT 121
5.1.6.3 IWRBIT 122
5.2 Process Control Instructions 122
5.2.1 List of Instructions 123
5.2.2 CJ 123
5.2.3 LBL 123
5.2.4 CALL 125
5.2.5 SSRET 126
5.2.6 EI 126
5.2.7 DI 127
5.2.8 WDT 128
5.2.9 FOR 128
5.2.10 NEXT 130
5.3 SFC Instructions 131
5.3.1 List of Instructions 131
5.3.2 STL 131
5.3.3 RET 132
5.3.4 OUTSTL 132
5.3.5 SETSTL 133
5.3.6 RSTETL 134
5.4 Contact Operation Instructions 135
5.4.1 Contact Comparison 135
5.4.1.1 List of Instructions 135
5.4.1.2 AND# 138
5.4.1.3 LD# 140
5.4.1.4 OR# 141
5.4.1.5 FLDD# 143
5.4.1.6 FANDD# 145
5.4.1.7 FORD# 147
5.4.1.8 LDZ# 149
5.4.1.9 ANDZ# 151

5.4.1.10 ORZ# 153
5.4.2 Contact Logical Operation 154
5.4.2.1 List of Instructions 155
5.4.2.2 LD* 155
5.4.2.3 AND* 156
5.4.2.4 OR* 158
5.5 Data Operation Instructions 159
5.5.1 Arithmetic Operation 159
5.5.1.1 List of Instructions 159
5.5.1.2 ADD 160
5.5.1.3 SUB 161
5.5.1.4 MUL 163
5.5.1.5 DIV 164
5.5.1.6 ABS 165
5.5.1.7 DEABS 167
5.5.1.8 MOD 168
5.5.1.9 EADD 170
5.5.1.10 ESUB 171
5.5.1.11 EMUL 172
5.5.1.12 EDIV 174
5.5.1.13 INC 175
5.5.1.14 DEC 176
5.5.2 Data Logic Operation 177
5.5.2.1 List of Instructions 177
5.5.2.2 WAND 177
5.5.2.3 WOR 178
5.5.2.4 WXOR 180
5.5.2.5 NEG 181
5.5.2.6 ENEG 183
5.5.3 Word Bit Operation 184
5.5.3.1 List of Instructions 184
5.5.3.2 BLD 184
5.5.3.3 BLDI 186
5.5.3.4 BAND 187
5.5.3.5 BANDI 188
5.5.3.6 BOR 189
5.5.3.7 BORI 190

5.5.3.8 BOUT 191
5.5.3.9 BSET 192
5.5.3.10 BRST 193
5.5.4 Trigonometric Function 194
5.5.4.1 List of Instructions 194
5.5.4.2 SIN 195
5.5.4.3 TAN 197
5.5.4.4 COS 199
5.5.4.5 ASIN 201
5.5.4.6 ACOS 203
5.5.4.7 ATAN 206
5.5.4.8 RAD 208
5.5.4.9 DEG 209
5.5.4.10 SINH 210
5.5.4.11 COSH 211
5.5.4.12 TANH 212
5.5.5 Table Operation 213
5.5.5.1 List of Instructions 213
5.5.5.2 WSUM 213
5.5.5.3 MEAN 214
5.5.5.4 LIMIT 215
5.5.5.5 BZAND 217
5.5.5.6 ZONE 219
5.5.5.7 SCL 221
5.5.5.8 SCL2 223
5.5.6 Exponent Operation 226
5.5.6.1 List of Instructions 226
5.5.6.2 EXP 227
5.5.6.3 LOG 228
5.5.6.4 LOGE 230
5.5.6.5 ESQR 231
5.5.6.6 SQR 232
5.5.6.7 POW 233
5.5.6.8 LN 234
5.5.6.9 EXPT 235
5.5.6.10 SQRT 236

5.6 Bit Operator Instructions 237
5.6.1 List of Instructions 237
5.6.2 AND 237
5.6.3 OR 238
5.7 Data Processing Instructions 239
5.7.1 Data Conversion 239
5.7.1.1 List of Instructions 240
5.7.1.2 INT 242
5.7.1.3 BCD 243
5.7.1.4 BIN 244
5.7.1.5 FLT 245
5.7.1.6 EBCD 247
5.7.1.7 EBIN 248
5.7.1.8 DABIN 248
5.7.1.9 BINDA 250
5.7.1.10 WBIT 252
5.7.1.11 WTOB 253
5.7.1.12 BITW 255
5.7.1.13 BTOW 256
5.7.1.14 UNI 257
5.7.1.15 MCPY 259
5.7.1.16 MSET 261
5.7.1.17 DIS 263
5.7.1.18 WTODW 264
5.7.1.19 DWTOW 265
5.7.1.20 ASCI 266
5.7.1.21 HEX 268
5.7.1.22 DECO 270
5.7.1.23 ENCO 272
5.7.1.24 SCALE_X 273
5.7.1.25 NORM_X 275
5.7.1.26 ROUND 276
5.7.1.27 TRUNC 277
5.7.1.28 CELL 278
5.7.1.29 FLOOR 279
5.7.1.30 AryByteTo 280
5.7.1.31 AryToString 285

5.7.1.32 BOOL_TO_(TYPE) 287
5.7.1.33 DINT_TO_(TYPE) 288
5.7.1.34 INT_TO_(TYPE) 289
5.7.1.35 REAL_TO_(TYPE) 290
5.7.1.36 BYTE_TO_(TYPE) 292
5.7.1.37 STRING_TO_(TYPE) 293
5.7.1.38 StringToAry 295
5.7.1.39 TO_(TYPE) 297
5.7.1.40 ToAryByte 298
5.7.2 Data Transfer and Comparison 303
5.7.2.1 List of Instructions 303
5.7.2.2 MOV 303
5.7.2.3 EMOV 304
5.7.2.4 BMOV 306
5.7.2.5 SMOV 307
5.7.2.6 FMOV 308
5.7.2.7 CML 309
5.7.2.8 CMP 310
5.7.2.9 ECMP 312
5.7.2.10 ZCP 313
5.7.2.11 EZCP 314
5.7.3 Table Operation 316
5.7.3.1 List of Instructions 316
5.7.3.2 SORTR 316
5.7.3.3 SORTC 319
5.7.3.4 SER 322
5.7.3.5 FDEL 324
5.7.3.6 FINS 326
5.7.3.7 POP 328
5.7.3.8 RAMP 330
5.7.4 Data Shift 331
5.7.4.1 List of Instructions 331
5.7.4.2 ROR 332
5.7.4.3 ROL 333
5.7.4.4 RCR 334
5.7.4.5 RCL 335
5.7.4.6 SFTR 336

5.7.4.7 SFTL 337
5.7.4.8 WSFR 339
5.7.4.9 WSFL 340
5.7.4.10 SFWR 341
5.7.4.11 SFRD 342
5.7.4.12 SFR 344
5.7.4.13 SFL 345
5.7.4.14 SHL 347
5.7.4.15 SHR 348
5.7.5 Other Data Processing 349
5.7.5.1 List of Instructions 349
5.7.5.2 SWAP 350
5.7.5.3 BON 351
5.7.5.4 SUM 352
5.7.5.5 RAND 353
5.7.5.6 XCH 355
5.7.5.7 EFMOV 356
5.7.5.8 CCD 357
5.7.5.9 CRC 359
5.7.5.10 LRC 360
5.8 Matrix Instructions 362
5.8.1 Matrix Operation 362
5.8.1.1 List of Instructions 362
5.8.1.2 BK+ 363
5.8.1.3 BK- 364
5.8.1.4 MAND 366
5.8.1.5 MOR 368
5.8.1.6 MXNR 369
5.8.1.7 MXOR 371
5.8.1.8 MINV 373
5.8.2 Matrix Comparison 374
5.8.2.1 List of Instructions 374
5.8.2.2 BKCMP# 374
5.9 String Instructions 377
5.9.1 List of Instructions 377
5.9.2 STR 378
5.9.3 STRMOV 383

5.9.4 VAL 384
5.9.5 ESTR 389
5.9.6 EVAL 393
5.9.7 $ADD 396
5.9.8 LEN 398
5.9.9 INSTR 400
5.9.10 RIGHT 402
5.9.11 LEFT 405
5.9.12 MIDW 408
5.9.13 MIDR 410
5.9.14 $MOV 412
5.9.15 ClearString 413
5.9.16 CONCAT 414
5.9.17 DELETE 415
5.9.18 FIND 416
5.9.19 REPLACE 417
5.9.20 INSERT 418
5.10 Array Instructions 419
5.10.1 List of Instructions 419
5.10.2 AryMax 420
5.10.3 AryMin 422
5.10.4 ArySearch 424
5.10.5 SizeOfAry 426
5.10.6 AryMove 427
5.10.7 AryExchange 429
5.10.8 CLEAR 431
5.10.9 AryAddV 433
5.10.10 ArySubV 434
5.10.11 ArySHL 436
5.10.12 ArySHR 438
5.10.13 AryAnd 440
5.10.14 AryOr 442
5.10.15 AryXor 444
5.10.16 AryXorN 446
5.10.17 ArySD 448
5.10.18 AryMean 450

5.11 Selection 451
5.11.1 List of Instructions 451
5.11.2 MAX 451
5.11.3 MIN 452
5.11.4 SEL 454
5.12 Clock Instructions 455
5.12.1 List of Instructions 455
5.12.2 TCMP 456
5.12.3 TZCP 457
5.12.4 TADD 459
5.12.5 TSUB 460
5.12.6 HTOS 462
5.12.7 STOH 463
5.12.8 TRD 464
5.12.9 TWR 466
5.12.10 HOUR 467
5.12.11 TZSET 469
5.12.12 TZGET 472
5.13 MC Axis Control (EtherCAT and Pulse Output) 474
5.13.1 Basic Instructions 474
5.13.1.1 List of Instructions 474
5.13.1.2 Description of MC Axis State Machine 476
5.13.1.3 MC_Power 477
5.13.1.4 MC_Reset 479
5.13.1.5 MC_ReadStatus 482
5.13.1.6 MC_ReadAxisError 486
5.13.1.7 MC_ReadDigitalInput 488
5.13.1.8 MC_ReadActualPosition 490
5.13.1.9 MC_ReadActualTorque 492
5.13.1.10 MC_ReadActualVelocity 494
5.13.1.11 MC_SetPosition 496
5.13.1.12 MC_TouchProbe 499
5.13.1.13 MC_MoveRelative 507
5.13.1.14 MC_MoveVelocity 514
5.13.1.15 MC_MoveAbsolute 518
5.13.1.16 MC_Jog 526
5.13.1.17 MC_TorqueControl 531

5.13.1.18 MC_Home 536
5.13.1.19 MC_Stop 540
5.13.1.20 MC_Halt 543
5.13.1.21 MC_MoveFeed 547
5.13.1.22 MC_MoveBuffer 557
5.13.1.23 MC_ImmediateStop 563
5.13.1.24 MC_MoveSuperImposed 567
5.13.1.25 MC_MoveVelocityCSV 570
5.13.1.26 MC_SyncMoveVelocity 573
5.13.1.27 MC_SyncTorqueControl 575
5.13.1.28 MC_SetAxisConfigPara 577
5.13.1.29 MC_SetOverride 583
5.13.1.30 MC_FollowVelocity 587
5.13.1.31 MC_MoveVelocityPV 589
5.13.1.32 MC_MoveAbsRelPP 593
5.13.1.33 MC_JogP 601
5.13.1.34 MC_MoveBufferPP 607
5.13.1.35 MC_TorqueControlPT 612
5.13.1.36 MC_ResetFollowingError 617
5.13.1.37 Axis Fault Codes 621
5.13.2 Instructions of Electronic Cam and Electronic Gear 631
5.13.2.1 List of Instruction 631
5.13.2.2 MC_CamIn 632
5.13.2.3 MC_CamOut 645
5.13.2.4 MC_GetCamTablePhase 647
5.13.2.5 MC_GetCamTableDistance 649
5.13.2.6 MC_GearIn 650
5.13.2.7 MC_GearOut 654
5.13.2.8 MC_Phasing 655
5.13.2.9 MC_SaveCamTable 658
5.13.2.10 MC_GenerateCamTable 659
5.13.2.11 MC_DigitalCamSwitch 665
5.13.2.12 MC_GearInPos 669
5.13.2.13 Fault Codes 676
5.13.3 Axis Group Control Instructions 684
5.13.3.1 List of Instructions 685
5.13.3.2 MC_MoveLinear 685

5.13.3.3 MC_MoveCircular 694
5.13.3.4 MC_MoveEllipse 700
5.13.3.5 MC_GroupStop 705
5.13.3.6 MC_GroupPause 708
5.13.3.7 Fault Codes 711
5.14 MC Axis Control (CANopen) 717
5.14.1 List of Instructions 717
5.14.2 MC_Power_CO 718
5.14.3 MC_Reset_CO 720
5.14.4 MC_ReadActualVelocity_CO 722
5.14.5 MC_ReadActualPosition_CO 723
5.14.6 MC_Halt_CO 724
5.14.7 MC_Stop_CO 726
5.14.8 MC_MoveVelocity_CO 728
5.14.9 MC_MoveRelative_CO 730
5.14.10 MC_MoveAbsolute_CO 733
5.14.11 MC_Home_CO 736
5.14.12 MC_Jog_CO 738
5.14.13 MC_WriteParameter_CO 741
5.14.14 MC_ReadParameter_CO 743
5.14.15 Error Codes 745
5.15 HC Axis Control (Pulse Input) 747
5.15.1 List of Instructions 747
5.15.2 ENC_Counter 749
5.15.3 ENC_Reset 754
5.15.4 ENC_Preset 756
5.15.5 ENC_TouchProbe 761
5.15.6 ENC_ArrayCompare 772
5.15.7 ENC_StepCompare 783
5.15.8 ENC_Compare 788
5.15.9 ENC_GroupArrayCompare 790
5.15.10 ENC_ReadStatus 794
5.15.11 ENC_DigitalOutput 796
5.15.12 ENC_ResetCompare 797
5.15.13 ENC_SetUnit 800
5.15.14 ENC_SetLineRotationMode 803
5.15.15 ENC_SetCompareOutputPara 805

5.15.16 HC_Preset 807
5.15.17 HC_Counter 811
5.15.18 HC_TouchProbe 814
5.15.19 HC_Compare 820
5.15.20 HC_ArrayCompare 823
5.15.21 HC_StepCompare 826
5.15.22 Error Codes 830
5.16 Timer Instructions 838
5.16.1 Parameters 838
5.16.2 List of Instructions 838
5.16.3 TPR 839
5.16.4 TONR 841
5.16.5 TOFR 843
5.16.6 TACR 845
5.17 Pointer Instructions 847
5.17.1 List of Instructions 847
5.17.2 PTGET 848
5.17.3 PTINC 849
5.17.4 PTDEC 850
5.17.5 PTADD 851
5.17.6 PTSUB 852
5.17.7 PTSET 854
5.17.8 PTMOV 856
5.17.9 PT# 857
5.18 Communication Protocol Instructions 859
5.18.1 List of Instructions 859
5.18.2 SerialSR 862
5.18.3 SerialSend 866
5.18.4 SerialRcv 868
5.18.5 Error Codes of Serial Port Free Protocol Communication Instructions 872
5.18.6 MB_Master 872
5.18.7 MB_Client 875
5.18.8 Error Codes of Modbus Communication Instructions 879
5.18.9 SysHC_NTPClient 880
5.18.10 TCP_Listen 882
5.18.11 TCP_Accept 884
5.18.12 TCP_Connect 886

5.18.13 TCP_Close 888
5.18.14 TCP_Send 890
5.18.15 TCP_Receive 893
5.18.16 TCP Server Communication Instance 895
5.18.17 TCP Client Communication Instance 896
5.18.18 Connection-Oriented TCP Socket Communication 897
5.18.19 Connectionless UDP Socket Communication 898
5.18.20 UDP_Bind 899
5.18.21 UDP_Receive 901
5.18.22 UDP_Send 903
5.18.23 UDP_JoinGroup 906
5.18.24 UDP Communication Instance 908
5.18.25 Error Codes of Socket Communication Instructions 908
5.18.26 SFTP_GetFile 913
5.18.27 SFTP_PutFile 916
5.18.28 SFTP_GetFileList 919
5.18.29 SFTP_DeleteFile 921
5.18.30 SFTP_MoveFile 924
5.18.31 SFTP_CreateDir 927
5.18.32 SFTP_RenameFile 929
5.18.33 Error Codes of SFTP Instructions 932
5.18.34 ETC_ReadParameter_CoE 933
5.18.35 ETC_WriteParameter_CoE 936
5.18.36 ETC_RestartMaster 940
5.18.37 Instruction Code Information 943
5.18.38 EIP_Generic_Service 945
5.18.39 EIP_Get_Attributes_All 948
5.18.40 EIP_Get_Attribute_Single 951
5.18.41 EIP_Set_Attributes_All 954
5.18.42 EIP_Set_Attribute_Single 957
5.18.43 EIP_Apply_Attributes 959
5.18.44 EIP_Generic_Service通信实例 962
5.18.45 EIP Combined Application Instance 963
5.18.46 EIP_NOP 964
5.18.47 EIP_Reset 967
5.18.48 EIP_Start 969

5.18.49 EIP_Stop 972
5.18.50 EIP_OpenWithDataSize 974
5.18.51 EIP_Close 977
5.18.52 EIP_Read 980
5.18.53 EIP_Write 983
5.18.54 EIP Class3 Label Communication Instance 986
5.18.55 Error Codes of EIP Communication Instructions 987
5.18.56 OPCUA_Server_Shutdown 988
5.18.57 CANBus_DevRxMsg 990
5.18.58 CANBus_DevTxMsg 993
5.18.59 Description of CANBUS Structure 995
5.18.60 CANBUS Error Codes 997
5.19 User Program Backup Instructions 997
5.19.1 SysHC_Appbackup 997
5.20 Other Instructions 1001
5.20.1 List of Instructions 1001
5.20.2 PID 1002
5.20.3 PID_AT 1029
5.20.4 PID_ATC 1051
5.20.5 Error Codes 1078
6 Appendix 1082
6.1 ASCII Code Conversion 1082
6.2 Error Codes 1087

2 New Functions

V4.11.0.5

Date	Software and Firmware Version	Release Version	Description
January 2026	• Software version: AutoShop V4.11.0.5 • Firmware version: V6.11.0.1	A20	-

V4.11.0.0

Date	Software and Firmware Version	Release Version	Description
September 2025	• Software version: AutoShop V4.11.0.0 • Firmware version: V6.5.0.0	A20	• New ◦ Added 3.3.5 Defining an Enumerated Variable. ◦ Added SFTP instructions, including 5.18.26 SFTP_GetFile, 5.18.27 SFTP_PutFile, SFTP_GetFileList, SFTP_Delete File, 5.18.30 SFTP_MoveFile, 5.18.31 SFTP_CreateDir, and SFTP_Rename File. ◦ Added the SysHC_NTPClient instruction. ◦ Added the 5.13.1.36 MC_ResetFollowingError instruction. • Modify ◦ Modified the description of the communication instructions 5.18.3 SerialSend and SerialRcv under Instruction Description. ◦ Combined LiteST Instructions and LD&LiteST Instructions into Instruction Description.

V4.10.2.0

Date	**Software and Firmware Version**	**Version**	**Description**
January 2025	• Software version: AutoShop V4.10.2.0 and later • Firmware version: V6.4.0.0 and later	A19	• **New** ◦ Added the following content to the "Communication Instructions" section. ▪ 5.18.50 EIP_OpenWithDataSize and 5.18.51 EIP_Close ▪ 5.18.52 EIP_Read, 5.18.53 EIP_Write, EIP Class 3 Label Communication Instance, and 5.18.55 Error Codes of EIP Communication Instructions

Date	Software and Firmware Version	Version	Description
			▪ 5.18.57 CANBus_DevRxMsg, 5.18.58 CANBus_DevTxMsg, CANBus Structures, CANBus Error Codes, and MC Axis Control Instructions (EtherCAT and Pulse Input) ◦ Added the following content to the "HC Axis Control Instructions (Pulse Input)" section: 5.15.15 ENC_SetCompareOutputPara ◦ Added the following content to the "User Program Backup Instructions" section: 5.19.1 SysHC_Appbackup • **Modify** ◦ Modified the following content in the "Instruction Description (LD and LiteST)" section: 5.20.3 PID_AT, 5.20.4 PID_ATC, and 5.20.5 Error Codes. ◦ Modified the following content to "MC Axis Control Instructions (EtherCAT and Pulse Output)" in the "Instruction Description (LD and LiteST)" section. ▪ Modified the function and instruction description in MC_MoveAbsolute. ▪ Modified the timing diagram of 5.13.1.3 MC_Power. ▪ Modified the function and instruction description and timing diagram of 5.13.1.29 MC_SetOverride, MC_MoveLinear, 5.13.3.3 MC_MoveCircular, 5.13.3.4 MC_MoveEllipse, MC_GroupSto, and 5.13.3.6 MC_GroupPause.

V4.10.0.0

Date	Software and Firmware Version	Version	Description
October 2024	• Software version: AutoShop V4.10.0.0 and later • Firmware version: V6.3.0.3 and later	A18	Corrected minor errors.
July 2024	• Software version: AutoShop V4.10.0.0 and later	A17	• Added the following content to "Immediate Input Instructions (Contact)" in the "Instruction Description (LD and LiteST)" section: ◦ Instruction List and ILD and ILDI ◦ IAND and IANDI and IOR and IORI

Date	Software and Firmware Version	Version	Description
	• Firmware version: V6.3.0.3 and later		• Added the following content to "Immediate Output Control Instructions" in the "Instruction Description (LD and LiteST)" section: ◦ Instruction List and 5.1.5.2 IOUT ◦ 5.1.5.3 ISET and 5.1.5.4 IRST ◦ 5.1.5.5 IZSET and 5.1.5.6 IZRST • Added the following content to "MC Axis Control Instructions (EtherCAT and Pulse Output)" in the "Instruction Description (LD and LiteST)" section: ◦ 5.13.1.34 MC_MoveBufferPP ◦ 5.13.1.35 MC_TorqueControlPT • Added the following content to "Communication Instructions" in the "Instruction Description (LD and LiteST)" section: ◦ 5.18.23 UDP_JoinGroup ◦ 5.18.56 OPCUA_Server_Shutdown • Added the following content to "Explicit Conversion Instructions" in the "Instruction Description (LiteST)" section: STRING_TO_(TYPE) • Added the following content to "Data Processing Instructions" in the "Instruction Description (LiteST)" section: ◦ Instruction List and 5.7.1.38 StringToAry ◦ 5.7.1.31 AryToString and 5.7.1.40 ToAryByte ◦ 5.7.1.30 AryByteTo • Added the following content to "Array Instructions" in the "Instruction Description (LiteST)" section: ◦ AryAddV and ArySubV and 5.10.18 AryMean ◦ 5.10.17 ArySD and ArySHL and ArySHR ◦ AryAnd, AryOr, AryXor, and AryXorN

Other historical versions

Date	Version	Description
March 2024	A16	• Modified the instruction description in 5.7.2.8 CMP. • Modified the instruction description in 5.7.2.9 ECMP. • Modified the instruction description in 5.7.2.11 EZCP. • Modified the instruction description in 5.7.2.11 EZCP.

Date	Version	Description
		• Modified the instruction description in 5.12.2 TCMP. • Modified the instruction description in 5.12.3 TZCP. • Modified related description in 5.15.7 ENC_StepCompare.
February 2024	A15	• Modified the instruction description in 5.5.1.2 ADD. • Modified the instruction description in 5.13.1.29 MC_SetOverride. • Modified the instruction description in 5.13.2.2 MC_CamIn. • Modified the instruction description in 5.18.4 SerialRcv. • Modified related description in 5.20.2 PID. • Modified related description in 6.2 Error Codes.
December 2023	A14	New • Added the following content to "String Instructions" in the "Instruction Description (LiteST)" section: ◦ Instruction List ◦ CLEARSTRING ◦ 5.9.17 DELETE ◦ 5.9.18 FIND ◦ 5.9.16 CONCAT ◦ 5.9.20 INSERT ◦ LEFT ◦ LEN ◦ 5.9.19 REPLACE ◦ RIGHT • Added the following content to "Array Instructions" in the "Instruction Description (LiteST)" section: Instruction List, AryMax and AryMin, 5.10.4 ArySearch, 5.10.5 SizeOfAry, 5.10.6 AryMove, 5.10.7 AryExchange, and Clear. • Added the following content to "Data Processing Instructions" in the "Instruction Description (LD and LiteST)" section: 5.7.1.24 SCALE_X, 5.7.1.25 NORM_X, 5.7.1.26 ROUND, 5.7.1.27 TRUNC, 5.7.1.28 CELL, and 5.7.1.29 FLOOR. • Added the following content to "Other Instructions" in the "Instruction Description (LD and LiteST)" section: 5.20.3 PID_AT, 5.20.4 PID_ATC, and 5.20.5 Error Codes. • Added the following content to "MC Axis Control Instructions (EtherCAT and Pulse Output)" in the "Instruction Description (LD and LiteST)" section: 5.13.1.31 MC_MoveVelocityPV, 5.13.1.32 MC_MoveAbsRelPP, and 5.13.1.33 MC_JogP.

Date	Version	Description
		Modify • Modified the following content to "Communication Instructions" in the "Instruction Description (LD and LiteST)" section: 5.18.3 SerialSend, 5.18.4 SerialRcv, 5.18.6 MB_Master, and 5.18.7 MB_Client. • Modified the following content to "Data Conversion Instructions" in the "Instruction Description (LiteST)" section: INT_TO_(TYPE) and DINT_TO_(TYPE). • Modified the following content to "MC Axis Control Instructions (EtherCAT and Pulse Output)" in the "Instruction Description (LD and LiteST)" section: 5.13.1.20 MC_Halt, 5.13.1.19 MC_Stop, Axis Error Codes, and 5.13.1.16 MC_Jog. • Modified 6.2 Error Codes in the "Appendix" section.
August 2023	A13	Corrected minor errors.
May 2023	A12	• Added the following explicit conversion instructions in the "Instruction Description (LiteST)" section: INT_TO_BYTE, DINT_TO_BYTE, BOOL_TO_BYTE, REAL_TO_BYTE, BYTE_TO_<TYPE>, and TO_BYTE. • Added the following communication instructions: MC_GearInPos, SerialSend, SerialRcv, MB_Master, and MB_Client. • Corrected some instruction errors.
November 2022	A11	• Added LiteST expression for some instructions. • Added the following instructions: ENC_SetLineRotationMode, ENC_SetUnit, and MC_SetOverride. • Corrected some instruction errors.
September 2022	A10	• Added some EIP communication instructions. • Added the following instructions: SORTC, DSORTC, SORTR, DSORTR, RAMP, and DRAMP. • Added description of Easy series PLC instructions. • Corrected some instruction errors.
May 2022	A09	Made minor changes.
August 2021	A08	• Added the following instructions: DECO, DDECO, ENCO, and DENCO. • Added the following instructions: ETC_RestartMaster, MC_FollowVelocity, MC_SetAxisConfigPara, and MC_DigitalCamSwitch. • Updated the list of instructions. • Updated the list of error codes.
May 2021	A07	Kept the material versions consistent.

Date	Version	Description
March 2021	A03	• Corrected errors in the previous version. • Added cam instructions and details. • Added serial encoder axis instructions and details. • Updated the list of error codes.
August 2020	A02	Corrected errors in the previous version.
June 2020	A01	• Added axis group instructions and details. • Updated the list of error codes. • Corrected errors in the previous version.
December 2019	A00	First release.

3 Instruction Overview

3.1 Instruction Structure

3.1.1 LD Instruction Structure

An LD instruction consists of the opcode and operand.

- Opcode: Instruction function description
- Operand: Data used in the instruction

The operand includes the input data, output data, and numeric data.

Input (S)

The input indicates data used in the operation.

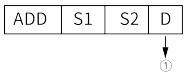
The usage of input data is described as follows according to the variables and elements specified in each instruction.

Category	Description
Constant	A constant specifies a numerical value used in the operation. It cannot be changed during program execution since it is configured when the program is created.
Soft element and variable	During program execution, data used in the instruction can be changed by modifying data stored in the specified element.

Output (D)

The output element stores data after operations. Sometimes, data used in the operation needs to be stored in the target before the operation, depending on the specific instruction.

The following is an example of the addition operation of INT data.



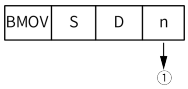
①: Only the operation result is stored.

The D element must be configured with a variable or element for storing data.

Number of elements, transfers, data entries, and strings (n)

In instructions involving multiple elements, the number of repetitions, the number of groups of data to be processed, and the number of strings, the numbers of elements, transfers, data entries, and strings used are determined by n.

The following is an example of a block transfer instruction.



①: The data to be transferred is specified by the BMOV instruction.

3.1.2 LiteST Instruction Structure

A LiteST instruction consist of an instruction name, parameters, and return values, which are defined as follows:

- Instruction name: Instruction function description
- Parameters: Data used in the instruction
- Return values: Result obtained after the instruction is executed

Input (S)

The input indicates data used in the operation.

The usage of input data is described as follows according to the variables and elements specified in each instruction.

Category	Description
Constant	A constant specifies a numerical value used in the operation. It cannot be changed during program execution since it is configured when the program is created.
Element and variable	During program execution, data used in the instruction can be changed by modifying data stored in the specified element.

Output (D)

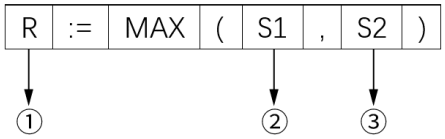
The output element stores data after operations. Sometimes, data used in the operation needs to be stored in the target before the operation, depending on the specific instruction.

Return Value (R)

The return value is the result obtained after the instruction is executed. Sometimes, data used to operate on the instruction result needs to be stored in the target after the operation, depending on the specific instruction.

The following is an example of the comparison operation of INT type data:

R:=MAX(S1, S2)



MAX(S1, S2)

①: Only the operation result

②: Input parameter

③: Input parameter

3.1.3 Lists of Soft Elements and Variables

The PLC supports bit soft elements, word soft elements, special soft elements, variables, arrays, structures, and custom variables.

Bit Soft Elements

Type	Range	Number of Points	Data Type	Description
X	X0 to X1777	1024, octal	BOOL	Not retained upon power failure
Y	Y0 to Y1777	1024, octal	BOOL	Not retained upon power failure
M	M0 to M7999	8000	BOOL	M0 to M999 not retained upon power failure, M1000 to M7999 retained upon power failure
S	S0 to S4095	4096	BOOL	S0 to S999 not retained upon power failure, S1000 to S4095 retained upon power failure
B	B0 to B32767	32768	BOOL	B0 to B999 not retained upon power failure, B1000 to B32767 retained upon power failure

Word Soft Elements

Type	Range	Number of Points	Data Type	Description
D	D0 to D7999	8000	BOOL, INT, DINT, and REAL	D0 to D999 not retained upon power failure, D1000 to D7999 retained upon power failure
R	R0 to R32767	32768	BOOL, INT, DINT, and REAL	R0 to R999 not retained upon power failure, R1000 to R32767 retained upon power failure
W	W0 to W32767	32768	BOOL, INT, DINT, and REAL	W0 to W999 not retained upon power failure, W1000 to W32767 retained upon power failure

Special Soft Elements

Type	Function	Range	Number of Points	Description
L	Jump label	L0 to L1023	1024	Used in combination with the CJ and LBL instructions
K	Decimal	K-32, 768 to K32, and 767 (16-bit) K-2, 147, 483, 648 to K2, 147, 483, and 647 (32-bit)	-	-
H	Hexadecimal	H0000 to HFFFF (16-bit)	-	-

Type	Function	Range	Number of Points	Description
		H00000000 to HFFFFFFFF (32-bit)		
E	Floating-point number or real number	$-1.0*2e^{128}$ to $-1.0*2e^{-126}$, 0, $1.0*2e^{-126}$ to $1.0*2e^{128}$ (32-bit)	-	-
Character	Character or string	-	-	Used as instruction parameters

- M8000 and D8000 Special Soft Elements

Special Soft Elements	Description	Access Permissions
M8000	ON during running of the user program	Read-only
M8001	Negated M8000 state	Read-only
M8002	ON in the first operation cycle of the user program	Read-only
M8003	Negated M8002 state	Read-only
-	-	-
M8011	Free-run clock with a cycle of 10 ms	Read-only
M8012	Free-run clock with a cycle of 100 ms	Read-only
M8013	Free-run clock with a cycle of 1s	Read-only
M8014	Free-run clock with a cycle of 1 minute	Read-only
-	-	-
M8020	Zero flag	Read-only
M8021	Borrow flag	Read-only
M8022	Carry flag	Read-only
M8029	Multi-cycle instruction execution completion flag, applicable to the RAMP, SORTC, and SORTR instructions	Read-only
-	-	-
M8040	SFC, SFC disabling flag	Read-write
-	-	-

Special Soft Elements	Description	Access Permissions
M8161	OFF: 16-bit mode; ON: 8-bit mode; Bit processing mode for ASCII/HEX/CCD/LRC/CRC/RS	Read-write
M8163	BINDA instruction output character switchover flag (retained or switched to 0000h)	Read-write
M8165	SORTR instruction ascending sort enabling flag	Read-write
M8168	SMOV instruction data format setting: OFF-BCD mode or ON-HEX mode	Read-write
M8333	Flag indicating all BKCMP instruction matrix comparison results are 1	Read-only

- Other undefined elements after M8000 cannot be used in the programs.

Special Soft Elements	Description	Access Permissions
D8066	Critical errors in user programs and instructions (triggered, not reset)	Read-only
D8067	Minor errors in user programs and instructions (triggered, not reset)	Read-only

NOTE

Access permissions:

- Read-only: The PLC output is read-only by the user. Data written by the user will be overwritten.
- Read-write: The PLC input can be read and written by the user.

Custom Variables

Type	Range	Capacity	Data Type	Description
Pointer	-	4096 points (32-bit)	BOOL, INT, DINT, and REAL arrays	Not retained upon power failure
BOOL	-	2 MB (8-bit)	BOOL, BYTE, INT, DINT, REAL, IP, and STRING variables BOOL, BYTE, INT, DINT, REAL, IP, and STRING arrays BOOL, BYTE, INT, DINT, REAL, IP, and STRING compound structures	256 KB data retained upon power failure, other data not retained upon power failure
BYTE				
INT				
DINT				
REAL				

Type	Range	Capacity	Data Type	Description
IP				
STRING				

3.2 Soft Elements

3.2.1 Bit Soft Elements

The PLC supports bit elements. The following table describes the type, range, number of points, and description of bit elements.

Data Type	Range	Number of Points	Data Type	Description
X	X0 to X1777	1024, octal	BOOL	Input
Y	Y0 to Y1777	1024, octal	BOOL	Output
M	M0 to M7999	8000	BOOL	M0 to M999 are not retained upon power failure. M1000 to M7999 are retained upon power failure.
S	S0 to S4095	4096	BOOL	S0 to S999 are not retained upon power failure. S1000 to S4095 are retained upon power failure.
B	B0 to B32767	32768	BOOL	B0 to B999 are not retained upon power failure. B1000 to B32767 are retained upon power failure.

NOTE

- Bit elements of the data type BOOL can be used as input or output parameters in instructions.
- X and Y can be used as input and output elements respectively.
- Only M1 to M7999 can be used in ordinary programming. M8000 and some M elements after it can be directly used in programming to provide special functions. For details about the special functions of M8000, see the "Special Soft Elements" section.

3.2.2 Word Soft Elements

The PLC supports word elements. The following table describes the specific type, range, number of points, and description of word elements.

Type	Range	Number of Points	Data Type	Description
D	D0 to D7999	8000	BOOL, INT, DINT, and REAL	D0 to D999 are not retained upon power failure. D1000 to D7999 are retained upon power failure.
R	R0 to R32767	32768	BOOL, INT, DINT, and REAL	R0 to R999 are not retained upon power failure. R1000 to R32767 are retained upon power failure.
W	W0 to W32767	32768	BOOL, INT, DINT, and REAL	W0 to W999 are not retained upon power failure. W1000 to W32767 are retained upon power failure.

- The range of words that are retained upon power failure cannot be changed.
- A word element can be used as an integer or floating-point number. As an element does not have the data type attribute, whether an element is used as an integer or floating-point number is determined by a parameter in the corresponding instruction.
- When a word element is used as an integer, it is a 16-bit or 32-bit integer based on the instruction parameter. When used as a 16-bit integer, a word element occupies one element. When used as a 32-bit integer, a word element occupies two elements.
- When used as a floating-point number, a word element occupies two elements.

Example

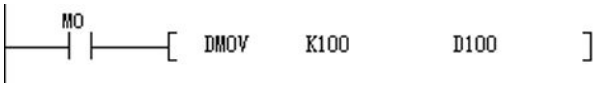
- Word element used as a 16-bit integer

Use the 16-bit assignment instruction to assign the value 100 to the word element D100, which occupies D100.



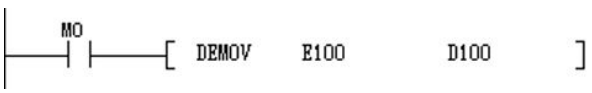
- Word element used as a 32-bit integer

Use the 32-bit assignment instruction to assign the value 100 to the word element D100, which occupies D100 (low-order) and D101 (high-order).



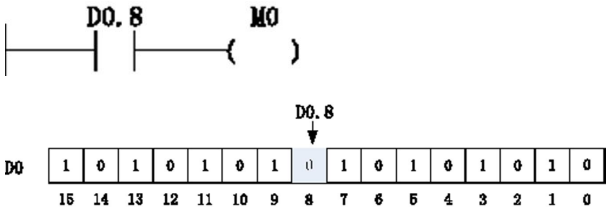
- Word soft element used as a floating-point number

Use the floating-point instruction to assign the value 100 to the word element D100, which occupies D100 and D101.



- Bit-based operation on word elements

Bit-based operations on word elements can be implemented by using a dot (.). For example, writing D0.8 during programming indicates an operation on the 8th bit of the word element D0.



The bits of the word element are counted from the 0th bit. When the 8th bit of D0 is 0, the output M0 is OFF. When the 8th bit of D0 is 1, the output M0 is ON.

3.2.3 Special Soft Elements

The PLC supports special elements. The following table describes the type, range, function, and description of the special elements supported by the PLC.

Type	Function	Range	Number of Points	Description
SBR	Subprogram label	SBR0 to SBR1023	1024	Used by the CALL instruction. Subprograms can be set as common subprograms or encrypted subprograms, which share the capacity of the system program area.
L	Jump label	L0 to L1023	1024	Used in combination with the CJ and LBL instructions
I	External interrupt	-	4	Interrupt label, which is on the rising edge, falling edge, or rising and falling edges of X port
	Timer interrupt	-	4	Timing duration, in ms
	Compare interrupt	-	16	Limited by the number of internal encoder axes (high-speed counters)
K	Decimal	K-32,768 to K32,767 (16-bit); K-2,147,483,648 to K2,147,483,647 (32-bit)	-	-
H	Hexadecimal	H0000 to HFFFF (16-bit); H00000000 to HFFFFFFFF (32-bit)	-	-
E	Floating-point number or real number	-3.402823e+38 to -1.175495e-38; 0; +1.175495e-38 to +3.402823e+38	-	Up to 7 decimal significant digits for a single-precision floating-point number (The excess part is automatically rounded off.)
Character	Character or string	-	-	Used as instruction parameters

A single-precision floating-point number has a maximum of 7 significant decimal digits. If the 9-bit binary floating-point number 1234567.89 is transmitted to the destination location D0, the actual value of D0 is 1234567.9. The precision is reduced.



3.3 Variables

3.3.1 Overview

In a PLC programming system, in addition to using direct addresses, such as the X, Y, M, D, R and other elements, for programming, you can also use variables without specific storage addresses for programming to implement the required control logic, or the complete control process of the application object, so as to facilitate code compiling and improve the readability of the code. The following table shows supported custom variables.

Type	Capacity	Data type	Description
Pointer	4096 points (32-bit)	BOOL/INT/DINT/REAL	Pointer variablenot retentive upon power failure
BOOL	2 MB (8-bit)	BOOL/INT/DINT/REAL/IP/STRING/BYTE variable BOOL/INT/DINT/REAL/IP/STRING/BYTE array BOOL/INT/DINT/REAL/IP/STRING/BYTE compound structure	256 KB data retentive upon power failure Other data not retentive power failure
INT			
DINT			
REAL			
IP			
STRING			
BYTE			

3.3.2 Defining a Variable

The PLC supports custom variables. You can define a global variable and directly use the variable name during programming. Abide by the following rules when naming a global variable:

- It contains only letters, digits, and underscores (_) and does not start with a digit or underscore (_).
- It is not the same as the name of an element, constant, standard data type, instruction, subprogram, or interrupt subprogram.
- It cannot be keywords such as ARRAY, TRUE, FALSE, ON, OFF, and NULL.

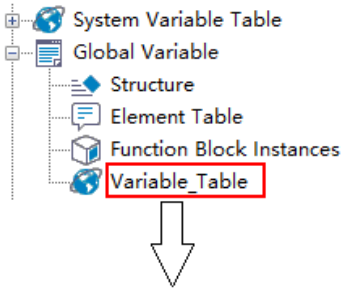
Variable Data Types

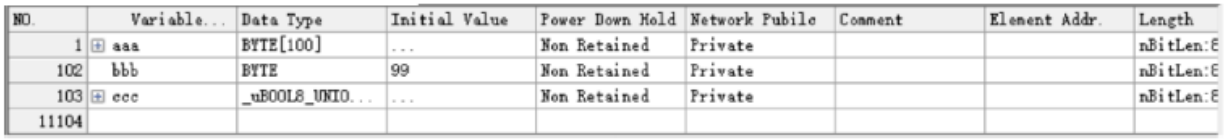
The data type of variables can be STRUCT and array. The following table lists the supported data types.

Data Type	Description
BOOL	Boolean
INT	Single-word integer
DINT	Double word integer
REAL	Real number
STRING	String
IP	IP
BYTE	Byte

Defining Global Variables

You can add, delete, and edit variables in **Global Variable** of the project management window.





NO.	Variable...	Data Type	Initial Value	Power Down Hold	Network Pubilc	Comment	Element Addr.	Length
1	aaa	BYTE[100]	...	Non Retained	Private			nBitLen:8
102	bbb	BYTE	99	Non Retained	Private			nBitLen:8
103	ccc	_uBOOL8_UNIO...	...	Non Retained	Private			nBitLen:8
11104								

1. Add a variable table and variables. Right-click **Global Variable** and choose **New Global Variable Table** to create a global variable table.



2. Double-click the variable table to go to the variable editing interface.
   - Edit a variable: Double-click the text box to edit the variable or click the drop-down list box to select the variable.
   - Add a variable: Right-click the variable and choose **Insert Row(&I)**.
   - Delete a variable: Right-click the row to be deleted and choose **Delete Row(&L)**.

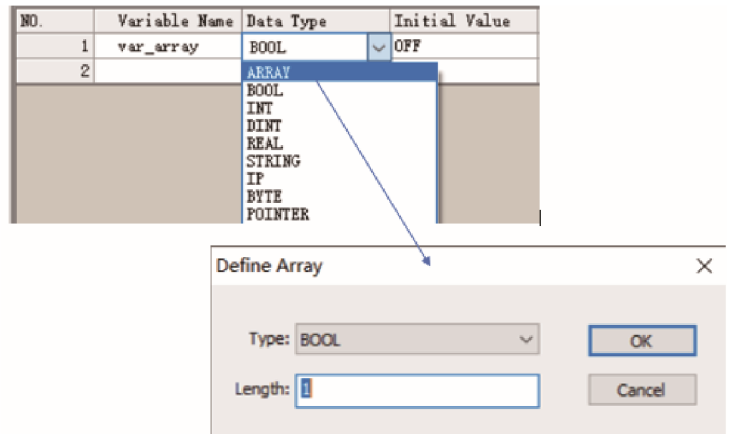
NO.	Variable ...	Data Type	Initial Value	Power Down Hold	Network Pubilc
1	Axis0	HMI_AxisContr...	...	Non Retained	Private
24	Axis1	HMI_AxisContr...	...	Non Retained	Private
47	bMovVel	BOOL	OFF	Non Retained	Private
48	bmovStart	BOOL	OFF	Non Retained	Private
49					

Parameter Name	Description
Variable name	Custom variable name. You can directly use the variable name for programming.
Data type	The data types include BOOL, INT, DINT, REAL, IP, STRING, and BYTE variables, BOOL, INT, DINT, REAL, IP, STRING, and BYTE arrays, and BOOL, INT, DINT, REAL, IP, STRING, and BYTE structures. If the data type is an array, you can set the type and length of the array variable in the displayed dialog box. If the data type is a pre-defined structure, you can define a structure variable.
Initial value	You can assign an initial value to a variable. For arrays and structures, the initial value of each element can be specified individually.
Power down hold	Variables can be retained or not retained upon power down. The specified initial value is valid only when the variable is non retained upon power down.
Network Public	This variable can be set to **Private**, **Public**, or **In/Out**. For structure, specific union, structure array, and specific union array variables, this variable must be set to **Private**. When this variable is set to **Public**, a label configuration file named **LabelConfig.xml** is generated in the **InteractiveFile** folder under the project directory after project compiling. You can import this configuration file into third-party software to enable label communication.

3.3.3 Defining an Array

During user programming, if the data type is set to ARRAY, an array can be defined.

1. Select the type and length of the array variable in the displayed dialog box and click OK to define an array.



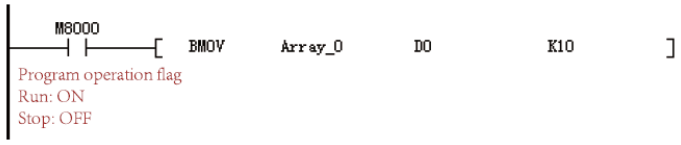
2. Click + next to the array variable to edit the initial values and comments of member variables.

NO.	Vari...	Data Type	Initial Value	Power Down Hold	Network Public	Comment	Element Addr.	Length
1	aaa	BOOL	OFF	Non Retained	Private			nBitLe
2	bbb	BYTE	99	Non Retained	Private			nBitLe
3	ccc	_uBOOL8_UNIO...	...	Non Retained	Private			nBitLe
4	ccc[0]	_uBOOL8_UNIO...	...					
5	ab	BOOL[8]	...					
6	ab[0]	BOOL	OFF					
7	ab[1]	BOOL	OFF					
8	ab[2]	BOOL	OFF					
9	ab[3]	BOOL	OFF					
10	ab[4]	BOOL	OFF					
11	ab[5]	BOOL	OFF					
12	ab[6]	BOOL	OFF					
13	ab[7]	BOOL	OFF					

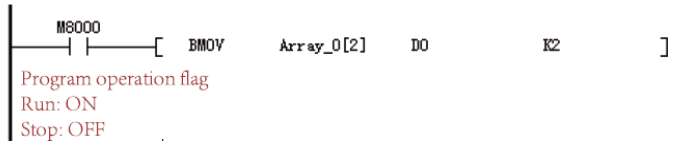
When an array is used in an instruction, if the array subscript is not specified, the access starts from the first element of the array. If the array subscript is specified, the access starts from the element specified by the subscript.

The following are two examples.

- Assign Array_0[0] to Array_0[9] to D0 to D9.



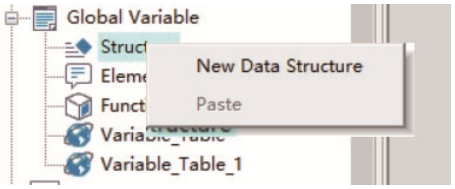
- Assign Array_0[2] and Array_0[3] to D0 and D1.

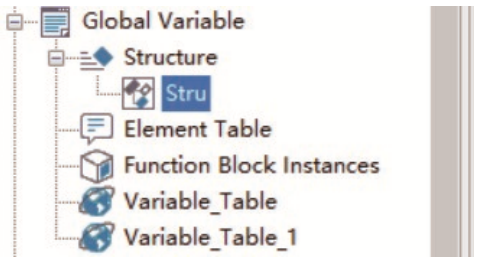


3.3.4 Defining a Structure

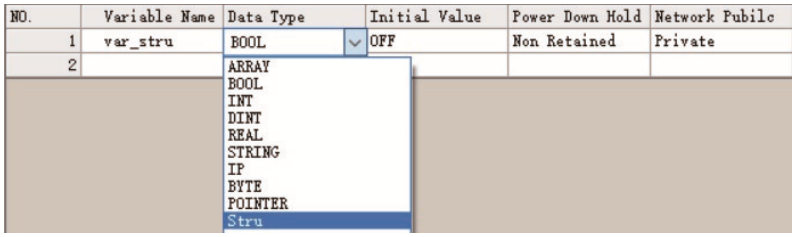
To define a structure variable, you need to define the data structure of the structure in advance.

- Right-click **Structure** under **Global Variable**, select **New Data Structure**, and enter a structure name. The structure is defined. When defining a variable in the variable table, you can select a variable with the data type STRUCT to define the variable as a structure variable.





- After the structure and member variables are created, you can select **Stru** in the **Data Type** column to define a structure variable.



- Click the **Initial Value** column of the structure variable to set the initial values of structure variable members.

3.3.5 Defining an Enumerated Variable

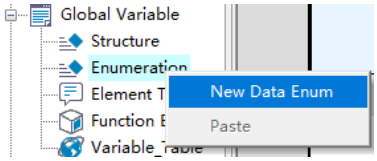
Prerequisites

- Firmware version: V6.5.0.0 and later
- Software version: AutoShop V4.11.0.0 and later

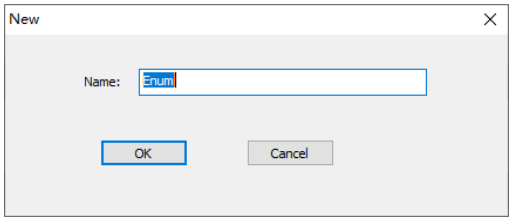
Introduction

To define an enumerated variable, you need to define the data structure of the enumerated variable in advance.

1. Choose "Global Variable" > "Enumeration" in the left-side project management tree and select "New Data Enum". When defining a variable in the variable table, you can select a variable with the data type Enum to define the variable as an enumerated variable.



2. In the "New" dialog box, enter the enumeration name and click "OK".

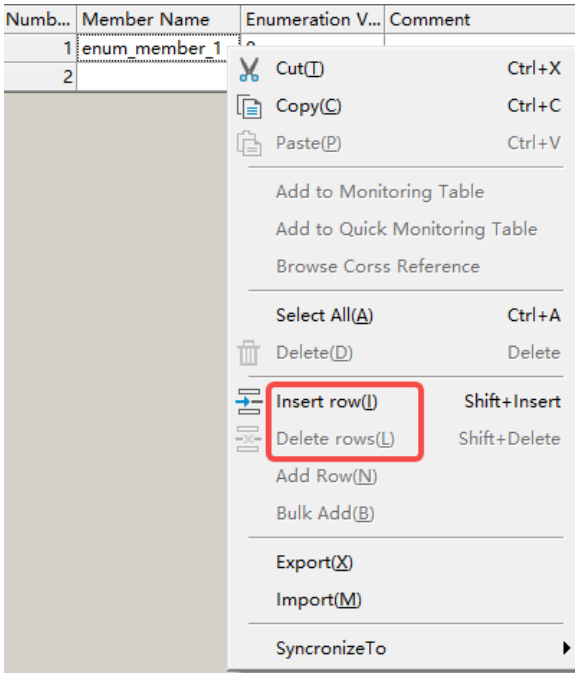


3. The "Enum" sheet displays enumerated variable members of the Enum structure. By default, the member "enum_member_1" and its value of 0 exist.

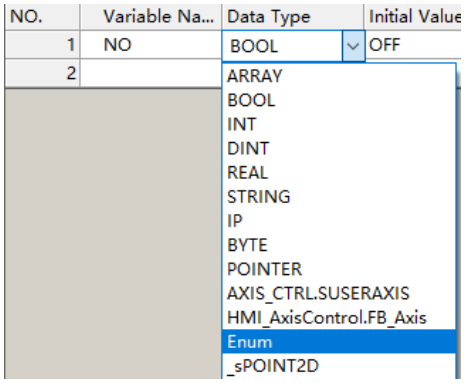
Numb...	Member Name	Enumeration V...	Comment
1	enum_member_1	0	
2			

Right-click on the sheet and select "Insert row" in the pop-up menu, or double-click a blank line on the sheet to automatically create an enumeration member. The value of the member increments by 1

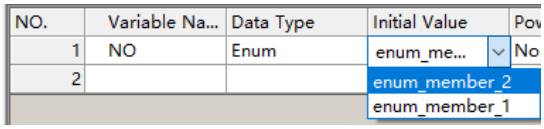
automatically. To delete an enumeration member, right-click the target line and select "Delete rows".



4. After an enumeration table is created and members are added to the table, select "Enum" from the "Data Type" column and define enumerated variables.



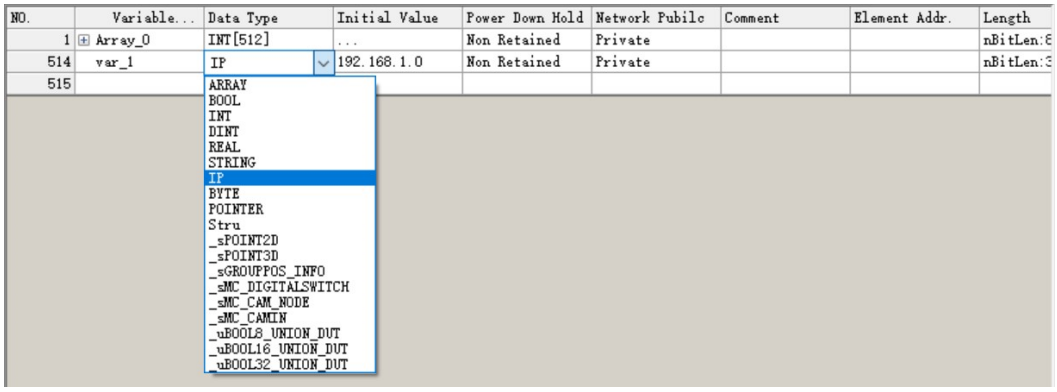
5. You can select a value for an enumerated variable from the "Initial Value" drop-down list. If no value is selected, the first option in the drop-down list is used by default.



3.3.6 Defining an IP Variable

You can define IP variables in the variable table or program. An IP variable occupies 32 bits, and the default value is 192.168.1.0.

- Select **IP** from the **Data Type** drop-down list of the variable table.



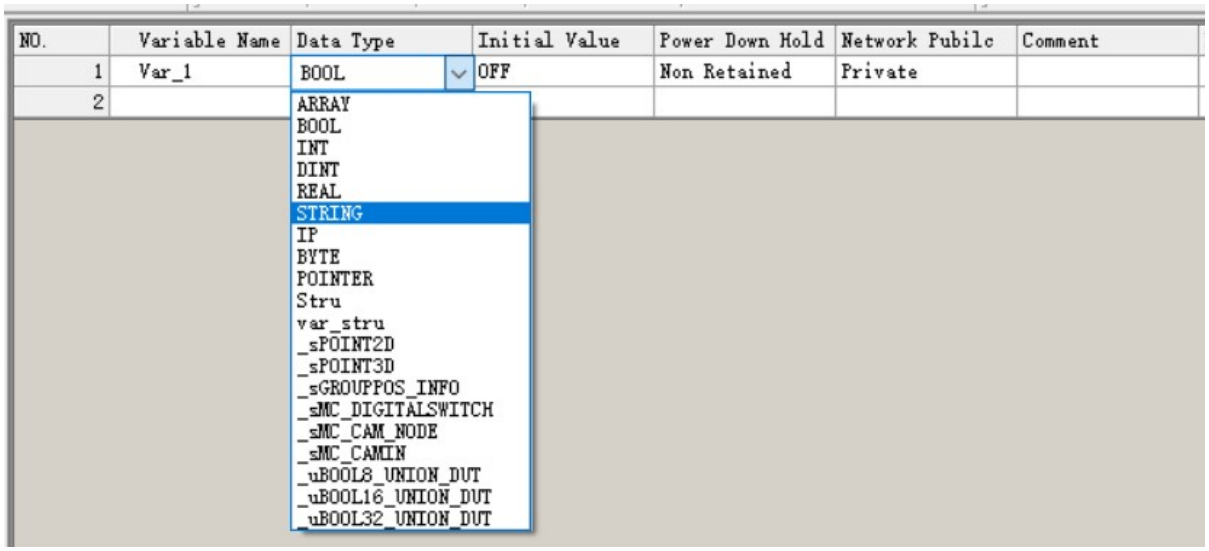
- Use an IP variable in the LiteST program, and assign a value to the IP variable by using single quotation marks.

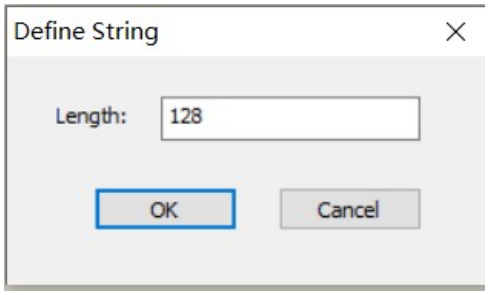
```
1 var_1:='10.45.121.90';
```

3.3.7 Defining a String

You can define string variables in the variable table or program.

- Select STRING from the Data Type drop-down list of the variable table, and set the length of the string in the displayed dialog box.
  The default length is 128 bytes and the maximum length is 256 bytes. The last byte is the terminator by default.





- Use a string variable in the LiteST program, and assign a value to the string variable by using single quotation marks.

```
1 var_1:='abc';
```

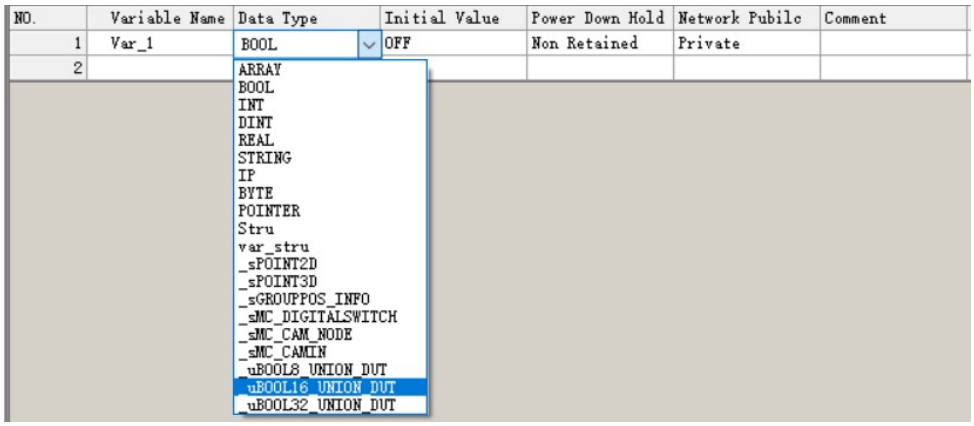
3.3.8 Defining a Specific Union

A specific union is similar to a structure in that they both are collections of different types of elements.

The difference lies in the fact that each member of a structure has its own independent storage space, while the members of a specific union share the same memory space. That is why a specific union is also called a union. Members of a specific union are not used simultaneously, but rather one after another.

You can define specific union variables in the variable table or program. There are three types of specific union variables: _uBOOL8_UNION_DUT, _uBOOL16_UNION_DUT, and _uBOOL32_UNION_DUT. The lengths of the three types are 1 byte, 2 bytes, and 4 bytes, respectively.

- Select the type of the required specific union variable from the **Data Type** drop-down list of the variable table.



Take _uBOOL32_UNION_DUT as an example. Create a variable in the variable table, and select **_uBOOL32_UNION_DUT** from the **Data Type** drop-down list.

NO.	Variab...	Data Type	Initial Value	Power Down Hold	Network Pubilc	Comment	Element Addr.	Length
1	var_1	_uBOOL32_UNI...	...	Non Retained	Private		D0	nBitLen:3
2	ab	BOOL[32]	...					
35	ai	INT[2]	...					
36	ai[0]	INT	0				D0	
37	ai[1]	INT	0				D1	
38	abyte	BYTE[4]	...					
39	abyte[0]	BYTE	0				D0	
40	abyte[1]	BYTE	0				D0.8	
41	abyte[2]	BYTE	0				D1	
42	abyte[3]	BYTE	0				D1.8	
43	byte0	BYTE	0				D0	
44	byte1	BYTE	0				D0.8	
45	byte2	BYTE	0				D1	
46	byte3	BYTE	0				D1.8	
47	i0	INT	0				D0	
48	i1	INT	0				D1	
49	f0	REAL	0.000000				D0	
50	di0	DINT	0				D0	
51								

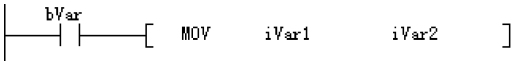
- In a program, you can access different members of a specific union variable by using the dot operator ("."). This allows you to parse variables in different scenarios.

```
IF eDataType=1 THEN
    var_1.byte0 := 200;
ELSIF eDataType=2 THEN
    var_1.i0 := 2000;
ELSIF eDataType=3 THEN
        var_1.f0 := 2000.22;
END_IF;
```

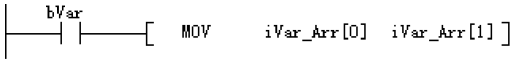
3.3.9 Using a Variable

After a variable is defined, you can directly use the variable name for programming without assigning elements.

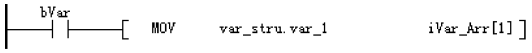
- When a common variable is used, directly use the variable name during programming.



- When an array variable is used, use [Number] to indicate an array element during programming. The number starts from 0.



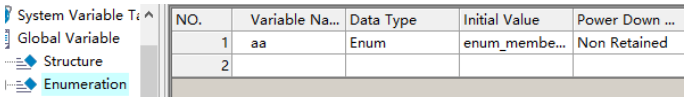
- When a structure variable is used, use Structure variable name.Member variable to indicate a structure member during LiteST programming.



- Pay attention to the following points when using an enumerated variable:
  - Directly access the corresponding enumerated member during programming.

```
1    aa:=enum_member_2;
```

    If aa is a variable of the Enum type, it must be defined in the variable table. Note that if aa is of another type, such as INT or DINT, enumerated value assignment is not supported.



  - For a variable of the Enum type, only the enumerated values can be assigned.

- When an IP or string variable is used, use a value enclosed in a pair of single quotation marks ('Value') to indicate the value of the variable.

```
1  var_1:='10.45.121.90';
```

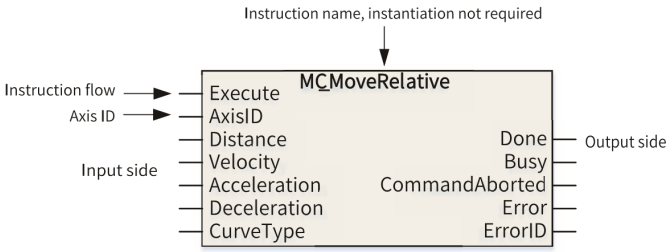
```
1  var_1:='abc';
```

For BYTE, INT, and DINT variables and arrays, you can perform bit operations using the syntax "Variable_name.digit" in programming. For details, see 3.2.2 Word Soft Elements.

3.4 Graphic Block Instructions

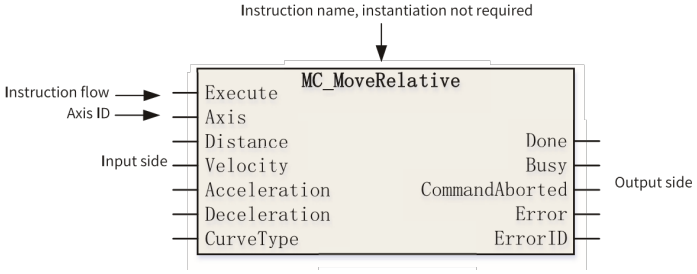
3.4.1 Instruction Structure

Some instructions support graphic block programming. A graphic block instruction is composed of the instruction name, flow signal, input, and output. The following figure shows the composition of a graphic block instruction of a motion control axis.



The floating-point numbers such as the target position and target velocity in the instructions are single-precision floating-point data. Therefore, the values in the instructions must meet the requirements on the range and precision of single-precision floating-point data when being processed in the PLC program. Their values must fall between -3.4E38 and +3.4E38, with a maximum of 7 significant digits. If a value has more than 7 significant digits, the excess part is automatically rounded.

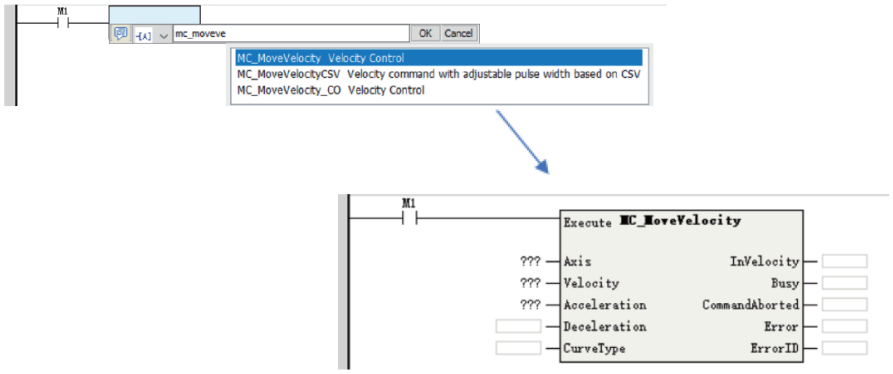
Since AutoShop 4.0.0.0 with PCB software 3.0.0.0, the motion control axis control instructions (EtherCAT or pulse output, pulse input) of graphic blocks support access by axis name. In these instructions, AxisID is changed to Axis, and access by axis ID is still supported.



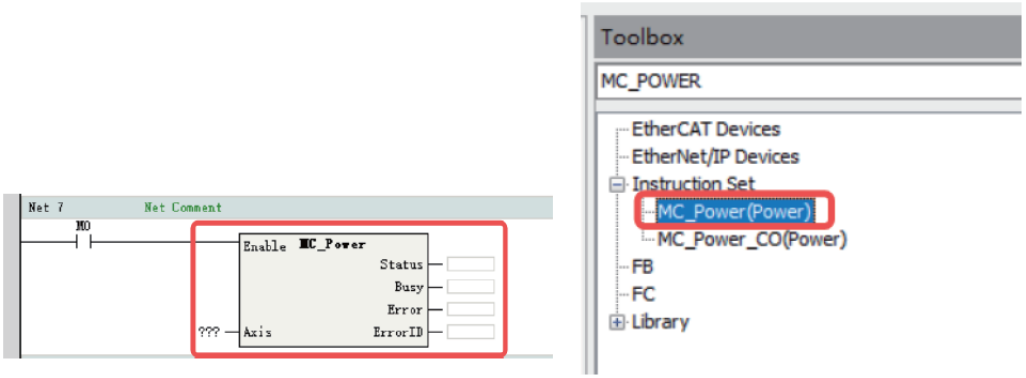
3.4.2 Instruction-Level Programming

During user programming, you only need to enter the name of a graphic block instruction and press **Enter** to add the graphic block instruction to the program network. You can directly edit the instruction parameters.

- When you edit a ladder diagram (LD), enter an instruction name or select an instruction name according to the instruction prompt and click **OK**. The graphic block instruction is added to the LD network.



- Enter parameters in the graphic block instruction to complete editing of the graphic block instruction. In the instruction, parameters (with "???") next to ① are mandatory, and parameters next to ② are optional. If a parameter is not specified, the default parameter value is used automatically in the instruction input, and the state cannot be obtained in the instruction output in the program or during monitoring and debugging.
- All instructions under **Instruction Set** on the **Toolbox** pane are in graphic block mode. During programming, you can directly double-click an instruction under **Instruction Set** to add the instruction to the current focus position of the LD.



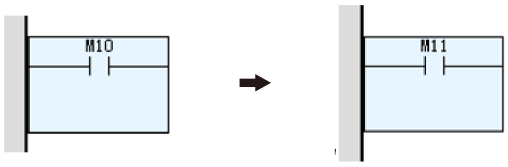
①: Double-click an instruction to add it to the LD. ②: The instruction is added successfully.

3.4.3 Label Number Function

Graphic blocks can be used to quickly increase or decrease label numbers and implement incremental paste.

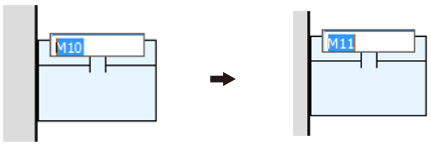
Quickly Increasing or Decreasing Label Numbers

When editing the ladder diagram (LD), you can press **Alt** + ↑ or ↓ to quickly increase or decrease the label number of an element or array subscript.

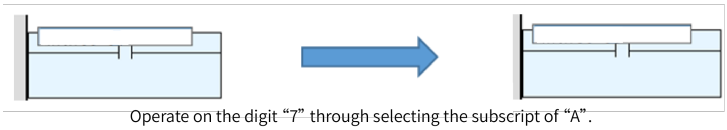


Select M10 and press **Alt** + ↑ to change it to M11.

- This function can be used during command editing.



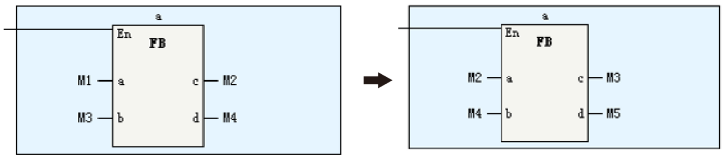
- For complex array variables, you can select the array subscript that needs to be increased or decreased.



Operate on the digit “7” through selecting the subscript of “A”.

Select the subscript of A to increase or decrease the digit 7.

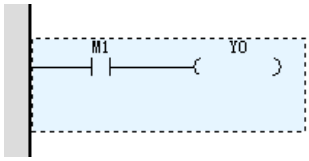
- When a function block is selected, the operation applies to all pins in the function block.



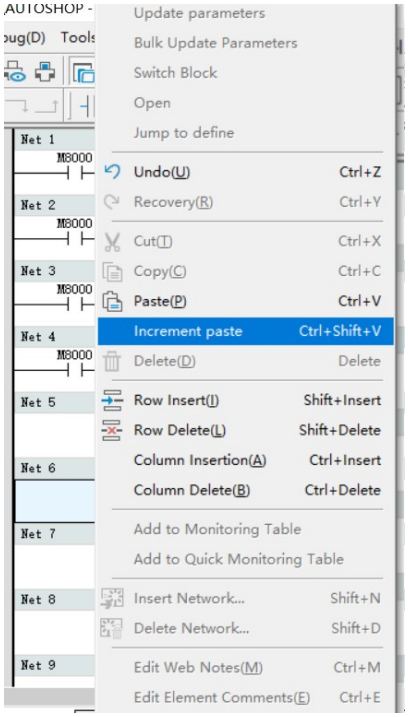
Incremental Paste

When editing the LD, you can use the incremental paste function to continuously paste the copied elements for multiple times. At the same time, the element number or array subscript can be specified during the process.

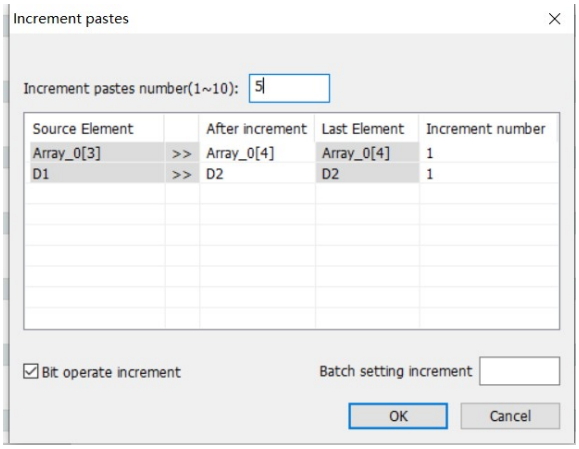
1. Select an element in the LD and press **Ctrl** + **C** or right-click the element and choose **Copy**.



2. Right-click the destination position and choose **Increment paste** from the shortcut menu (or press **Ctrl** + **S hift** + **V**).

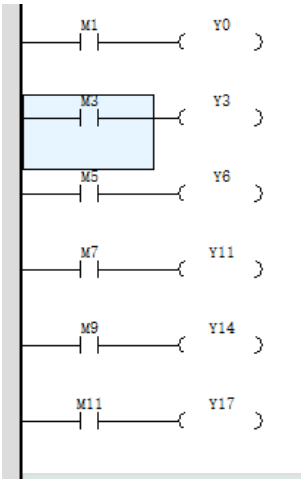


3. Specify the increment value and paste times in the displayed configuration window.



- Incremental pastes number(1~10): You can set the paste times.
- After increment: You can enter the expected value after increment, and Increment number is automatically calculated based on this value.
- Increment number: You can set the increment in the target element each time a paste operation is performed.
- Bit operate increment: During bit operation of an element, if this option is selected, the increment applies to the bits in the target element.
- Batch setting increment: **Y**ou can set the increment numbers in batches.

4. Click OK. The paste operation is performed based on the configuration.



3.5 Function Block and Function

3.5.1 Function Block

Introduction

A function block (FB) abstractly encapsulates the part used repeatedly in a program into a general program block that can be called repeatedly within the program.

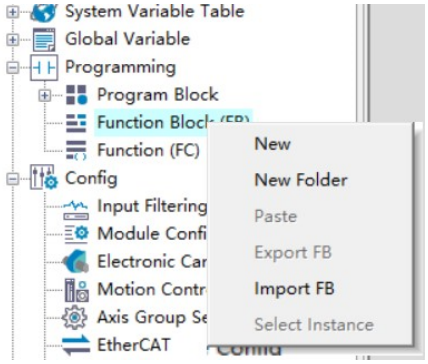
Using encapsulated function blocks in programming can improve program development efficiency, reduce programming errors, and improve program quality.

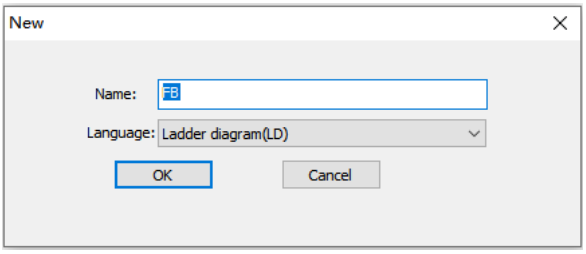
Different instances can be created based on the same function block. These instances can output one or more values during execution. The system allocates memory for internal variables of each instance, and these variables describe the running state of the function block. With the same input parameters, different instances provide different calculation results.

The basic steps of using a function block are as follows: Create a function block > Program the function block > Instantiate the function block > Run the function block > Encapsulate the function block > Import the function block.

Creating a Function Block

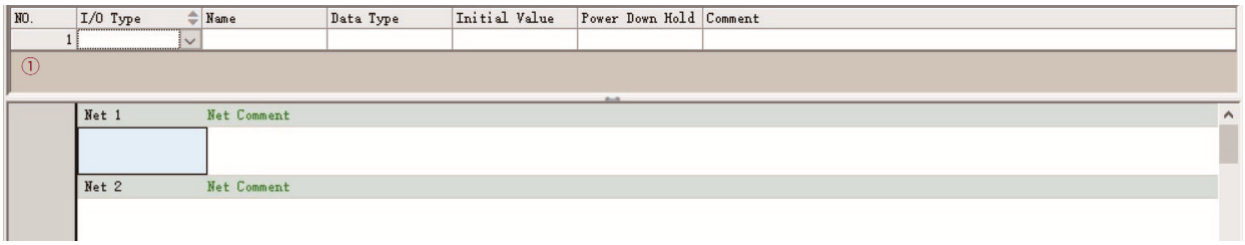
Expand the Programming node in the project management window, right-click Function Block (FB), or right-click a folder under Function Block (FB), choose New. In the dialog box that is displayed, enter a name, select a language, and click OK. A function block is created successfully.

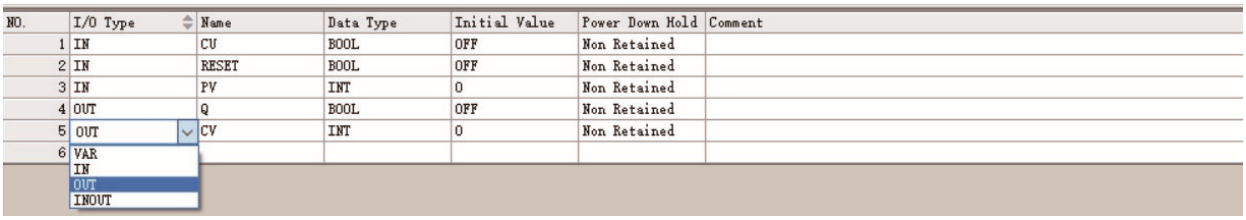




Programming the Function Block

Double-click the created function block under Function Block (FB) to go to the function block program editing interface. Compared with ordinary program editing, the function block program editing interface has two additional windows for defining input/output and local variables.





①: Windows for defining input/output and local variables

1. I/O Type: attribute of the function block variable

Option	Name	Description
IN	Input variable	The parameter is provided by the function block that calls the variable, and is transferred to the instruction of the function block.
OUT	Output variable	The parameter is provided to the function block that calls the variable. It is the structure data is output from the function block.
INOUT	Input/Output variable	Input/Output variables can be transferred to the called function block and modified within the called function block.
VAR	Local variable	A local variable is only valid in the current function block and cannot be accessed externally.

2. Name: name of the variable

3. Data Type
   The supported data types include BOOL, INT, DINT, REAL, BYTE, IP, and STRING. You can also define array variables and structure variables. To use structure variables, you need to create structure members in the structure of global variables.

4. Initial Value
   You can set the initial value of a variable when execution starts.

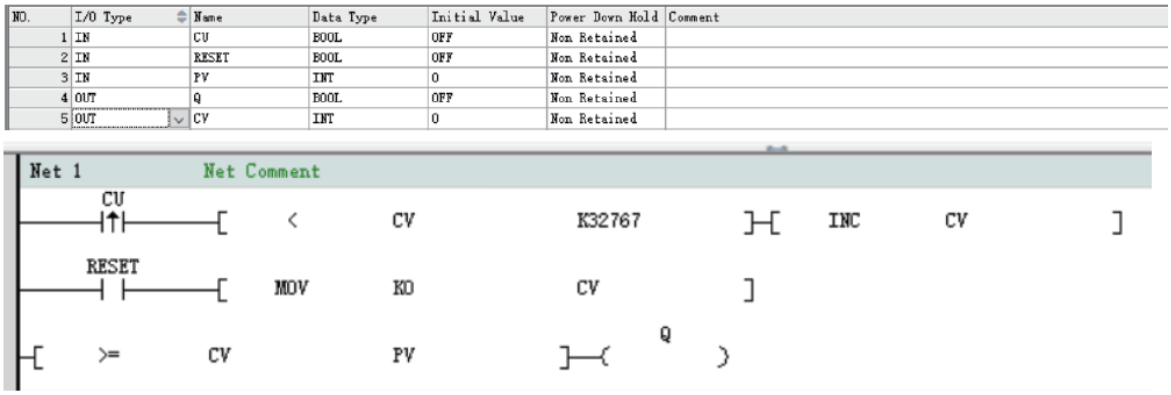
5. Power Down Hold

This attribute allows you to choose whether to retain the value of a variable upon power failure.

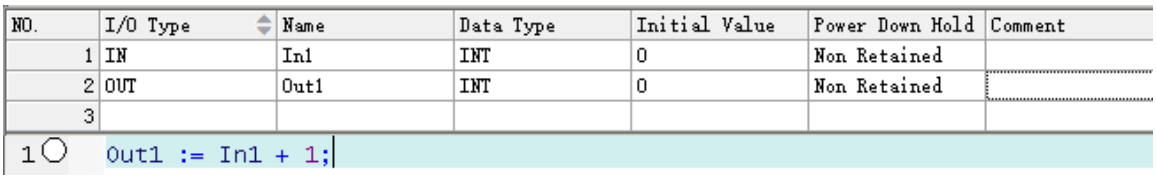
- Non Retained: The variable resumes the specified initial value after power-on.
- Retained: If you select Re-initialize retentive variables when downloading, the variable resumes the specified initial value during program downloading. Otherwise, it retains the previous value.

The function block can call functions (FCs) or function blocks and supports up to 8 levels of nested calls. In addition to variables, the function block program can also use supported elements, such as M8000, as global variables.

Example 1: Counting Up with Function Block Encapsulation (LD)



Example 2: Counting Up with Function Block Encapsulation (LiteST)

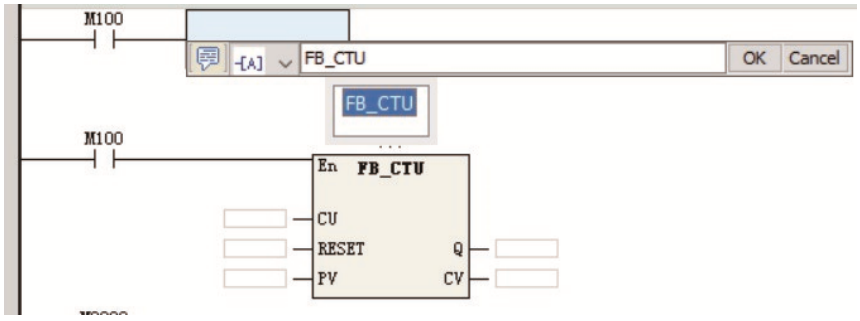


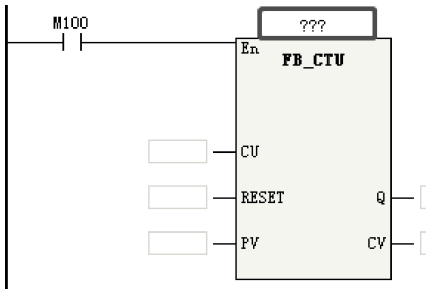
Instantiating and Calling the Function Block

After the function block program is compiled, it needs to be instantiated and called to be used in a ladder diagram (LD) or structured text (LiteST) application.

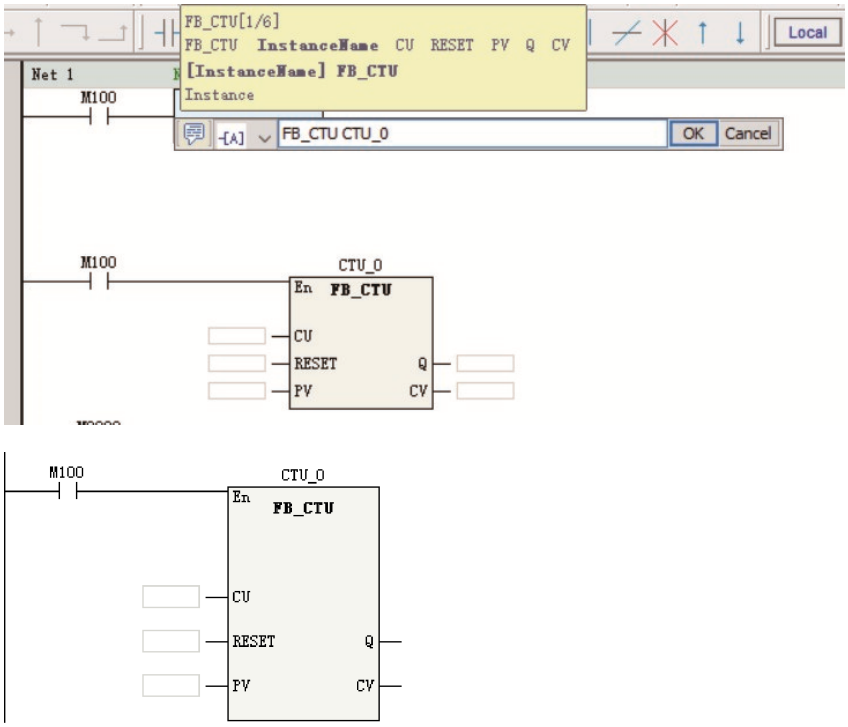
Calling the Function Block in an LD Application

- Method 1: Directly enter the function block name in the LD application, enter the instance name in the ??? field at the top of the function block instruction, and then click OK to instantiate the function block.



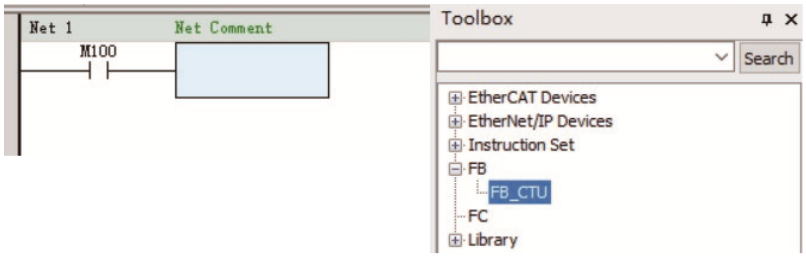


- Method 2: Directly enter the function block name and instance name in the LD application and click OK to instantiate the function block.



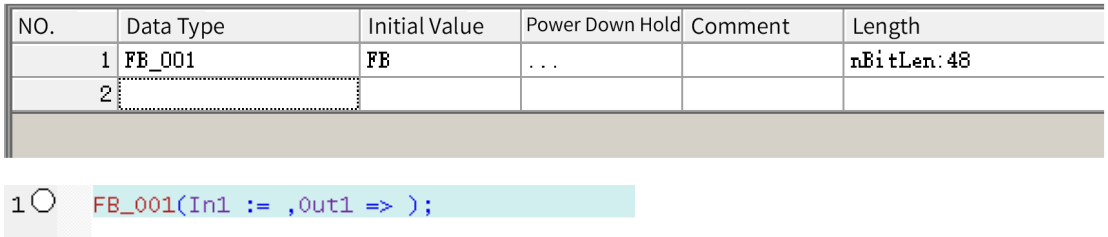
After instantiation is completed, edit the instruction parameters in the function block instruction as required by the program to call the instantiated function block.

- Method 3: Double-click the function block instruction under Function Block (FB) of the Toolbox pane to add the function block instruction to the selected position in the LD application. Then, enter the instance name in the graphic block instruction to define the instance.



Calling the Function Block in a LiteST Application

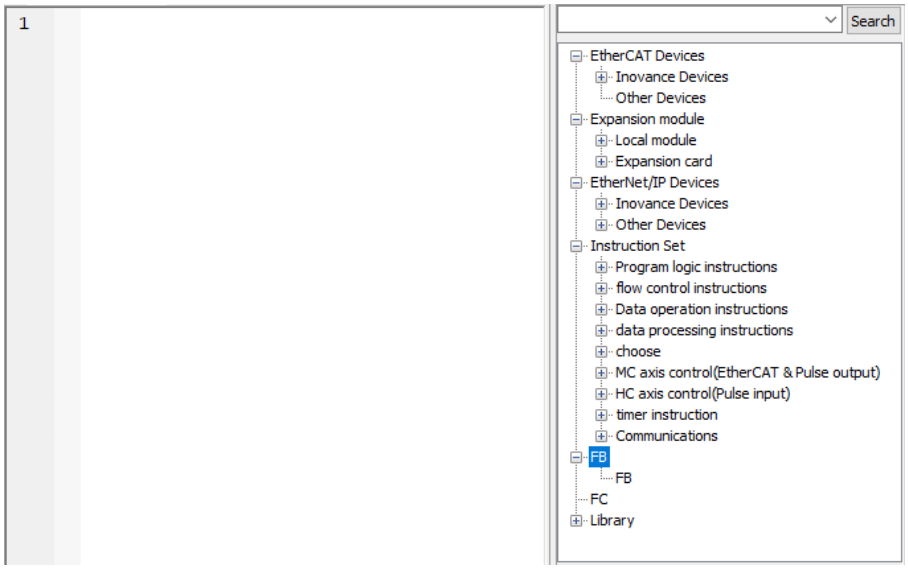
- Method 1: Declare the function block instance in the function block instance list, directly enter the function block instance name in the LiteST application, and then press Tab.

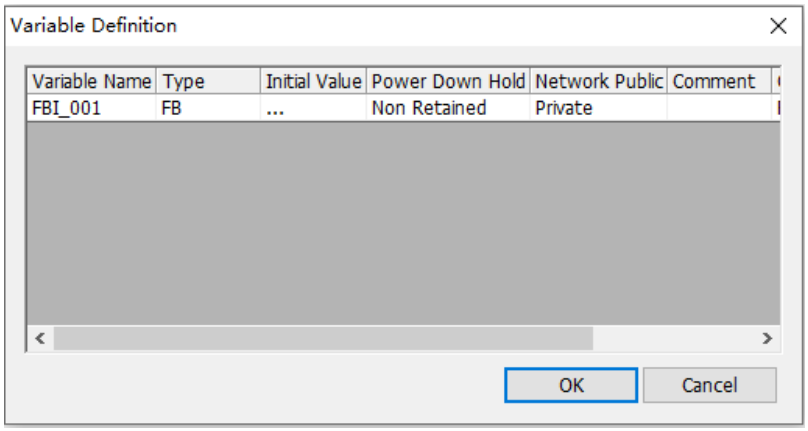


NO.	Data Type	Initial Value	Power Down Hold	Comment	Length
1	FB_001	FB	...		nBitLen:48
2					

```
1  FB_001(In1 := ,Out1 => );
```

- Method 2: Double-click the function block instruction under Function Block (FB) of the Toolbox pane. In the dialog box that is displayed, enter the name of the function block instance to be registered and click OK to add the function block instruction to the selected position in the LiteST application.

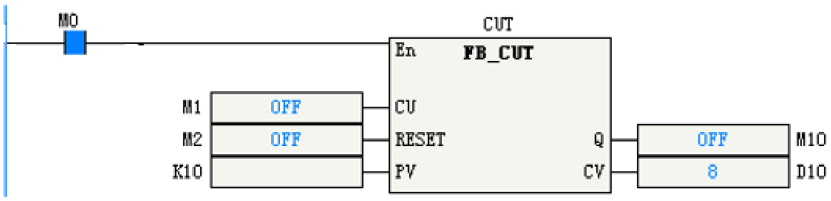




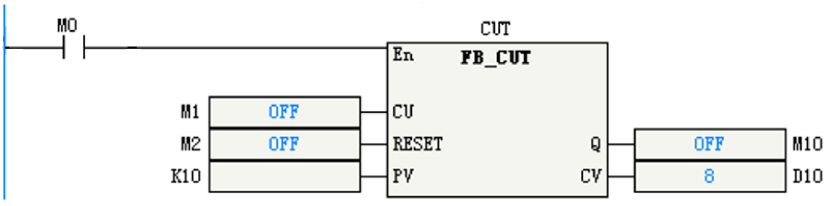
Running the Function Block

Running the Function Block in an LD Application

After the function block is instantiated, the En of the function block is connected to the LD network. When the En network energy flow is ON, the function block program is executed, and the output of the function block changes with the input state and internal variable state. When the En network energy flow is OFF, the function block program is not executed, and output of the function block is not refreshed.



When the energy flow of the counter function block (CUT) is ON, the function block is executed. The output CV increases by 1 when the input condition CU changes on the rising edge.



When the energy flow of the CUT is OFF, the function block is not executed. The output CV is not refreshed when the input condition CU changes on the rising edge.

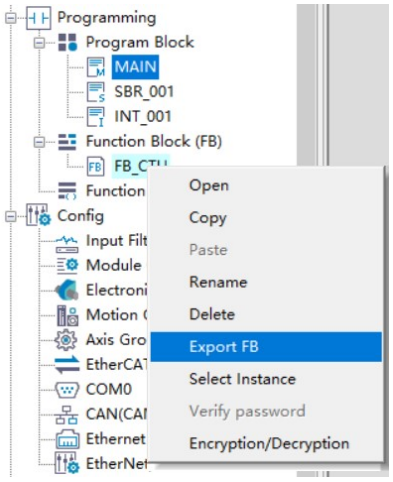
Running the Function Block in a LiteST Application

When the function block in the LiteST application is executed, the content in the function block is executed.

Encapsulating the Function Block

The function block can be encapsulated into a library after being edited and debugged. The function block encapsulated into a library can be multiplexed in different programs through library management of AutoShop.

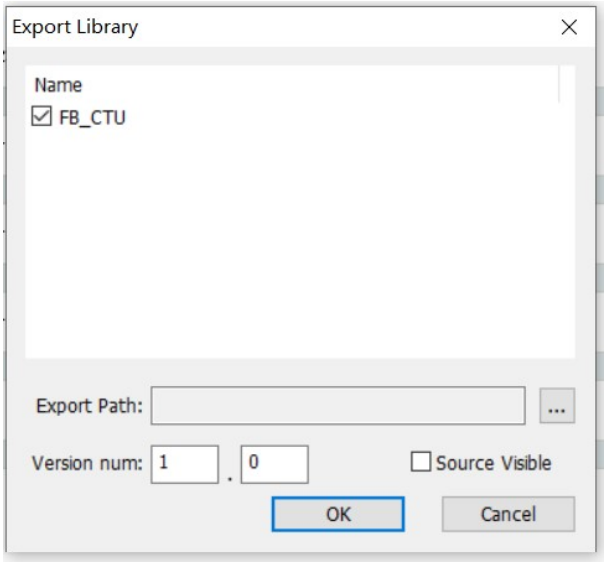
1. Right-click a function block under Function Block (FB) and select Export FB.



2. If you deselect Source Visible, you can only call but cannot view or modify the function block program in the project after the library is imported.

NOTE

- Encrypted function blocks, function blocks that call encrypted function blocks, and function blocks that call encrypted functions cannot be exported.
- If the selected function block contains other function blocks or structures, these function blocks or structures are also exported.
  - If you select Source Visible, you can debug or modify the function block program after it is imported to the project.
  - If you deselect Source Visible, you can only call but cannot view or modify the function block program in the project after the library is imported.In the Export Library dialog box, select the function block to be encapsulated, specify Export Path and Version num, determine whether to select Source Visible, and then click OK. The function block is exported to the specified location, and a function block in .fe format is generated.
  - If you select Source Visible, you can debug or modify the function block program after it is imported to the project.

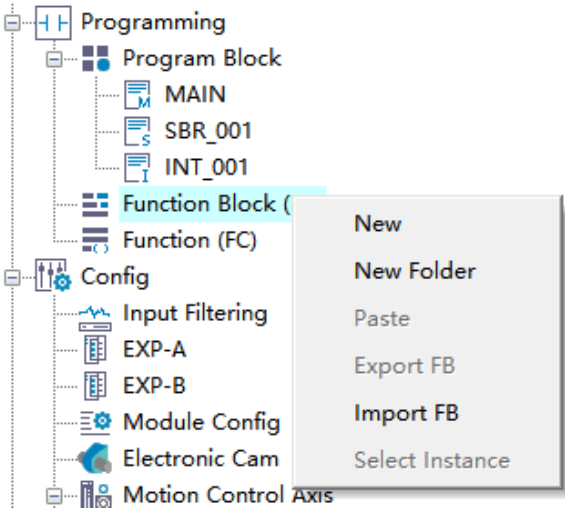


In the Export Library dialog box, select the function block to be encapsulated, specify Export Path and Version num, determine whether to select Source Visible, and then click OK. The function block is exported to the specified location, and a function block in .fe format is generated.

Importing the Function Block

After the function block is exported as a library, it can be called in other programs after being imported. You can use either of the following methods to import the function block:

- Method 1: Right-click Function Block (FB) under Programming in the project management tree and select Import FB to import the library.



This method can only be used to import function blocks of which the source code is visible. You can double-click the imported function block program to edit and debug it. The function block library imported using this method is managed in the project. If you want to call the function block in a new project, you need to re-import the library.

- Method 2: Choose Library > Custom Library on the Toolbox pane and select Import Library to import the library.
  This method can be used to import function blocks regardless of whether the source code is visible. The libraries imported this way are managed as custom libraries, and the function blocks in the libraries can be used directly when you create a new project. You can double-click the function block library imported by using the took kit to directly add it to the LD or LiteST program as an instruction. If you need to view or

modify a function block program of which the source code is visible, you need to import it through project management.

> **NOTE**
>
> If the library imported by using the tool kit contains other function blocks or structures, these function blocks or structures can be selected and used in variable tables and instance tables.

3.5.2 Function

Introduction

A function (FC) is an independently encapsulated program block that defines input and output parameters or non-static internal variables. When a function is called with the same input parameters, the output results are the same.

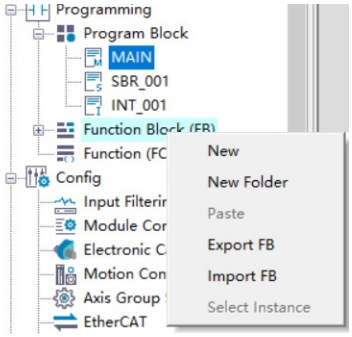
The major difference between a function and a function block (FB) is that internal variables of a function are static without internal state storage, so the same input parameters yield the same output.

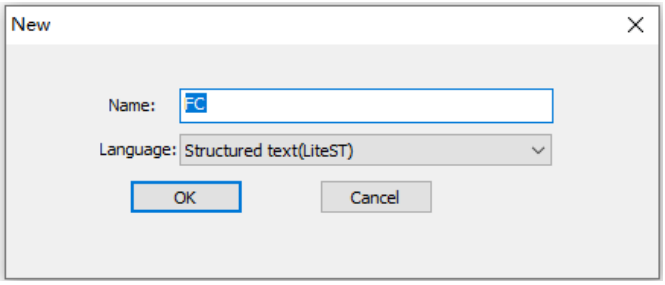
As a basic arithmetic unit, functions are often used in various mathematical operations. For example, sin(x) and sqrt(x) are typical functions.

The basic steps of using a function are as follows: Create a function > Program the function > Call the function > Run the function > Encapsulate the function.

Creating a Function

Expand the Programming node in the project management window, right-click Function (FC), or right-click a folder under Function (FC), and choose New. In the dialog box that is displayed, enter a function name, select a language, and click OK. A function is created successfully.

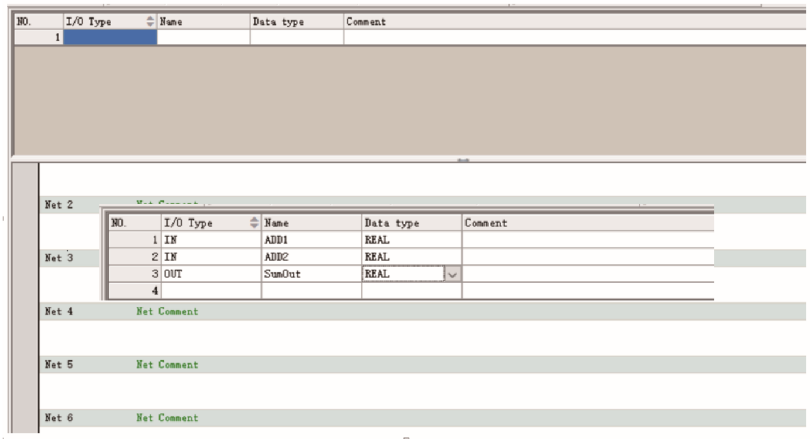




Programming the Function

Double-click the created function under Function (FC) to go to the function program editing interface. The editing interface of the function is similar to that of the function block. Compared with ordinary program

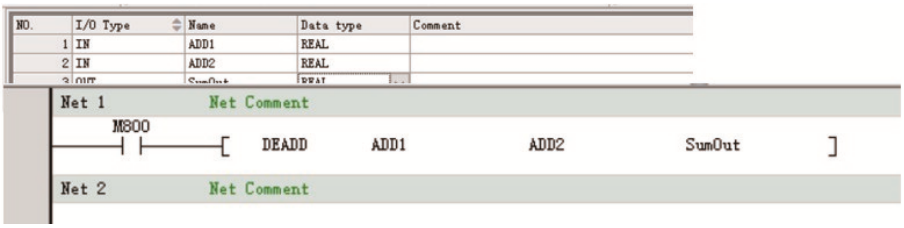
editing, the function program editing interface has two additional windows for defining input/output and local variables.



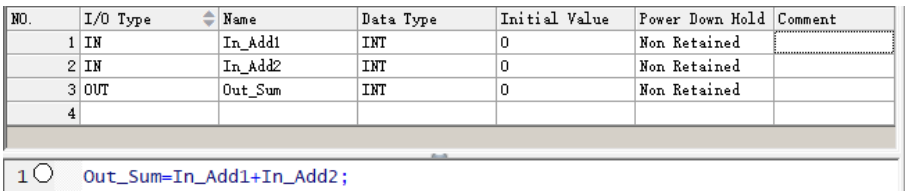
In the windows for defining input/output and local variables, you can define the input (IN), output (OUT), input/output (INOUT), and local variable (VAR) of a function block. The supported data types include BOOL, INT, DINT, REAL, BYTE, IP, and STRING. You can also define array variables and structure variables. To use structure variables, you need to create structure members in the structure of global variables.

- Compared with variables of function blocks, you cannot specify initial values for variables of a function and all its local variables are non-retentive.
- The function program adopts ladder diagram (LD) programming. It can call functions. A function can be called by other functions, function blocks, and programs.
- In addition to variables, the function program can also use M8000 as an always ON variable.
- Instructions related to states or executed for multiple cycles, such as LDP and MC_Power, cannot be used in a function program.

Example 1: Encapsulating an ADD function with a function (LD)



Example 2: Encapsulating an ADD function with a function (LiteST)

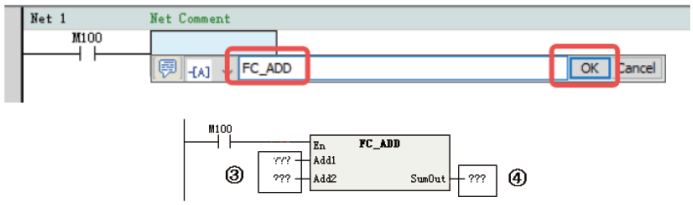


Calling the Function

The function program can be called directly or used in an LD or structure text (LiteST) application after it is compiled.

Calling the Function in an LD Application

- Method 1: Directly enter the function name and press Enter in the LD application. In the graphic block instruction that is displayed, edit the input and output parameters.

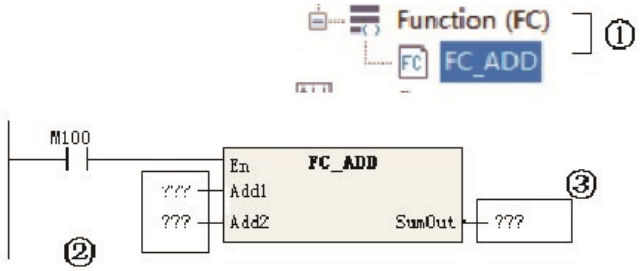


1. Enter the function name.

2. Click OK.

3/4. Add input and output variables.

- Method 2: After a function program is created, the corresponding instruction is generated under Function (FC) on the Toolbox pane. Double-click the FC instruction under Function (FC) to add the FC instruction to the selected position in the LD application.



1. Double-click the FC instruction to add it.
2. Add input parameters.

3. Add output parameters.

Calling the Function in a LiteST Application

- Directly enter the function name in the LiteST application and press Tab.

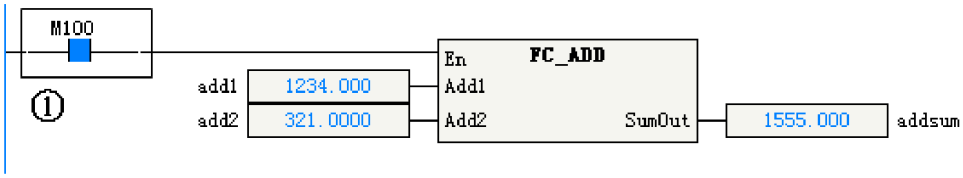
```
1  FC(In_Add1 := ???,In_Add2 := ???,Out_Sum => );
```

- After a function program is created, the corresponding instruction is generated under Function (FC) on the Toolbox pane. Double-click the FC instruction under Function (FC) to add the FC instruction to the selected position in the LiteST application.

Running the Function

Running the Function in an LD Application

After the function is called, the En of the function is connected to the LD network. When the En network energy flow is ON, the function program is executed, and the output of the function changes with the input state. When the En network energy flow is OFF, the function program is not executed, and output of the function is not refreshed.



1. The function is executed when the En network energy flow is ON.

Running the Function in a LiteST Application

When the function in the LiteST application is executed, the content in the function is executed.

Encapsulating the Function

The encapsulation procedure of functions is similar to that of function blocks. For details, see "Encapsulating the Function Block" in the Function Blocks section.

Importing the Function

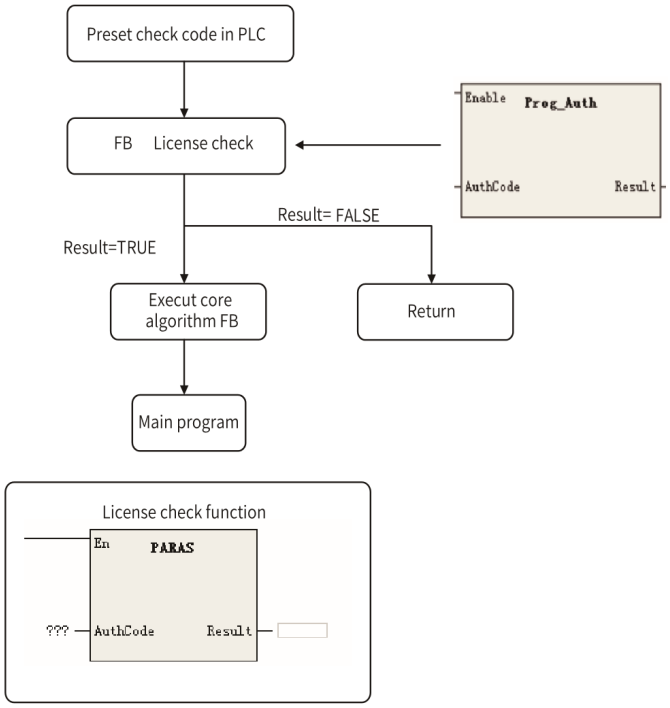
The import procedure of functions is similar to that of function blocks. For details, see "Importing the Function Block" in the Function Blocks section.

3.5.3 Authorizing a Function Block

Introduction

You can use the Prog_Auth or PARAS function to control the core algorithm function blocks and compile them into a library file.

Only authorized PLCs that pass the verification can use this library file, protecting the intellectual property of the equipment manufacturer.



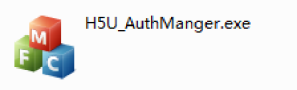
1 - Check code
2 - Results returned (BOOL type); ON: Succeed; OFF: Failed

The differences between the Prog_Auth and PARAS functions are as follows:

- The Prog_Auth function is triggered by contact. After the authorization code passes verification, the Result pin executes the user program in the function block.
- The PARAS function does not need to be triggered by contact. After the authorization code passes verification, the user program in the function block is executed. If verification fails, program execution directly returns.

Setting the Authorization Code by Using a Tool

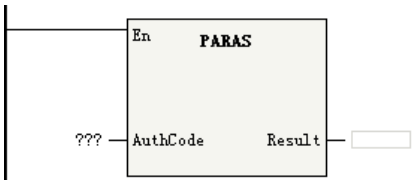
1. Double-click **H5U_AuthManger.exe** in the software installation directory to open it.



2. Select **Ethernet (IP address)** for **Communication Type**.
3. Enter the IP address of the PLC for which you want to set the authorization code in the **Connected PLC IP** field, enter the 8-digit authorization code, and click **Set Authorization Code**.
4. Click **Generate Authorization Code**. The authorization code is generated for instructions.
5. Click **Verify Authorization Code** or **Clear Authorization Code**. (You can perform this operation only after you enter the authorization code.)

Adding a Program Block

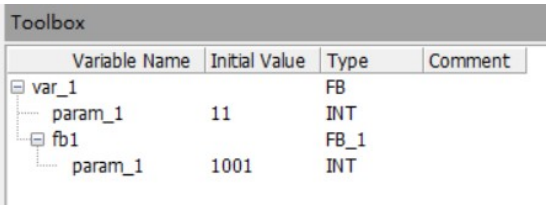
1. Open the function block to be authorized, and add the PARAS function block.



2. Enter the authorization code for instructions generated by the software in the **AuthCode** field.
3. The function block is authorized. If the authorization code of the PLC differs from that in the function block, the program in the function block cannot run.

Example

OFF is returned because the authorization code obtained by using the Prog_Auth function is different from the preset authorization code in the PLC. In this case, the ADD instruction in the following program is not executed.



3.5.4 Setting the Initial Value of Function Block

Introduction

The initial values of function blocks (FBs) can be modified based on the FB type or FB instance.

- If you modify the initial values based on the FB type, modify the initial value of the type.
- If you modify the initial values based on the FB instance, modify the initial value of the instance.
- After you modify the initial values of an instance, modified values are displayed for the member variables of the FB instance and the background color of the cells is yellow.
- If you do not modify the initial values of an instance, the default values are displayed for the member variables of the FB instance and the background color of the cells is white.

The initial values of the FB type are the default values of the instance. After you change the initial values of an instance to the default values, the background color of the cells changes from yellow to white.

Modifying Initial Values When the FB Is Not Nested

If you change the initial value of the FB type from 0 to 10, the default value of the FB instance 10 is displayed, which is the same as the initial value of the FB type.

NO.	I/O Type	Name	Data Type	Initial Value	Power Down Hold	Comment
1	VAR	param_1	INT	10	Non Retained	
2						



NO.	Variable Name	Data Type	Initial V...	Comment	Length
1	var_1	FB	...		nBitLen:16
2					

Toolbox

Variable Name	Initial Value	Type	Comment
var_1		FB	
param_1	10	INT	

If you change the initial value of the FB instance from 10 to 100, the initial value of the FB block instance 100 is displayed. If you change the initial value of the FB type to 11, the displayed value is still 100.

Toolbox

Variable Name	Initial Value	Type	Comment
var_1		FB	
param_1	100	INT	

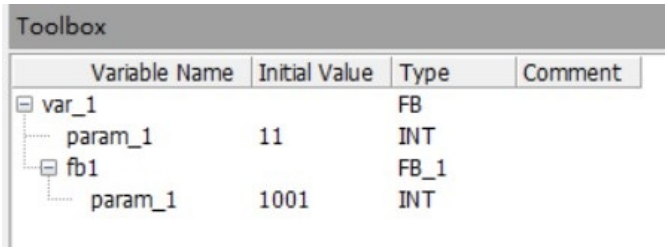
The following figures show an example. In the ladder diagram, double-click **FB** to display the FB instance. In the FB view that is displayed, the initial value of the FB instance is displayed, rather than the initial value of the FB type. If the initial value of the variable is changed to be different from that of the FB, the background color changes to yellow. Initial value modification by using this method achieves the same effect as modifying the initial value of the instance in the FB instance table.

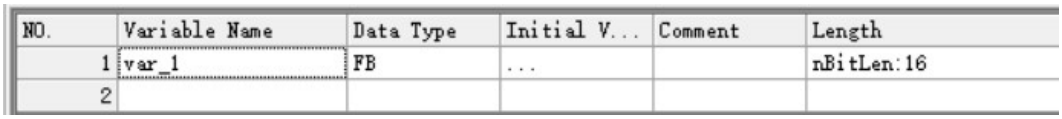
NO.	I/O Type	Name	Data Type	Initial V...	Power Dow...	Comment
1	VAR	param_1	INT	100	Non Retained	
2						

Modifying Initial Values When the FB Is Nested

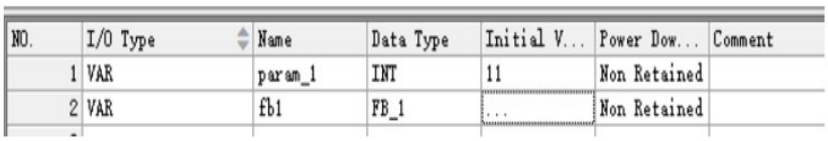
Add a variable fb1 in the FB type, and set the data type to FB_1. Change the initial value of FB_1 from 1000 to 1001. The member variable fb1 of the FB type automatically takes the default value 1001 as the initial value, and the member variable fb1 of the FB instance also automatically takes the default value 1001 as the initial value.

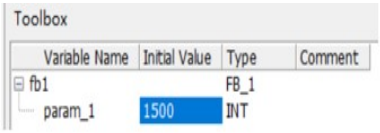
NO.	I/O Type	Name	Data Type	Initial V...	Power Dow...	Comment
1	VAR	param_1	INT	11	Non Retained	
2	VAR	fb1	FB_1	...	Non Retained	
3						



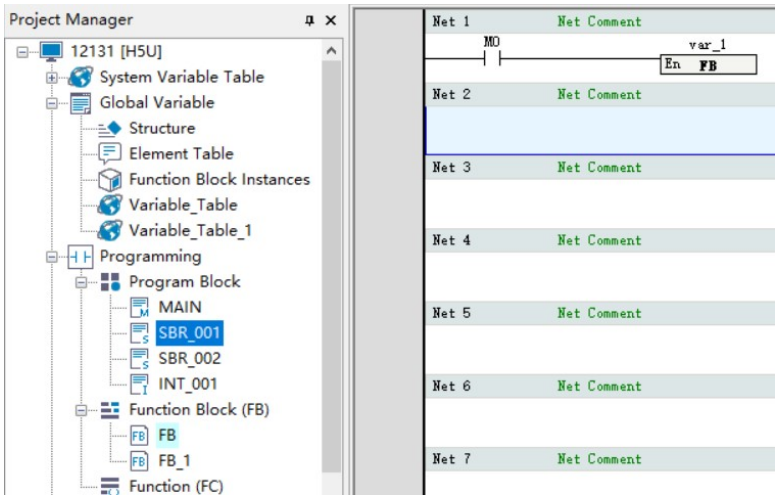


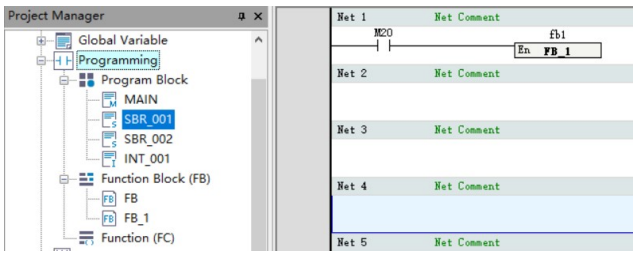
FB is the middle layer between the instance and FB_1. Modify the initial value of FB_1 to 1500 on the FB type interface. Then the initial value of the FB type changes to 1500, and the background color changes to yellow. At this time, the initial value of FB_1 of the FB instance is also 1500, but the background color is white, indicating that the default value is used.

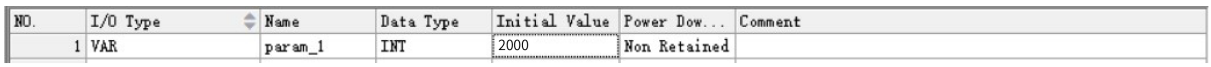




- Enter the instance interface from the main program. The initial value of the FB instance is displayed. Double-click **FB_1** to enter the FB_1 instance interface, and change the initial value to 2000. After you double-click the FB_1 type, you can find that the initial value is still 1001. After you double-click the FB instance FB_1, you can find that the initial value is 2000.







At this time, the tab name is **FB_1(var_1.fb1)**.

- Double-click **FB** in the Project Manager navigation tree. You can find that the initial value of FB_1 on the FB type interface is 1500. Double-click **FB_1** in the ladder diagram of the FB type interface to enter the FB_1 instance interface. You can find that the initial value of the FB_1 instance is 1500. Change it to 2500 and return to the FB type interface. You can find that the initial value of FB_1 also changes to 2500.

NO.	I/O Type	Name	Data Type	Initial Value	Power Dow...	Comment
1	VAR	param_1	INT	2500	Non Retained	

At this time, the tab name is **FB_1(fb1)**.

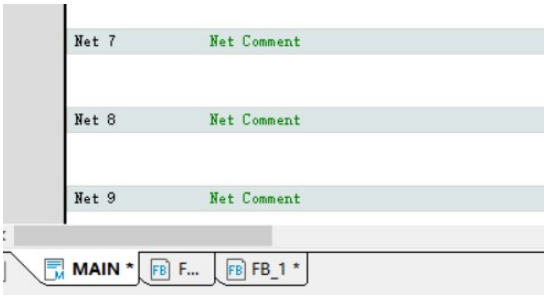
NO.	I/O Type	Name	Data Type	Initial V...	Power Dow...	Comment
1	VAR	param_1	INT	11	Non Retained	
2	VAR	fb1	FB_1	...	Non Retained	

Toolbox

Variable Name	Initial Value	Type	Comment
fb1		FB_1	
param_1	2500	INT	

Tab at the Bottom of the FB View

The tab displayed at the bottom of the FB view contains the following information from left to right: node name, instance name, and flag indicating whether the data is saved. The node name is the name of the project tree node, and the instance name is enclosed in a pair of parentheses.



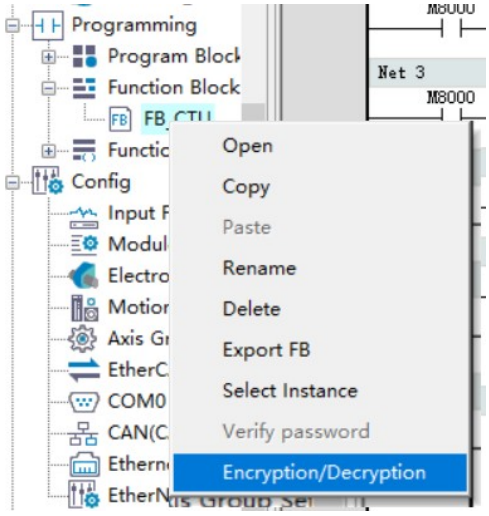
As shown in the preceding figure, the node name is **FB**, the instance name is **var_1.fb1**, and the asterisk (*) indicates that the data is not saved.

Since the tab needs to be parsed, characters including the period (.), asterisk (*), and parentheses (()) are not allowed when FBs and structures are renamed.

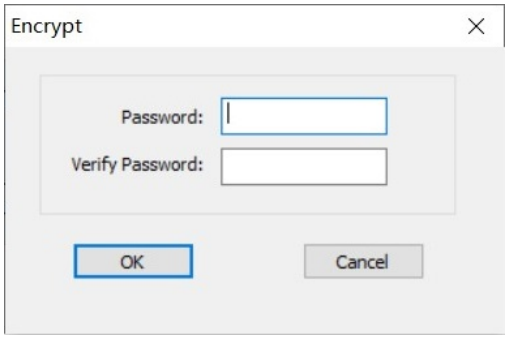
3.5.5 Encrypting a Function Block or Function

This section takes encryption of function blocks as an example. The process is similar for encrypting functions. After encryption, the method of calling the function blocks or functions remains unchanged.

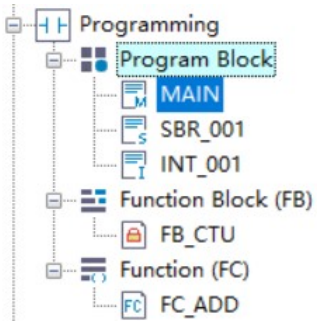
1. Choose "Programming" > "Function Block (FB)" in the project management window, right-click "FB", and choose "Encryption/Decryption".



2. Enter and confirm the password in the displayed "Encrypt" dialog box.



The following figure shows a function block after encryption.



Performing the preceding steps on an encrypted function block will decrypt it and restore it to its original unencrypted state.

To access an encrypted function block, you can double-click the encrypted node, or right-click the encrypted node and choose "Password verification" from the shortcut menu, and enter the correct password in the displayed dialog box.



4 Instructions Cheat Sheet

The following table lists all the instructions supported by the PLC that are classified by function.

Category	Name	Function Description	Language Support
Contact instructions	LD	Load NO contact	LD
	LDI	Load NC contact	LD
	AND	Serial connection of NO contacts	LD
	ANI	Serial connection of NC contacts	LD
	OR	Parallel connection of NO contacts	LD
	ORI	Parallel connection of NC contacts	LD
	LDP	Obtain pulse rising edge	LD
	LDF	Obtain pulse falling edge	LD
	ANDP	Serial connection of pulse rising edge	LD
	ANDF	Serial connection of pulse falling edge	LD
	ORP	Parallel connection of pulse rising edge	LD
	ORF	Parallel connection of pulse falling edge	LD
	MEP	Conversion of operation result to rising edge pulse	LD
	MEF	Conversion of operation result to falling edge pulse	LD
Output control instructions	OUT	Coil drive	LD
	SET	SET action storage coil	LD
	RST	Contact or cache clearing	LD
	ZSET	Set all data	LD and LiteST
	ZRST	Reset all data	LD and LiteST
	PLS	Pulse rising edge detection coil	LD
	PLF	Pulse falling edge detection coil	LD
	ALT	Alternate output	LD
	R_TRIG	Rising edge detection	LD and LiteST

Category	Name	Function Description	Language Support
	F_TRIG	Falling edge detection	LD and LiteST
Flow control instructions	INV	Operation result inversion	LD
Immediate input (contact) instructions	ILD	Load NO immediate contact	LD and LiteST
	ILDI	Load NC immediate contact	LD and LiteST
	IAND	Serial NO immediate contact	LD and LiteST
	IANI	Serial NC immediate contact	LD and LiteST
	IOR	Parallel NO immediate contact	LD and LiteST
	IORI	Parallel NC immediate contact	LD and LiteST
Immediate output instructions	IOUT	Immediately output single local I/O	LD and LiteST
	ISET	Immediately output single local I/O	LD and LiteST
	IRST	Immediately reset single local I/O	LD and LiteST
	IZSET	Immediately set local I/Os in specified range in batches	LD and LiteST
	IZRST	Immediately reset local I/Os in specified range in batches	LD and LiteST
Immediate I/O instructions	IRDBIT	Immediately read bits	LiteST
	IWRBIT	Immediately write bits	LiteST
Process control instructions	CJ	Conditional jump	LD
	LBL	Label	LD
	CALL	Call subprogram	LD
	SSRET	Conditional subprogram return	LD
	EI	Enable interrupt	LD and LiteST
	DI	Disable interrupt	LD and LiteST
	WDT	Watchdog timer reset	LD
	FOR	Start of a loop	LD
	NEXT	End of a loop	LD
SFC instructions	STL	Program jump to secondary bus	LD

Category	Name	Function Description	Language Support
	RET	Program return to primary bus	LD
	OUTSTL	Output program jump to secondary bus	LD
	SETSTL	Setting program jump to secondary bus	LD
	RSTSTL	Resetting program jump to secondary bus	LD
Contact comparison instructions	AND=	AND contact comparison equal to	LD
	AND>	AND contact comparison greater than	LD
	AND<	AND contact comparison less than	LD
	AND<>	AND contact comparison not equal to	LD
	AND>=	AND contact comparison greater than or equal to	LD
	AND<=	AND contact comparison less than or equal to	LD
	LD=	LD contact comparison equal to	LD
	LD>	LD contact comparison greater than	LD
	LD<	LD contact comparison less than	LD
	LD<>	LD contact comparison not equal to	LD
	LD>=	LD contact comparison greater than or equal to	LD
	LD<=	LD contact comparison less than or equal to	LD
	OR=	OR contact comparison equal to	LD
	OR>	OR contact comparison greater than	LD
	OR<	OR contact comparison less than	LD
	OR<>	OR contact comparison not equal to	LD
	OR>=	OR contact comparison greater than or equal to	LD
	OR<=	OR contact comparison less than or equal to	LD
	FLDD>		LD

Category	Name	Function Description	Language Support
		Conductive when floating-point number greater than comparative state contact (S1 > S2)	
	FLDD>=	State contact of floating-point comparison >=, conductive when S1 ≥ S2	LD
	FLDD<	Conductive when floating-point number less than comparative state contact (S1 < S2)	LD
	FLDD<=	State contact of floating-point comparison <=, conductive when S1 ≤ S2	LD
	FLDD=	Conductive when floating-point number equal to comparative state contact (S1 = S2)	LD
	FLDD<>	Conductive when floating-point number not equal to comparative state contact (S1 ≠ S2)	LD
	FANDD>	Conductive when floating-point number greater than comparative AND state contact (S1 > S2)	LD
	FANDD>=	AND state contact of floating-point comparison >=, conductive when S1 ≥ S2	LD
	FANDD<	Conductive when floating-point number less than comparative AND state contact (S1 < S2)	LD
	FANDD<=	AND state contact of floating-point comparison <=, conductive when S1 ≤ S2	LD
	FANDD=	Conductive when floating-point number equal to comparative AND state contact (S1 = S2)	LD
	FANDD<>	Conductive when floating-point number not equal to comparative AND state contact (S1 ≠ S2)	LD
	FORD>	Conductive when floating-point number greater than comparative OR state contact (S1 > S2)	LD

Category	Name	Function Description	Language Support
	FORD>=	OR state contact of floating-point comparison >=, conductive when S1 ≥ S2	LD
	FORD<	Conductive when floating-point number less than comparative OR state contact (S1 < S2)	LD
	FORD<=	OR state contact of floating-point comparison <=, conductive when S1 ≤ S2	LD
	FORD=	Conductive when floating-point number equal to comparative OR state contact (S1 = S2)	LD
	FORD<>	Conductive when floating-point number not equal to comparative OR state contact (S1 ≠ S2)	LD
	LDZ>	Conducting when absolute value > comparative state contact |S1 - S2| > |S3|	LD
	LDZ>=	Conducting when absolute value >= comparative state contact |S1 - S2| ≧ |S3|	LD
	LDZ<	Conducting when absolute value < comparative state contact |S1 - S2| < |S3|	LD
	LDZ<=	Conducting when absolute value <= comparative state contact |S1 - S2| ≦ |S3|	LD
	LDZ=	Conducting when absolute value = comparative state contact |S1 - S2| = |S3|	LD
	LDZ<>	Conductive when absolute value not equal to comparative state contact (|S1 - S2| ≠ |S3|)	LD
	ANDZ>	Conducting when absolute value > comparative AND state contact |S1 - S2| > |S3|	LD
	ANDZ>=	Conducting when absolute value >= comparative AND state contact |S1 - S2| ≧ |S3|	LD
	ANDZ<	Conducting when absolute value < comparative AND state contact |S1 - S2| < |S3|	LD
	ANDZ<=		LD

Category	Name	Function Description	Language Support
		Conducting when absolute value <= comparative AND state contact |S1 - S2| ≦ |S3|	
	ANDZ=	Conducting when absolute value = comparative AND state contact |S1 - S2| = |S3|	LD
	ANDZ<>	Conductive when absolute value not equal to comparative AND state contact (|S1 - S2| ≠ |S3|)	LD
	ORZ>	Conducting when absolute value > comparative OR state contact |S1 - S2| > |S3|	LD
	ORZ>=	Conducting when absolute value >= comparative OR state contact |S1 - S2| ≧ |S3|	LD
	ORZ<	Conducting when absolute value < comparative OR state contact |S1 - S2| < |S3|	LD
	ORZ<=	Conducting when absolute value <= comparative OR state contact |S1 - S2| ≦ |S3|	LD
	ORZ=	Conducting when absolute value = comparative OR state contact |S1 - S2| = |S3|	LD
	ORZ<>	Conductive when absolute value not equal to comparative OR state contact (|S1 - S2| ≠ |S3|)	LD
Contact logic operation instructions	LD&	LD logical AND operation	LD
	LD|	LD logical OR operation	LD
	LD^	LD logical XOR operation	LD
	AND&	AND logical AND operation	LD
	AND|	AND logical OR operation	LD
	AND^	AND logical XOR operation	LD
	OR&	OR logical AND operation	LD
	OR|	OR logical OR operation	LD
	OR^	OR logical XOR operation	LD
Arithmetic operation instructions	ADD	Binary data addition	LD

Category	Name	Function Description	Language Support
	SUB	Binary data subtraction	LD
	MUL	Binary data multiplication	LD
	DIV	Binary data division	LD
	ABS	Absolute value	LD and LiteST
	DEABS	Absolute value of floating-point number	LD
	MOD	Remainder by binary data division	LD and LiteST
	EADD	Binary floating-point addition	LD
	ESUB	Binary floating-point subtraction	LD
	EMUL	Binary floating-point multiplication	LD
	EDIV	Binary floating-point division	LD
	INC	Binary data increment by 1	LD
	DEC	Binary data decrement by 1	LD
Data logic operation instructions	WAND	Binary data logic AND	LD
	WOR	Binary data logic OR	LD
	WXOR	Binary data logic XOR	LD
	NEG	Binary data negation	LD
	ENEG	Binary floating-point sign negation	LD
Word bit operation instructions	BLD	Word or dword bit contact	LD
	BLDI	Word or dword bit inversion contact	LD
	BAND	Word or dword bit AND contact	LD
	BANDI	Word or dword bit AND inversion contact	LD
	BOR	Word or dword bit OR contact	LD
	BORI	Word or dword bit OR inversion contact	LD
	BOUT	Word or dword bit data output	LD
	BSET	Word or dword bit data setting	LD
	BRST	Word or dword bit data reset	LD

Category	Name	Function Description	Language Support
Trigonometric function instructions	SIN	Floating-point SIN operation	LD and LiteST
	COS	Floating-point COS operation	LD and LiteST
	TAN	Floating-point TAN operation	LD and LiteST
	ASIN	Binary floating-point ARCSIN operation	LD and LiteST
	ACOS	Binary floating-point ARCCOS operation	LD and LiteST
	ATAN	Binary floating-point ARCTAN operation	LD and LiteST
	RAD	Binary floating-point degree-to-radian conversion	LD
	DEG	Binary floating-point radian-to-degree conversion	LD
	SINH	Binary floating-point SINH operation	LD
	COSH	Binary floating-point COSH operation	LD
	TANH	Binary floating-point TANH operation	LD
Table operation instructions	WSUM	Data sum calculation	LD
	MEAN	Mean calculation	LD
	LIMIT	Upper and lower limit control	LD
	BZAND	Dead zone control	LD
	ZONE	Zone control	LD
	SCL	Coordinate determination (coordinate data of different points)	LD
	SCL2	Coordinate determination 2 (X and Y coordinates)	LD
Exponent operation instructions	EXP	Binary floating-point exponentiation operation	LD
	LOGE	Binary floating-point natural logarithm operation	LD
	LOG	Binary floating-point logarithm operation with a base of 10	LD and LiteST
	ESQR	Binary floating-point square root operation	LD

Category	Name	Function Description	Language Support
	SQR	Binary data square root operation	LD
	POW	Floating-point weight	LD
	LN	Base e (2.71828) logarithm	LiteST
	EXPT	Power operation	LiteST
	SQRT	Square root operation	LiteST
Bit operator instructions	AND	AND operation	LiteST
	OR	OR operation	LiteST
Data conversion instructions	INT	Conversion from binary floating-point number into BIN integer	LD
	BCD	Conversion from binary into BCD	LD
	BIN	Conversion from BCD into binary	LD
	FLT	Conversion from binary into binary floating-point	LD
	EBCD	Conversion from binary floating-point into decimal floating-point	LD
	EBIN	Conversion from decimal floating-point into binary floating-point	LD
	DABIN	Conversion from decimal ASCII into binary	LD
	BINDA	Conversion from binary into decimal ASCII	LD
	WTOB	Conversion from word to byte	LD
	BITW	Conversion from bit to word	LD and LiteST
	BTOW	Conversion from byte to word	LD
	WBIT	Conversion from word to bit	LD and LiteST
	WTODW	Conversion from word to dword	LD
	DWTOW	Conversion from dword to word	LD
	MCPY	Data copy (memory copy and type conversion)	LD and LiteST
	MSET	Data setting (memory setting and reset)	LD and LiteST
	UNI	4-bit combination of 16-bit data	LD

Category	Name	Function Description	Language Support
	DIS	4-bit separation of 16-bit data	LD
	ASCI	Conversion from hexadecimal into ASCII	LD
	HEX	Conversion from ASCII into hexadecimal	LD
	DECO	Data decoding	LD
	ENCO	Data encoding	LD
	SCALE_X	Scaling	LD
	NORM_X	Normalization	LD
	ROUND	Round up	LD
	TRUNC	Truncation	LD
	CELL	Floating-point number celling	LD
	FLOOR	Floating-point number flooring	LD
	AryByteTo	Combination of byte array members and saving to variables	LiteST
	AryToString	Conversion from byte array to string	LiteST
	BOOL_TO_<TYPE>	Conversion from BOOL to <TYPE>	LiteST
	DINT_TO_<TYPE>	Conversion from DINT to <TYPE>	LiteST
	INT_TO_<TYPE>	Conversion from INT to <TYPE>	LiteST
	REAL_TO_<TYPE>	Conversion from REAL to <TYPE>	LiteST
	BYTE_TO_<TYPE>	Conversion from byte to <TYPE>	LiteST
	STRING_TO_<TYPE>	Conversion from string to <TYPE>	LiteST
	StringToAry	Conversion from string to byte array	LiteST
	TO_<TYPE>	Conversion from variable to <TYPE>	LiteST
	ToAryByte	Variable separation by 1 byte and saving to byte array	LiteST
Data transfer and comparison instructions	MOV	Move	LD
	EMOV	Binary floating-point move	LD

Category	Name	Function Description	Language Support
	SMOV	Shift move	LD
	BMOV	Batch move	LD
	FMOV	Multi-point move	LD
	CML	Complement	LD
	CMP	Comparison	LD
	ECMP	Floating-point comparison	LD
	ZCP	Zone comparison	LD
	EZCP	Floating-point zone comparison	LD
Table management instructions	SORTR	Data sorting	LD
	SORTC	Data sorting 2	LD
	SER	Data search	LD
	FDEL	Deletion of data from data table	LD
	FINS	Insertion of data to data table	LD
	POP	Last-in data read	LD
	RAMP	Ramp	LD
Data shift instructions	ROR	Rotation right	LD
	ROL	Rotation left	LD
	RCR	Rotation right with carry	LD
	RCL	Rotation left with carry	LD
	SFTR	Bit shift right	LD
	SFTL	Bit shift left	LD
	WSFR	Word shift right	LD
	WSFL	Word shift left	LD
	SFWR	Data write (FIFO)	LD
	SFRD	Data read (FIFO)	LD
	SFR	16-bit data shift right with carry by n bits	LD

Category	Name	Function Description	Language Support
	SFL	16-bit data shift left with carry by n bits	LD
	SHL	Shift left operation	LiteST
	SHR	Shift right operation	LiteST
Other data processing instructions	SWAP	Byte swap	LD
	BON	Bit state check	LD
	SUM	Sum of ON bits	LD
	RAND	Random number generation within limits	LD
	XCH	Data exchange	LD
	EFMOV	Multi-point floating-point move	LD
	CCD	Check code	LD
	CRC	CRC code calculation	LD
	LRC	LRC code calculation	LD
Matrix operation instructions	BK+	Block data addition	LD
	BK-	Block data subtraction	LD
	MAND	Matrix AND	LD
	MOR	Matrix OR	LD
	MXOR	Matrix XOR	LD
	MXNR	Matrix XNOR	LD
	MINV	Matrix inversion	LD
Matrix comparison instructions	BKCMP=	Matrix comparison equal-to (S1 = S2)	LD
	BKCMP>	Matrix comparison greater-than (S1 > S2)	LD
	BKCMP<	Matrix comparison less-than (S1 < S2)	LD
	BKCMP<>	Matrix comparison not equal to (S1 ≠ S2)	LD
	BKCMP<=	Matrix comparison less than or equal to (S1 ≤ S2)	LD
	BKCMP>=	Matrix comparison greater than or equal to (S1 ≥ S2)	LD

Category	Name	Function Description	Language Support
String instructions	STR	Conversion from integer into string	LD
	STRMOV	String assignment	LD
	VAL	Conversion from string into integer	LD
	ESTR	Conversion from binary floating-point into string	LD
	EVAL	Conversion from string into binary floating-point	LD
	$ADD	String combination	LD
	LEN	Detect string length	LD and LiteST
	INSTR	String search	LD
	RIGHT	Retrieve from string right	LD and LiteST
	LEFT	Retrieve from string left	LD and LiteST
	MIDR	Retrieve from any position of string	LD
	MIDW	String replacement at any position	LD
	$MOV	String transfer	LD
	ClearString	Clear string	LiteST
	CONCAT	Concatenate and output two strings	LiteST
	DELETE	Delete specific-length string at specified position and output processed string	LiteST
	FIND	Search input source string for position of target string	LiteST
	INSERT	Insert string	LiteST
	REPLACE	Replace characters in string	LiteST
Array instructions	AryMax	Array maximum	LiteST
	AryMin	Array minimum	LiteST
	ArySearch	Search value in one-dimensional array	LiteST
	SizeOfAry	Get number of array elements	LiteST
	AryMove	Array move	LiteST

Category	Name	Function Description	Language Support
	AryExchange	Array data exchange	LiteST
	Clear	Initialize variable	LiteST
	AryAddV	Array value addition	LiteST
	ArySubV	Array value subtraction	LiteST
	ArySHL	Array N-element left shift	LiteST
	ArySHR	Array N-element right shift	LiteST
	AryAnd	Array logic AND	LiteST
	AryOr	Array logic OR	LiteST
	AryXor	Array logic XOR	LiteST
	AryXorN	Array logic XNOR	LiteST
	AryMean	Mean of array elements	LiteST
	ArySD	Standard deviation of array elements	LiteST
Comparison instructions	MAX	Max operation	LiteST
	MIN	Min operation	LiteST
	SEL	Select operation	LiteST
Clock instructions	TCMP	Clock data comparison	LD
	TZCP	Clock data zone comparison	LD
	TADD	Clock data addition	LD
	TSUB	Clock data subtraction	LD
	HTOS	Conversion from hour-minute-second into second	LD
	STOH	Conversion from second into hour-minute-second	LD
	TRD	Read clock data	LD
	TWR	Write clock data	LD
	HOUR	Hour meter	LD
	TZSET	Set time zone	LD

Category	Name	Function Description	Language Support
	TZGET	Read time zone	LD
EtherCAT/Local high-speed pulse output motion control axis instructions	MC_Power	Axis enable	LD and LiteST
	MC_Reset	Fault reset	LD and LiteST
	MC_ReadStatus	Read the Axis state	LD and LiteST
	MC_ReadAxisError	Axis error read	LD and LiteST
	MC_ReadDigitalInput	Digital input read	LD and LiteST
	MC_ReadActualPosition	Current position read	LD and LiteST
	MC_ReadActualVelocity	Current velocity read	LD and LiteST
	MC_ReadActualTorque	Current torque read	LD and LiteST
	MC_SetPosition	Current position setting	LD and LiteST
	MC_TouchProbe	Probe instructions	LD and LiteST
	MC_MoveRelative	Relative positioning	LD and LiteST
	MC_MoveAbsolute	Absolute positioning	LD and LiteST
	MC_MoveVelocity	Velocity control	LD and LiteST
	MC_Jog	Stop motion and motion can be interrupted	LD and LiteST
	MC_TorqueControl	Torque control	LD and LiteST
	MC_Home	Homing	LD and LiteST
	MC_Stop	Axis stop	LD and LiteST
	MC_Halt	Axis halt	LD and LiteST
	MC_MoveFeed	Interrupt positioning	LD and LiteST
	MC_MoveBuffer	Multi-position positioning	LD and LiteST

Category	Name	Function Description	Language Support
	MC_Immediate Stop	Immediate stop	LD and LiteST
	MC_MoveSuper Imposed	Motion superimposition	LD and LiteST
	MC_MoveVeloci tyCSV	CSV-based velocity control with adjustable pulse width	LD and LiteST
	MC_SyncMoveV elocity	CSV-based synchronous velocity control with adjustable pulse width	LD and LiteST
	MC_FollowVelo city	CSP-based synchronous velocity control	LD and LiteST
	MC_SyncTorque Control	Synchronous torque control	LD and LiteST
	MC_SetAxisCon figPara	Axis parameter configuration	LD and LiteST
	MC_SetOverride	Target velocity adjusting during motion	LD and LiteST
	MC_ResetFollo wingError	Following error reset	LD and LiteST
Electronic cam instructions	MC_CamIn	Start cam operation	LD and LiteST
	MC_CamOut	End cam operation	LD and LiteST
	MC_GetCamTab lePhase	Obtain cam table phase	LD and LiteST
	MC_GetCamTab leDistance	Obtain cam table displacement	LD and LiteST
	MC_DigitalCam Switch	Electronic cam tappet control	LD and LiteST
	MC_GearIn	Start gear operation	LD and LiteST
	MC_GearOut	Cancel gear operation	LD and LiteST
	MC_Phasing	Master axis phase shifting	LD and LiteST
	MC_SaveCamTa ble	Save cam table	LD and LiteST
	MC_GenerateCa mTable	Update cam table	LD and LiteST

Category	Name	Function Description	Language Support
	MC_GearInPos	Start the gear operation at the specified position	LD and LiteST
Axis group control instruction	MC_MoveLinear	Linear interpolation	LD and LiteST
	MC_MoveCircular	Circular interpolation	LD and LiteST
	MC_MoveEllipse	Elliptical Interpolation	LD and LiteST
	MC_GroupStop	Stop axis group operation	LD and LiteST
	MC_GroupPause	Pause axis group operation	LD and LiteST
CANopen motion control axis instructions	MC_Power_CO	Enable servo axis through communication	LD
	MC_Reset_CO	Reset servo axis fault through communication	LD
	MC_ReadActualPosition_CO	Read current position of axis through communication	LD
	MC_ReadActualVelocity_CO	Read current velocity of axis through communication	LD
	MC_Halt_CO	Halt servo axis through communication (can be aborted)	LD
	MC_Stop_CO	Stop servo axis through communication (cannot be aborted)	LD
	MC_MoveAbsolute_CO	Control absolute positioning of axis through communication	LD
	MC_MoveRelative_CO	Control relative positioning of axis through communication	LD
	MC_MoveVelocity_CO	Control axis velocity operation mode through communication	LD
	MC_Jog_CO	Control axis jogging through communication	LD
	MC_Home_CO	Control axis homing through communication	LD
	MC_WriteParameter_CO	Write axis parameters through communication	LD

Category	Name	Function Description	Language Support
	MC_ReadParameter_CO	Read axis parameters through communication	LD
EtherCAT/Local high-speed pulse output motion control axis instructions	MC_FollowVelocity	CSP-based synchronous velocity control	LD and LiteST
	MC_MoveVelocityPV	Profile velocity control	LD
	MC_MoveAbsRelPP	Positioning based on the profile position mode	LD
	MC_JogP	Jogging based on the profile mode	LD
	MC_MoveBufferPP	Multi-segment positioning in profile position mode	LD
	MC_TorqueControlPT	Torque control in profile torque mode	LD
Bus encoder axis instructions (H5U)	ENC_Counter	Encoder enable	LD and LiteST
	ENC_Reset	Reset encoder	LD and LiteST
	ENC_Preset	Encoder preset	LD and LiteST
	ENC_TouchProbe	Encoder probe	LD and LiteST
	ENC_ArrayCompare	Encoder one-dimensional array comparison	LD and LiteST
	ENC_StepCompare	Encoder one-dimensional step comparison	LD and LiteST
	ENC_GroupArrayCompare	Encoder two-dimensional array comparison	LD and LiteST
	ENC_ReadStatus	Encoder state read	LD and LiteST
	ENC_DigitalOutput	Encoder DO control	LD and LiteST
	ENC_ResetCompare	Encoder comparison output reset	LD and LiteST
	ENC_SetUnit	Set gear ratio	LD and LiteST
	ENC_SetLineRotationMode	Set axis operation mode	LD and LiteST

Category	Name	Function Description	Language Support
	HC_Counter	High-speed counter enable	LD and LiteST
	HC_Preset	High-speed counter preset	LD and LiteST
	HC_TouchProbe	High-speed counter probe	LD and LiteST
	HC_Compare	High-speed counter comparison	LD and LiteST
	HC_ArrayCompare	High-speed counter array comparison	LD and LiteST
	HC_StepCompare	High-speed counter step comparison	LD and LiteST
Bus encoder axis and high-speed counter instructions (Easy)	ENC_Counter	Encoder enable	LD and LiteST
	ENC_Reset	Reset encoder	LD and LiteST
	ENC_Preset	Encoder preset	LD and LiteST
	ENC_TouchProbe	Encoder probe	LD and LiteST
	ENC_ArrayCompare	Encoder one-dimensional array comparison	LD and LiteST
	ENC_StepCompare	Encoder one-dimensional step comparison	LD and LiteST
	ENC_Compare	Single-point comparison output	LD
	ENC_GroupArrayCompare	Encoder two-dimensional array comparison	LD and LiteST
	ENC_ReadStatus	Encoder state read	LD and LiteST
	ENC_DigitalOutput	Encoder DO control	LD and LiteST
	ENC_ResetCompare	Encoder comparison output reset	LD and LiteST
	ENC_SetUnit	Set gear ratio	LD and LiteST
	ENC_SetLineRotationMode	Set axis operation mode	LD and LiteST
	HC_Counter	High-speed counter enable	LD and LiteST
	HC_Preset	High-speed counter preset	LD and LiteST

Category	Name	Function Description	Language Support
	HC_TouchProbe	High-speed counter probe	LD and LiteST
	HC_Compare	High-speed counter comparison	LD and LiteST
	HC_ArrayCompare	High-speed counter array comparison	LD and LiteST
	HC_StepCompare	High-speed counter step comparison	LD and LiteST
High-speed counter instructions (H5U)	HC_Counter	High-speed counter enable	LD and LiteST
	HC_Preset	High-speed counter preset	LD and LiteST
	HC_TouchProbe	Probe	LD and LiteST
	HC_Compare	High-speed counter comparison	LD and LiteST
	HC_ArrayCompare	High-speed counter array comparison	LD and LiteST
	HC_StepCompare	High-speed counter step comparison	LD and LiteST
Timer	TPR	Pulse timer	LD and LiteST
	TONR	Connection delay timer	LD and LiteST
	TOFR	Off delay timer	LD and LiteST
	TACR	Accumulation timer	LD and LiteST
Pointer instructions	PTGET	Pointer variable assignment	LD
	PTINC	Pointer variable address incremented by 1	LD
	PTDEC	Pointer variable address decremented by 1	LD
	PTADD	Pointer variable address addition	LD
	PTSUB	Pointer variable address subtraction	LD
	PTSET	Pointer variable assignment	LD
	PTMOV	Pointer variable mutual assignment	LD
	PTLD>	Pointer variable contact comparison greater than	LD
	PTLD>=	Pointer variable contact comparison greater than or equal to	LD

Category	Name	Function Description	Language Support
	PTLD<=	Pointer variable contact comparison less than or equal to	LD
	PTLD=	Pointer variable contact comparison equal to	LD
	PTLD<>	Pointer variable contact comparison not equal to	LD
	PTAND>	Pointer variable AND contact comparison greater than	LD
	PTAND>=	Pointer variable AND contact comparison greater than or equal to	LD
	PTAND<	Pointer variable AND contact comparison less than	LD
	PTAND<=	Pointer variable AND contact comparison less than or equal to	LD
	PTAND=	Pointer variable AND contact comparison equal to	LD
	PTAND<>	Pointer variable AND contact comparison not equal to	LD
	PTOR>	Pointer variable OR contact comparison greater than	LD
	PTOR>=	Pointer variable OR contact comparison greater than or equal to	LD
	PTOR<	Pointer variable OR contact comparison less than	LD
	PTOR<=	Pointer variable OR contact comparison less than or equal to	LD
	PTOR=	Pointer variable OR contact comparison equal to	LD
	PTOR<>	Pointer variable OR contact comparison not equal to	LD
Communication protocol instructions	SerialSR	Serial port free protocol transmission and reception	LD
	SerialSend	Serial port free protocol transmission	LD and LiteST

Category	Name	Function Description	Language Support
	SerialRcv	Serial port free protocol reception	LD and LiteST
	MB_Master	Transmission and reception through the serial Modbus protocol	LD and LiteST
	MB_Client	Transmission and reception through the Modbus TCP protocol	LD and LiteST
	NTP	SysHC_NTPClient	LD
	TCP_Listen	TCP listening	LD
	TCP_Accept	Accept TCP connection request	LD
	TCP_Connect	Initiate TCP connection request	LD
	TCP_Close	Close TCP connection	LD
	TCP_Send	Send TCP data	LD
	TCP_Receive	Receive TCP data	LD
	UDP_Bind	Bind UDP socket	LD
	UDP_Send	Send UDP data	LD
	UDP_Receive	Receive UDP data	LD
	UDP_JoinGroup	Add UDP port to multicast group	LD
	SFTP_GetFile	Obtain a file	LD and LiteST
	SFTP_PutFile	Upload a file	LD and LiteST
	SFTP_Get FileList	Obtain a file list	LD and LiteST
	SFTP_CreateDir	Create a directory	LD and LiteST
	SFTP_Delete File	Delete a file or directory	LD and LiteST
	SFTP_MoveFile	Move a file or directory	LD and LiteST
	SFTP_Rename File	Rename a file or directory	LD and LiteST
	ETC_ReadParameter_CoE	Read SDO parameters of ETC_RestartMaster slave	LD and LiteST
	ETC_WriteParameter_CoE	Write SDO parameters of ETC_RestartMaster slave	LD and LiteST
		Restart EtherCAT master	LD and LiteST

Category	Name	Function Description	Language Support
	ETC_RestartMaster		
	EIP_Generic_Service	Call Generic service	LD
	EIP_Get_Attributes_All	Call Get_Attributes_All service	LD
	EIP_Get_Attribute_Single	Call Get_Attribute_Single service	LD
	EIP_Set_Attributes_All	Call Set_Attributes_All service	LD
	EIP_Set_Attribute_Single	Call Set_Attribute_Single service	LD
	EIP_Apply_Attributes	Call Apply_Attributes service	LD
	EIP_NOP	Call NOP service	LD
	EIP_Reset	Call Reset service	LD
	EIP_Start	Call Start service	LD
	EIP_Stop	Call Stop service	LD
	EIP_OpenWithDataSize	Create EIP connection	LD
	EIP_Close	Close EIP connection	LD
	EIP_Read	Read CIP class 3	LD
	EIP_Write	Write CIP class 3	LD
	OPCUA_Server_Shutdown	Shut down OPC UA server	LD and LiteST
	CANBus_DevRxMsg	Receive CAN free protocol data	LD and LiteST
	CANBus_DevTxMsg	Send CAN free protocol data	LD and LiteST
FB/FC instructions	PROG_AUTH	Program block (FB/FC) authorization verification	LD
			LD

Category	Name	Function Description	Language Support
User program backup instructions	SysHC_Appbackup	Back up the current running program of PLC	
Other instructions	PID	PID calculation	LD
	PID_AT	Universal PID controller	LD
	PID_ATC	Bipolar temperature PID controller	LD

5 Instruction Description

5.1 Program Logic Instructions

5.1.1 Contact Instructions

5.1.1.1 List of Instructions

The following table lists the contact instructions.

Category	Instruction	Function	Language Support
Contact instructions	LD	Load NO contact	LD
	LDI	Load NC contact	LD
	AND	Serial connection of NO contacts	LD
	ANI	Serial connection of NC contacts	LD
	OR	Parallel connection of NO contacts	LD
	ORI	Parallel connection of NC contacts	LD
	LDP	Obtain pulse rising edge	LD
	LDF	Obtain pulse falling edge	LD
	ANDP	Serial connection of pulse rising edge	LD
	ANDF	Serial connection of pulse falling edge	LD
	ORP	Parallel connection of pulse rising edge	LD
	ORF	Parallel connection of pulse falling edge	LD
	MEP	Conversion of operation result to rising edge pulse	LD
	MEF	Conversion of operation result to falling edge pulse	LD

5.1.1.2 LD

LD -- Load NO contact

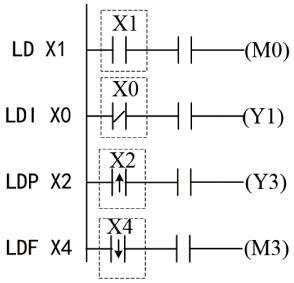
16-bit Instruction	LD: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√			-	-	√	-	-	-

Function and Instruction Description

The LD instruction is applicable to contacts starting from the left bus.



5.1.1.3 LDI

LDI -- Load NC contact

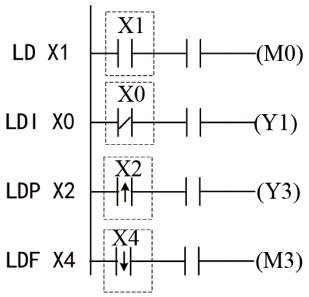
16-bit Instruction	LDI: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The LDI instruction is applicable to contacts starting from the left bus.



5.1.1.4 LDP

LDP -- Obtain pulse rising edge

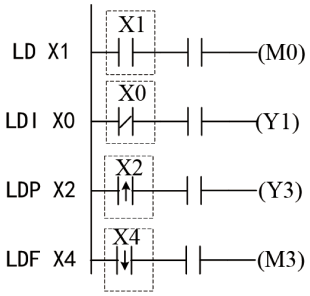
16-bit Instruction	LDP: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The LDP instruction, applicable to contacts starting from the left bus, detects the rising edge of a contact signal. If rising transition is detected in the signal, the contact becomes active, and it becomes inactive upon the next scan operation.



5.1.1.5 LDF

LDF -- Obtain pulse falling edge

16-bit Instruction	LDF: Continuous execution

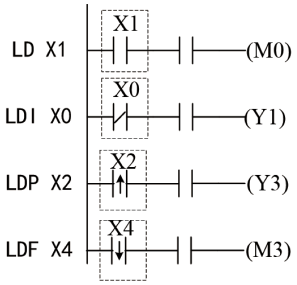
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The LDF instruction, applicable to contacts starting from the left bus, detects the falling edge of a contact signal. If falling transition is detected in the signal, the contact becomes active, and it becomes inactive upon the next scan operation.



5.1.1.6 AND

AND -- Serial connection of NO contacts

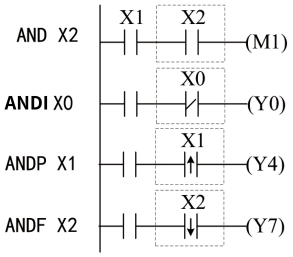
16-bit Instruction	AND: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The AND instruction is used for state operations of serial contacts. This instruction reads the state of the designated serial contact and performs an AND operation on the contact state and the logic operation result of the contact. The AND result is stored in the cache.



5.1.1.7 ANDI

ANDI -- Serial connection of NC contacts

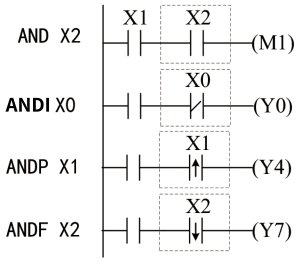
16-bit Instruction	ANDI: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The ANDI instruction is used for state operations of serial contacts. This instruction reads the state of the designated serial contact and performs an AND operation on the contact state and the logic operation result of the contact. The AND result is stored in the cache.



5.1.1.8 ANDP

ANDP -- Serial connection of pulse rising edge

16-bit Instruction	ANDP: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

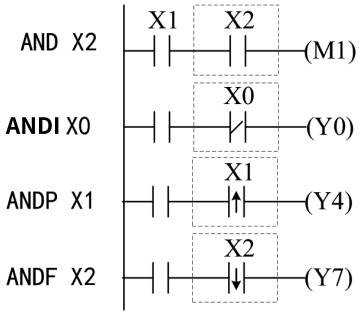
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The ANDP instruction is used for state operations of serial contacts. This instruction reads the state of the designated serial contact and performs an AND operation on the contact state and the logic operation result of the contact. The AND result is stored in the cache.

The ANDP instruction obtains the rising edge transition state of the contact for an AND operation.



5.1.1.9 ANDF

ANDF -- Serial connection of pulse falling edge

16-bit Instruction	ANDF: Continuous execution

32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

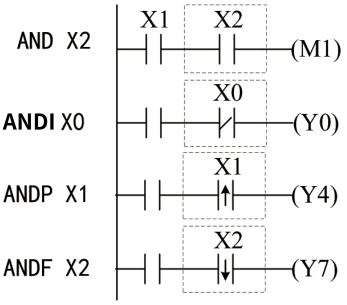
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The ANDF instruction is used for state operations of serial contacts. This instruction reads the state of the designated serial contact and performs an AND operation on the contact state and the logic operation result of the contact. The AND result is stored in the cache.

The ANDF instruction obtains the falling edge transition state of the contact for an AND operation.



5.1.1.10 OR

OR -- Parallel connection of NO contacts

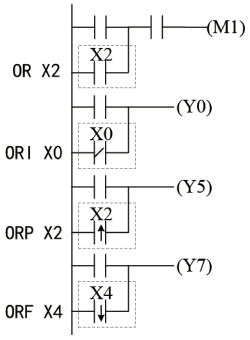
16-bit Instruction	OR: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The OR instruction is used for state operations of parallel contacts. This instruction reads the state of the designated parallel contact and performs an OR operation on the contact state and the logic operation result of the contact. The OR result is stored in the cache.



5.1.1.11 ORI

ORI -- Parallel connection of NC contacts

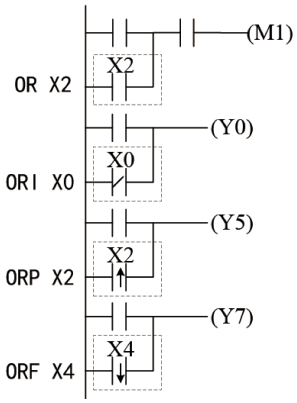
16-bit Instruction	ORI: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The ORI instruction is used for state operations of parallel contacts. This instruction reads the state of the designated parallel contact and performs an OR operation on the contact state and the logic operation result of the contact. The OR result is stored in the cache.



5.1.1.12 ORP

ORP -- Parallel connection of pulse rising edge

16-bit Instruction	ORP: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

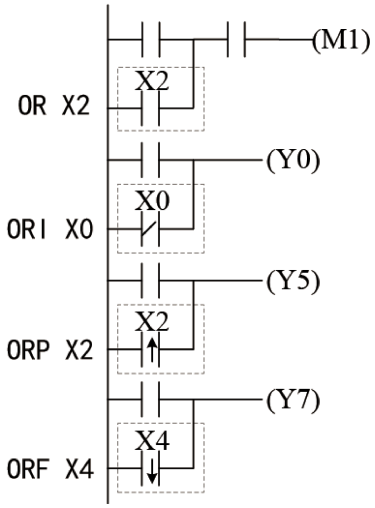
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

5.1.1.10 OR and ORI are used for state operations of parallel contacts. These instructions read the status of the designated parallel contact and perform an OR operation on the contact status and the contact's logical operation result. The OR result is stored in the cache.

The ORP instruction obtains the rising edge transition state of the contact for an OR operation.



5.1.1.13 ORF

ORF -- Parallel connection of pulse falling edge

16-bit Instruction	ORF: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

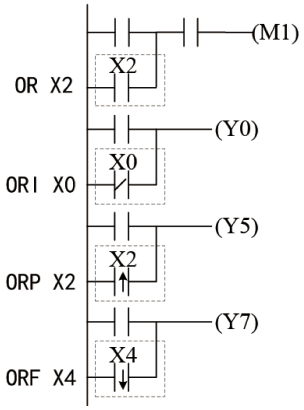
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

OR and ORI are used for state operations of parallel contacts. These instructions read the status of the designated parallel contact and perform an OR operation on the contact status and the contact's logical operation result. The OR result is stored in the cache.

The ORF instruction obtains the falling edge transition state of the contact for an OR operation.



5.1.1.14 MEP

MEP -- Conversion of operation result to rising edge pulse

16-bit Instruction	MEP: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	BOOL

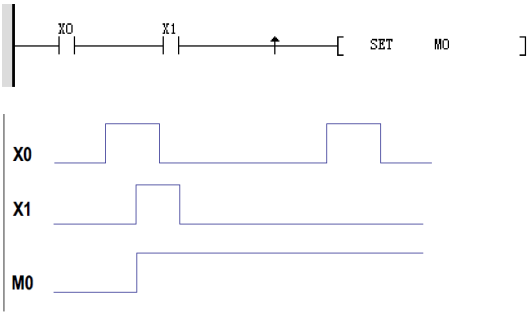
Function and Instruction Description

The operation results prior to the MEP instruction become conductive when the driving contacts turn from OFF to ON.

The MEP instruction simplifies the process of changing driving contacts to pulses when multiple contact points connect in a series.

Instruction Example

ON during rising edge of operation results



5.1.1.15 MEF

MEF-- Conversion of operation result to falling edge pulse

16-bit Instruction	MEF: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	BOOL

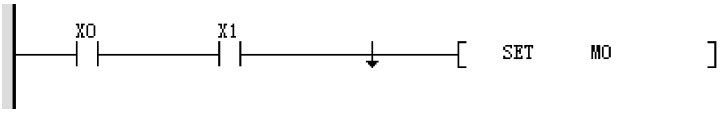
Function and Instruction Description

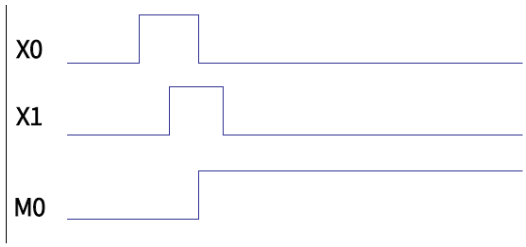
The operation results prior to the MEF instruction become conductive when the driving contacts turn from ON to OFF.

The MEF instruction simplifies the process of changing driving contacts to pulses when multiple contact points connect in a series.

Instruction Example

ON during falling edge of operation results





5.1.2 Output Control Instructions

5.1.2.1 List of Instructions

The following table lists the output control instructions.

Category	Instruction	Function	Language Support
Output control instructions	OUT	Coil drive	LD
	SET	SET action storage coil	LD
	RST	Contact or cache clearing	LD
	ZSET	Set all data	LD and LiteST
	ZRST	Reset all data	LD and LiteST
	PLS	Pulse rising edge detection coil	LD
	PLF	Pulse falling edge detection coil	LD
	ALT	Alternate output	LD
	R_TRIG	Rising edge detection	LD and LiteST
	F_TRIG	Falling edge detection	LD and LiteST

5.1.2.2 OUT

OUT: Coil drive

16-bit instruction	OUT: continuous execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
D	Output element	Bit element or variable to output	-	BOOL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	√[1]	√	√	-	-	√	-	-	-

NOTE

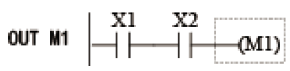
[1] The X element is not supported. The S element applies to separate instructions. For details, see 5.3 SFC Instructions.

Function and Instruction Description

The OUT instruction outputs the logical operation results prior to this instruction to the designated element.

If the operand D is a pointer variable, use PGET to initialize the pointer variable. Otherwise, the system will report an error indicating that the address is invalid.

Example



5.1.2.3 SET

SET: SET action storage coil

16-bit instruction	SET: continuous execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type

D	Output element	Bit element or variable to output	-	BOOL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	√[1]	√	√	-	-	√	-	-	-

NOTE

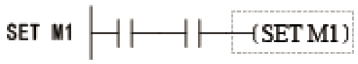
[1] The X element is not supported. The S element applies to separate instructions. For details, see 5.3 SFC Instructions.

Function and Instruction Description

When the SET instruction is driven, the component designated by this instruction is set to ON and remains so regardless of whether the instruction is still driven. You can use the RST instruction to set the component to OFF.

If the operand D is a pointer variable, use PGET to initialize the pointer variable. Otherwise, the system will report an error indicating that the address is invalid.

Example



5.1.2.4 RST

RST: Contact or cache clearing

16-bit instruction	RST: Continuous execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
D	Output component	Bit component or variable to be output	-	BOOL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	√[1]	√	√	-	-	√	-	-	-

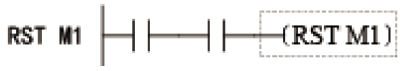
NOTE

[1] The X element is not supported. The S element applies to separate instructions. For details, see 5.3 SFC Instructions.

Function

When the RST instruction is driven, the component designated by this instruction is set to OFF and remains so regardless of whether the instruction is still driven. Use the SET instruction to set the component to ON.

Example



5.1.2.5 ZSET

ZSET -- Set all data

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ZSET	Set all data	ZSET ??? ???	-

16-bit Instruction	ZSET (bit): Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D1	Element start address	Start address of elements or variables to be set in batches	-	BOOL
D2	Element end address	End address of elements or variables to be set in batches	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

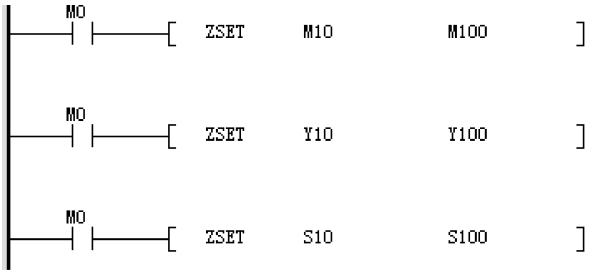
The ZSET instruction sets the values of all variables between D1 and D2 to 1. D1 and D2 can be set to Y, M, S, or B bit elements or other bit variables.

Note the following:

D1 and D2 must be of the same element type.

D1 cannot be greater than D2. If they are the same, only the specified element is set.

Instruction Example



Additional Information

Bit elements Y, M, S, and B can be set independently using the SET instruction.

5.1.2.6 ZRST

ZRST -- Reset all data

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ZRST	Reset all data	ZRST ??? ???	ZRST(???,???);

16-bit Instruction	ZRST: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D1	Element start address	Start address of elements or variables to be reset in batches	-	BOOL, INT, and DINT Array*(D2-D1+1)
D2	Element end address	End address of elements or variables to be reset in batches	-	BOOL, INT, and DINT Array*(D2-D1+1)

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D1	√[1]	√	√	√	√	-	-	-	-
D2	√[1]	√	√	√	√	-	-	-	-

NOTE

[1]: The X element is not supported.

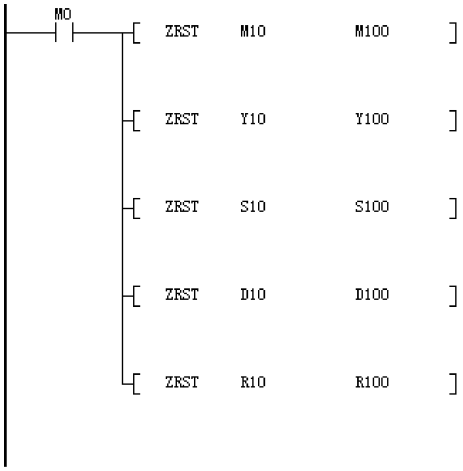
Function and Instruction Description

The ZRST instruction clears values of all variables between D1 and D2. D1 and D2 can be specified as word elements, word variables, bit elements, or bit variables.

Note the following:

- D1 and D2 must be of the same element type.
- D1 cannot be greater than D2. If they are the same, only the specified element is reset.

Instruction Example



Additional Information

Bit elements Y, M, S, and B can be reset independently using the RST instruction. Word elements D, R, and W can be reset independently using the ZRST instruction.

5.1.2.7 PLS

PLS -- Detect coil on pulse rising edge

16-bit Instruction	PLS: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	Output element	Bit element or variable to output	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	√	√	-	-	-	-	-	-

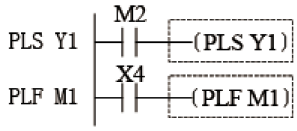
NOTE

[1]: The X element is not supported.

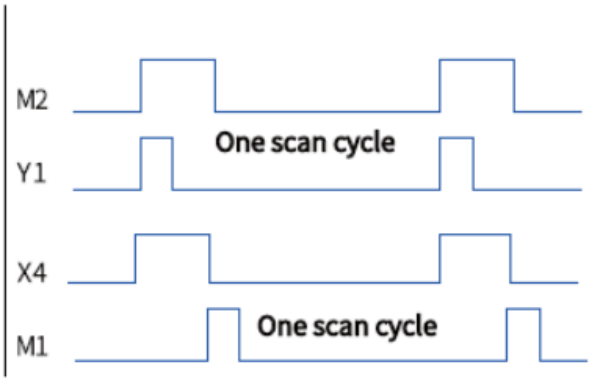
Function and Instruction Description

When the PLS instruction is driven by the rising edge, the element designated by this instruction is set to ON and remains so within only one scan cycle.

Instruction Example



Timing Diagram



5.1.2.8 PLF

PLF -- Pulse (F) falling edge detection coil

16-bit Instruction	PLF: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	Output element	Bit element or variable to output	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	√	√	-	-	-	-	-	-

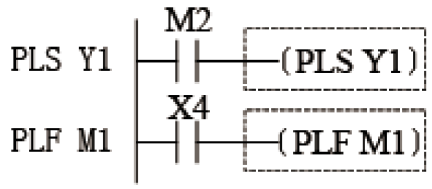
NOTE

[1]: The X element is not supported.

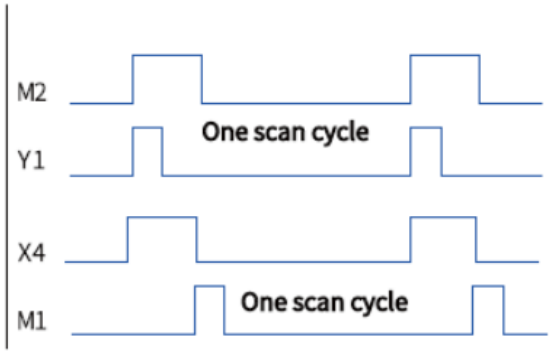
Function and Instruction Description

When the PLF instruction is driven by the falling edge, the element designated by this instruction is set to ON and remains so within only one scan cycle.

Instruction Example



Timing Diagram



5.1.2.9 ALT

ALT -- Alternate output

When the driving conditions are met, the ALT instruction reverses the state (ON or OFF) of the bit element D.

16-bit Instruction	ALT: Continuous execution/ALTP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	Execution device	Bit element	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	√	√	-	-	-	-	-	-

NOTE

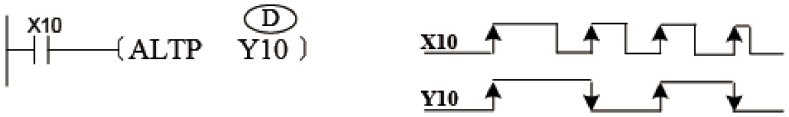
[1]: The X element is not supported.

Function and Instruction Description

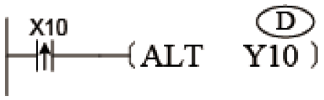
The ALT instruction reverses the state of the D element when the flow is active. D is a bit element or bit variable.

The ALTP instruction of the pulse execution type is usually used.

Instruction Example



The action generated by the following instruction is the same as that generated by the ALTP instruction.



5.1.2.10 R_TRIG

R_TRIG: Rising edge detection

The R_TRIG instruction is used to output a TRUE signal for only one task cycle when an input signal transitions from FALSE to TRUE (rising edge).

Instruction	Name	LD Expression	LiteST Expression
R_TRIG	Rising edge detection		R_TRIG(CLK:=,Q=>);

16-bit instruction	Rising edge detection			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
CLK	Input value	Bit element	-	BOOL
Q	Input result	Bit element	-	BOOL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
CLK	√[1]	√	√	-	-	-	-	-	-
Q	√[1]	√	√	-	-	-	-	-	-

> **NOTE**
>
> [1] The X element is not supported.

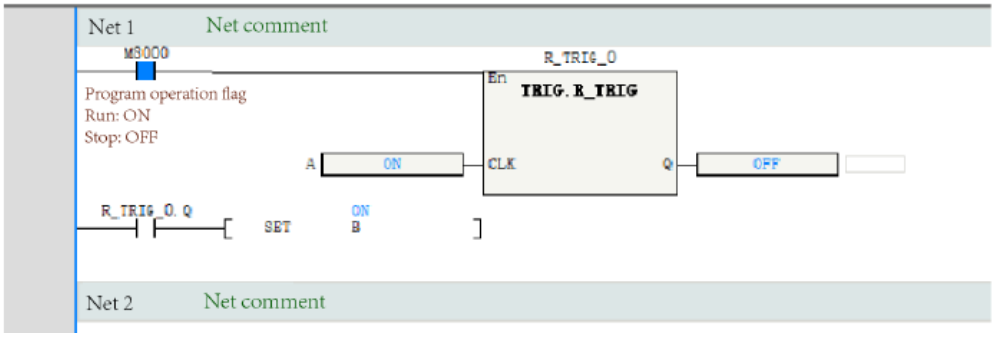
Function and Instruction Description

This instruction is used to output a TRUE signal for only one task cycle when an input signal transitions from FALSE to TRUE (rising edge).

> **NOTE**
>
> The R_TRIG instruction is used in the same way as a function block. You need to declare it in the instance table before calling.

Example

LD



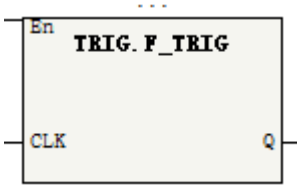
LiteST

```
1   R_TRIG_0(CLK := A ON ,Q => );
2   IF R_TRIG_0.Q OFF THEN
3       B ON := TRUE;
4   END_IF;
```

5.1.2.11 F_TRIG

F_TRIG: Falling edge detection

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

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

16-bit instruction	Down trigger

32-bit instruction	-			
Operands	Name	Description	Value range	Data type
CLK	Input value	Bit element	-	BOOL
Q	Input result	Bit element	-	BOOL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
CLK	√[1]	√	√	-	-	-	-	-	-
Q	√[1]	√	√	-	-	-	-	-	-

NOTE

[1] The X element is not supported.

Function

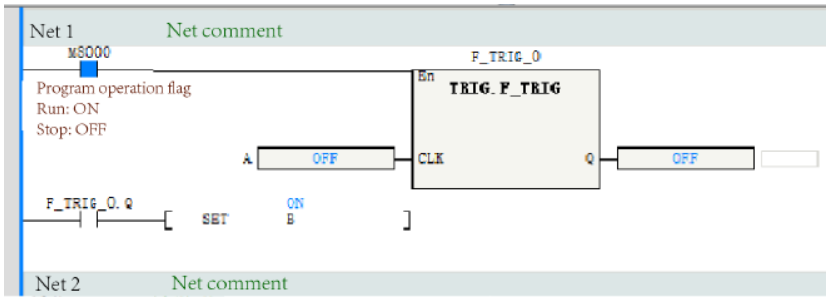
This instruction is used to output a TRUE signal for only one task cycle when an input signal transitions from TRUE to FALSE (falling edge).

NOTE

The F_TRIG instruction is used in the same way as a function block. You need to declare it in the instance table before calling.

Example

LD



LiteST

```
F_TRIG_0(CLK := A OFF,Q => );
IF F_TRIG_0.Q OFF THEN
    B ON := TRUE;
END_IF;
```

5.1.3 Energy Flow Control Instructions

5.1.3.1 INV

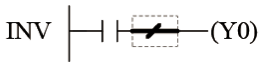
INV -- Operation result inversion

16-bit Instruction	INV: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	-

Function and Instruction Description

The INV instruction inverts the logic operation result prior to this instruction. The result is stored in the cache. After the INV instruction is executed, the flow state switches from ON to OFF, or vice versa.

Instruction Example



5.1.4 Immediate Input (Contact) Instructions

5.1.4.1 List of Instructions

The following table lists the immediate input (contact) instructions.

Category	Instruction	Function	Language Support
Immediate input (contact) instructions	ILD	Load NO immediate contact	LD and LiteST
	ILDI	Load NC immediate contact	LD and LiteST
	IAND	Serial NO immediate contact	LD and LiteST
	IANDI	Serial NC immediate contact	LD and LiteST
	IOR	Parallel NO immediate contact	LD and LiteST
	IORI	Parallel NC immediate contact	LD and LiteST

NOTE

- It takes 5 μs more to execute an immediate input (contact) instruction than a contact instruction.
- Immediate input (contact) instructions are supported only when the software version is V4.10.1.0 or later and the firmware version is V6.3.0.0 or later.

5.1.4.2 ILD

ILD -- Load NO immediate contact

16-bit Instruction	ILD: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	X elements that need to be read and loaded immediately	• H5U: X0 to X7 and X10 to X17, a total of 16 points • Easy: X0 to X7	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√[1]	-	-	-	-	-	-	-	-

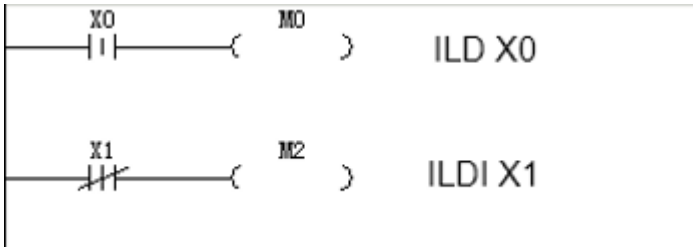
NOTE

[1]: Only X0 to XN elements are supported. N indicates the maximum number of input points of the PLC.

Function and Instruction Description

The LDI instruction immediately reads the external input state of X elements at the point where the current program is executed and updates the state to the contact state.

Instruction Example



5.1.4.3 ILDI

ILDI -- Load NC immediate contact

16-bit Instruction	ILDI: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	X elements that need to be read and loaded immediately	• H5U: X0 to X7 and X10 to X17, a total of 16 points • Easy: X0 to X7	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√[1]	-	-	-	-	-	-	-	-

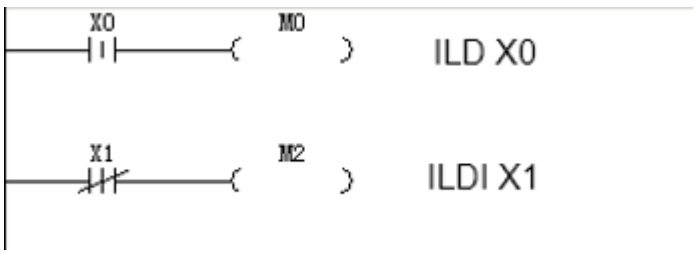
NOTE

[1]: Only X0 to XN elements are supported. N indicates the maximum number of input points of the PLC.

Function and Instruction Description

The ILDI instruction immediately reads the external input state of X elements at the point where the current program is executed and updates its reverse state to the contact state.

Instruction Example



5.1.4.4 IAND

IAND -- Serial NO immediate contact

16-bit Instruction	IAND: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	X elements that need to be read and loaded immediately	H5U: X0 to X7 and X10 to X17, a total of 16 points Easy: X0 to X7	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√[1]	-	-	-	-	-	-	-	-

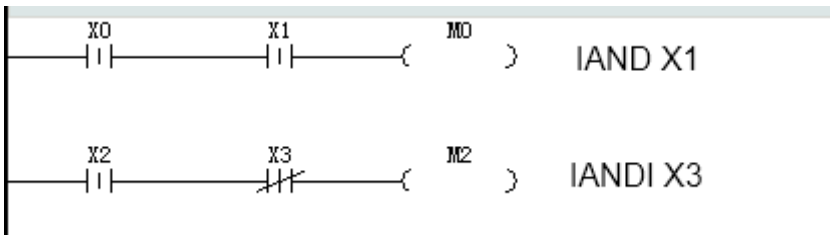
NOTE

[1]: Only X0 to XN elements are supported. N indicates the maximum number of input points of the PLC.

Function and Instruction Description

The IAND instruction is used for state operations of serial contacts. This instruction immediately reads the external input state of the designated serial contact X and performs an AND operation on the contact state and the logic operation result of the contact. The AND result is stored in the cache.

Instruction Example



5.1.4.5 IANDI

IANDI -- Serial NC immediate contact

16-bit Instruction	IANDI: Continuous execution
32-bit Instruction	-

Operand	Name	Description	Range	Data Type
S	Bit element	X elements that need to be read and loaded immediately	H5U: X0 to X7 and X10 to X17, a total of 16 points Easy: X0 to X7	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√[1]	-	-	-	-	-	-	-	-

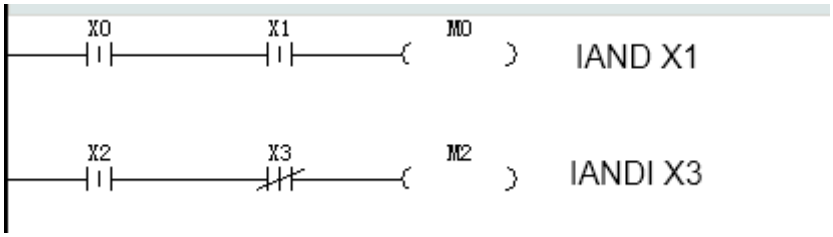
NOTE

[1]: Only X0 to XN elements are supported. N indicates the maximum number of input points of the PLC.

Function and Instruction Description

The IANDI instruction is used for state operations of serial contacts. This instruction immediately reads the external input state of the designated serial contact X and performs an AND operation on the contact state and the logic operation result of the contact. The AND result is stored in the cache.

Instruction Example



5.1.4.6 IOR

IOR -- Parallel NO immediate contact

16-bit Instruction	IOR: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	X elements that need to be read and loaded immediately	• H5U: X0 to X7 and X10 to X17, a total of 16 points • Easy: X0 to X7	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√[1]	-	-	-	-	-	-	-	-

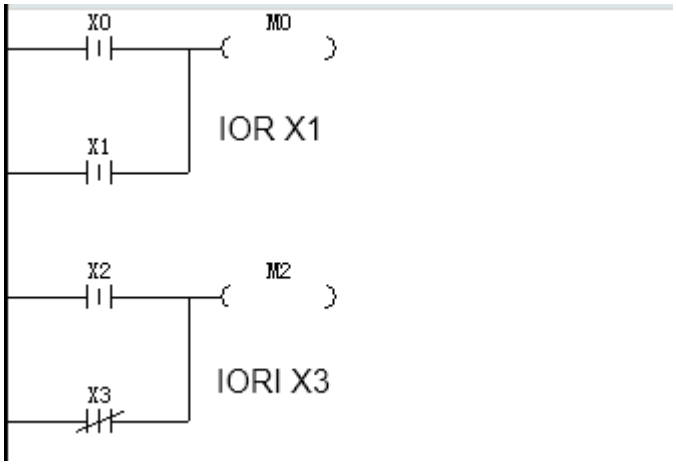
NOTE

[1]: Only X0 to XN elements are supported. N indicates the maximum number of input points of the PLC.

Function and Instruction Description

The IOR instruction is used for state operations of parallel contacts. This instruction immediately reads the external input state of the designated parallel contact X and performs an OR operation on the contact state and the logic operation result of the contact. The OR result is stored in the cache.

Instruction Example



5.1.4.7 IORI

IORI -- Parallel NC immediate contact

16-bit Instruction	IORI: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	X elements that need to be read and loaded immediately	• H5U: X0 to X7 and X10 to X17,	BOOL

			a total of 16 points • Easy: X0 to X7	

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√[1]	-	-	-	-	-	-	-	-

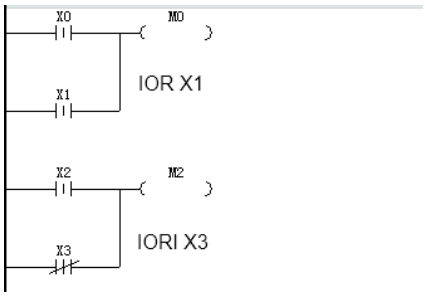
NOTE

[1]: Only X0 to XN elements are supported. N indicates the maximum number of input points of the PLC.

Function and Instruction Description

The IORI instruction is used for state operations of parallel contacts. This instruction immediately reads the external input state of the designated parallel contact X and performs an OR operation on the contact state and the logic operation result of the contact. The OR result is stored in the cache.

Instruction Example



5.1.5 Immediate Output Instructions

5.1.5.1 List of Instructions

The following table lists the immediate output instructions.

Category	Instruction	Function	Language Support
Immediate output instructions	IOUT	Immediately output single local I/O	LD and LiteST
	ISET	Immediately output single local I/O	LD and LiteST
	IRST	Immediately reset single local I/O	LD and LiteST
	IZSET	Immediately set local I/Os in specified range in batches	LD and LiteST

Category	Instruction	Function	Language Support
	IZRST	Immediately reset local I/Os in specified range in batches	LD and LiteST

NOTE

- It takes 5 μs more to execute an immediate output control instruction than an output control instruction.
- Immediate output control instructions are supported only when the software version is V4.10.1.0 or later and the firmware version is V6.3.0.0 or later.

5.1.5.2 IOUT

IOUT -- Immediately output specific single local I/O

16-bit Instruction	IOUT: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	Output element	Local I/O that needs to be output immediately	H5U: Y0 to Y7 and Y10 to Y15 Easy: Y0 to Y7	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	-	-	-	-	-	-	-	-

NOTE

[1]: Only Y0 to YN elements are supported. N indicates the maximum number of output points of the PLC.

Function and Instruction Description

The IOUT instruction immediately outputs the logic operation result prior to the instruction to the Y output point of the specified local I/O.

Instruction Example



5.1.5.3 ISET

ISET -- Immediately set specific single local I/O

16-bit Instruction	ISET: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	Output element	Local I/O that needs to be set immediately	H5U: Y0 to Y7 and Y10 to Y15 Easy: Y0 to Y7	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	-	-	-	-	-	-	-	-

NOTE

[1]: Only Y0 to YN elements are supported. N indicates the maximum number of output points of the PLC.

Function and Instruction Description

When the ISET instruction is driven to be active, it immediately outputs the specific local Y output point and keeps it in the Output state, regardless of whether the ISET instruction is still be driven. The IRST instruction can be used to close the output of this element.

Instruction Example



5.1.5.4 IRST

IRST -- Immediately reset specific single local I/O

16-bit Instruction	IRST: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	Output element	Local I/O that needs to be reset immediately	H5U: Y0 to Y7 and Y10 to Y15 Easy: Y0 to Y7	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	-	-	-	-	-	-	-	-

NOTE

[1]: Only Y0 to YN elements are supported. N indicates the maximum number of output points of the PLC.

Function and Instruction Description

When the IRST instruction is driven to be active, it immediately closes the specific local Y output point and keeps it in the Closed state, regardless of whether the IRST instruction is still be driven. The ISET instruction can be used to immediately output this element.

Instruction Example



5.1.5.5 IZSET

IZSET -- Immediately set local I/Os in specified range in batches

16-bit Instruction	IZSET: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type

D1	Start address of output element	Start address of local I/Os that need to be set immediately	• H5U: Y0 to Y7 and Y10 to Y15 • Easy: Y0 to Y7	BOOL
D2	End address of output element	End address of local I/Os that need to be set immediately	• H5U: Y0 to Y7 and Y10 to Y15 • Easy: Y0 to Y7	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D1	√[1]	-	-	-	-	-	-	-	-
D2	√[1]	-	-	-	-	-	-	-	-

NOTE

[1]: Only Y0 to YN elements are supported. N indicates the maximum number of output points of the PLC.

Function and Instruction Description

When the IZSET instruction is driven to be active, it immediately outputs local Y output points in the range of D1 to D2 and keeps them in the Output state, regardless of whether the ISET instruction is still be driven. The IZRST instruction can be used to close the output of elements in this range.

Instruction Example

When M1 is active, Y0 to Y3 immediately output data.

```
M1
─┤ ├─[ IZSET    Y0          Y3          ]
```

5.1.5.6 IZRST

IZRST -- Immediately reset local I/Os in specified range in batches

16-bit Instruction	IZRST: Continuous execution
32-bit Instruction	-

Operand	Operand	Description	Range	Data Type
D1	Start address of output element	Start address of local I/Os that need to be reset immediately	• H5U: Y0 to Y7 and Y10 to Y15 • Easy: Y0 to Y7	BOOL
D2	End address of output element	End address of local I/Os that need to be reset immediately	• H5U: Y0 to Y7 and Y10 to Y15 • Easy: Y0 to Y7	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D1	√[1]	-	-	-	-	-	-	-	-
D2	√[1]	-	-	-	-	-	-	-	-

NOTE

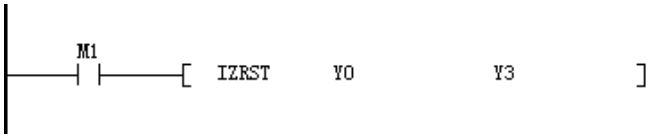
[1]: Only Y0 to YN elements are supported. N indicates the maximum number of output points of the PLC.

Function and Instruction Description

When the IZRST instruction is driven to be active, it immediately closes local Y output points in the range of D1 to D2 and keeps them in the Closed state, regardless of whether the IRST instruction is still be driven. The IZSET instruction can be used to immediately output elements in this range.

Instruction Example

When M1 is active, Y0 to Y3 immediately output data.



5.1.6 Immediate Input and Output Instructions

5.1.6.1 List of Instructions

he following table lists the immediate read and write instructions.

Category	Version	Name	Function	Language Support
Immediate read and write instruction	Software version: V4.10.1.0 and later Firmware version: V6.3.0.0 and later	IRDBIT	Immediately read bits	LiteST
		IWRBIT	Immediately write bits	LiteST

5.1.6.2 IRDBIT

IRDBIT -- Immediately read bits

Return Value Type	Immediately read bits		
Operand	Description	Range	Data Type
S1	Target bits to read	•H5U: X0 to X7 and X10 to X17, a total of 16 points •Easy: X0 to X7	BOOL
S2	Read result	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	√	-	-	-	-	-	-	-	-	-	-	-
S2	√	√	√	-	-	-	-	-	-	-	-	-

NOTE

Only X0 to XN elements are supported. N indicates the maximum number of input points of the PLC.

Function and Instruction Description

The IRDBIT instruction immediately reads the external input state of X elements at the point where the current program is executed and updates the state to S2.

Instruction Example

```
IRDBIT(X0,M0);
```

The PLC immediately reads the X0 state and updates the state to M0.

5.1.6.3 IWRBIT

IWRBIT -- Immediately write bits

Return Value Type	Immediately write bits		
Operand	Description	Range	Data Type
S1	Target bits to write	•H5U: Y0 to Y7 and Y10 to Y15 •Easy: Y0 to Y7	BOOL
S2	Values to write	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	√	-	-	-	-	-	-	-	-	-	-	-
S2	√	√	√	-	-	-	-	-	-	-	√	-

> **NOTE**
>
> Only Y0 to YN elements are supported. N indicates the maximum number of output points of the PLC.

Function and Instruction Description

The IWRBIT instruction immediately writes the S2 value to the Y output point of the specified local I/O where the current program is executed.

Instruction Example

```
IWRBIT(Y0,M0);
```

The PLC immediately writes the M0 value to Y0.

5.2 Process Control Instructions

5.2.1 List of Instructions

The following table lists the process control instructions.

Category	Instruction	Function	Language Support
Process control instructions	CJ	Conditional jump	LD
	LBL	Label	LD
	CALL	Call subprogram	LD
	SSRET	Conditional subprogram return	LD
	EI	Enable interrupt	LD and LiteST
	DI	Disable interrupt	LD and LiteST
	WDT	Watchdog timer reset	LD
	FOR	Start of a loop	LD
	NEXT	End of a loop	LD

5.2.2 CJ

CJ -- Conditional jump

16-bit Instruction	CJ: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Target label	Target label to jump to	-	-

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	-	-	-	L[1]

NOTE

[1]: Only the L element is supported.

5.2.3 LBL

LBL -- Labeling

16-bit Instruction	LBL: Continuous execution			
32-bit Instruction	-			
Operand	**Name**	**Description**	**Range**	**Data Type**
S	Label number	Current label number	-	-

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	-	-	-	L [1]

NOTE

[1]: Only the L element is supported.

Function and Instruction Description

- When the flow is active, the program automatically jumps from the address of the CJ instruction to the address specified by L***. Program execution continues after the jump, and the program instructions in the intermediate addresses are skipped.
- When the flow is inactive, the program is executed without jump. The CJ instruction is not executed.

Note the following:

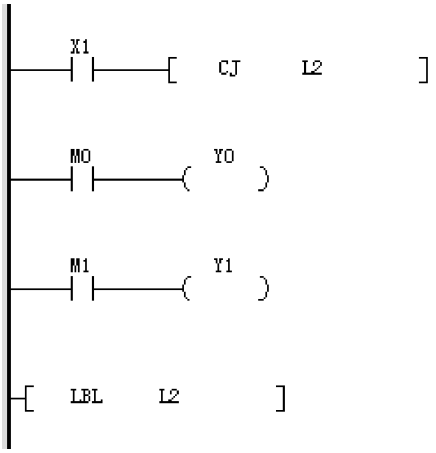
- The CJ instruction must work with the LBL instruction, and the target label must be located in the current program block. Jumping across program blocks is not allowed.
- The addresses defined by the label cannot be duplicate in the same program block.
- When part of a program does not need to be executed or two coils are used for output, this instruction can be used to avoid the double coil problem
- The CJ instruction can designate the same label repeatedly.

NOTE

The CJ instruction cannot be used in subprograms, interrupt subprograms, function blocks, and functions.

Instruction Example

The CJ instruction is used as follows in AutoShop:



When X1 is disconnected, the program scans normally. When X1 is connected, the program jumps directly to L2 upon detecting the CJ instruction.

5.2.4 CALL

CALL: Subroutine call

16-bit instruction	CALL: continuous execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Subroutine	No. of subroutine to be called	-	-

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	-	-	-	-	-	P [1]

NOTE

- [1] Only the PL element is supported.
- It is added automatically by AutoShop without user input.
- Up to six nested layers (including the main program layer) of subprograms are supported.

Function

When the flow is active, the program calls the subprogram specified by SBR_. After the subprogram is executed, the program returns to the next instruction of the CALL (or CALLP) statement to execute the subsequent statement.

Note the following about the SBR_ address pointer:

- The subprogram starting from SBR_ can be called in multiple locations or by another subprogram, but the number of nested layers cannot exceed six.
- A subroutine cannot be called within itself; otherwise, an infinite loop or program running timeout occurs.
- A subprogram cannot be called recursively.

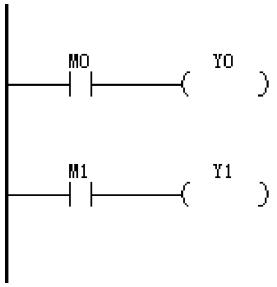
Example

The CALL instruction is used as follows in AutoShop:

- Main program



- Subprogram: SBR_001



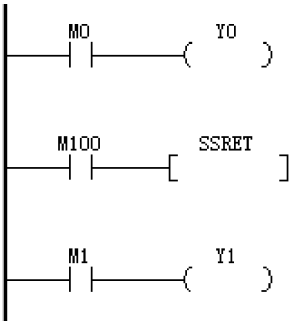
5.2.5 SSRET

SSRET -- Conditional subprogram return

16-bit Instruction	SSRET: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	-

Instruction Example

When M100 is ON, program execution directly returns to the main program. (The SSRET instruction can be executed only in a subprogram.)



5.2.6 EI

EI -- Enable interrupt

Instruction	Name	LD Expression	LiteST Expression
EI	Enable interrupt	-[EI]	EI();

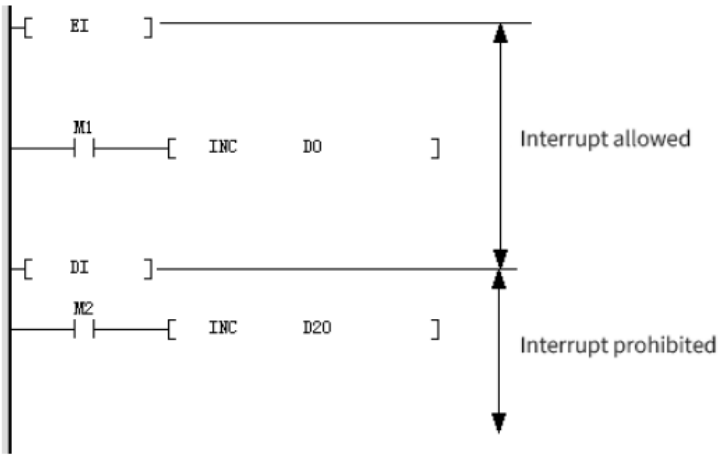
16-bit Instruction	EI: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	-

Function and Instruction Description

When the PLC program starts running, the interrupt function is disabled by default. The EI statement enables the interrupt function, and the 5.2.7 DI statement disables the interrupt function. If the program does not have a range in which the DI instruction can be inserted, the DI instruction cannot be used.

For the introduction and usage of the interrupt subprogram, see the "Interrupts" and "Subprograms" sections in the *AutoShop Programming and Application Manual*.

Interrupts are classified into the external signal input interrupt, high-speed count comparison interrupt, and timer interrupt.



5.2.7 DI

DI -- Disable interrupt

Instruction	Name	LD Expression	LiteST Expression
DI	Disable interrupt	-[DI]	DI();

16-bit Instruction	DI: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type

-	-	-	-	-

Function and Instruction Description

When the PLC program starts running, the interrupt function is disabled by default. The EI statement enables the interrupt function, and the DI statement disables the interrupt function. If the program does not have a range in which the DI instruction can be inserted, the DI instruction cannot be used.

For the introduction and usage of the interrupt subprogram, see the "Interrupts" and "Subprograms" sections in the *AutoShop Programming and Application Manual*.

Interrupts are classified into the external signal input interrupt, high-speed count comparison interrupt, and timer interrupt.

5.2.8 WDT

WDT -- Reset watchdog timer

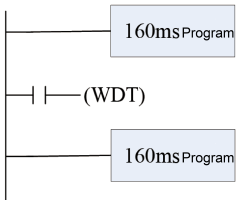
16-bit Instruction	WDT: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	-

Function and Instruction Description

The PLC uses an internal timer to monitor the user program execution cycle. If the execution cycle times out, the program is stopped and an alarm is generated. The WDT instruction resets the watchdog timer to make it start timing again, avoiding the timeout error.

An execution timeout error may occur when the operation performed by the user program is complicated, for example, too many loop calculations. To avoid this error, you can insert the WDT instruction (for example, insert it between the FOR and NEXT instructions) when necessary during programming.

Instruction Example



If the watchdog timer for the program is set to 200 ms, and the program scan cycle is 320 ms, running the program directly will cause the watchdog timer to time out. In this case, you can insert the WDT instruction to divide the program into two segments, each with a scan cycle of less than 200 ms.

5.2.9 FOR

FOR -- Start of a loop

16-bit Instruction	FOR: Continuous execution

32-bit Instruction	---			
Operand	Name	Description	Range	Data Type
S	Number of loops	Number of loops	-	INT

Function and Instruction Description

The FOR instruction identifies the start position and specifies the number of repetitions of the loop. It must be used with the 5.2.10 NEXT instruction. In the FOR instruction, S is the variable that specifies the number of loops.

The NEXT instruction identifies the end position of a loop. After the loop between the FOR and NEXT instructions is repeated for N times (specified in the FOR instruction), the PLC proceeds to subsequent execution.

The FOR-NEXT loop can be nested up to six layers (including the outermost layer). The PLC parses each FOR-NEXT loop layer in sequence during running. When the number of loops is large, the PLC scan cycle becomes longer, which may result in an error when the watchdog timer times out. To avoid this error, you can insert the WDT instruction between the FOR and NEXT instructions.

NOTE

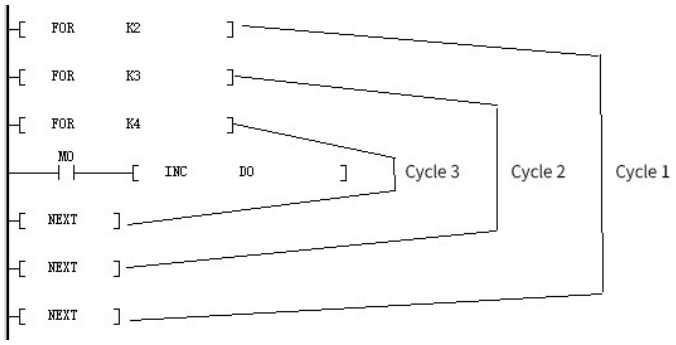
The FOR-NEXT loop can be nested for up to six layers.

Errors

- An error is reported in the following cases:
- The NEXT instruction precedes the FOR instruction.
- The FOR instruction exists without the NEXT instruction.
- The FOR and NEXT instructions are not equal in quantity.

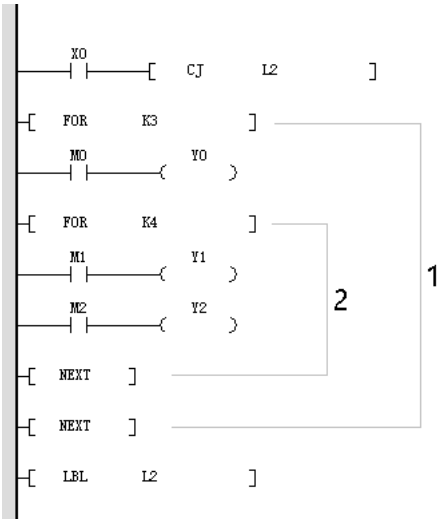
Instruction Example

- Example 1



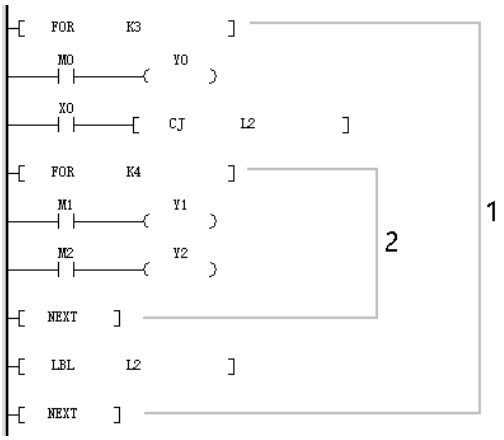
In the preceding figure, 1 indicates loop 1, 2 indicates loop 2, and 3 indicates loop 3. After loop 1 is executed twice, program execution after the NEXT instruction continues. Each time loop 1 is executed, loop 2 is repeated 3 times. Each time loop 2 is executed, loop 3 is repeated 4 times. Therefore, loop 3 is repeated a total of 24 times, and loop 2 is repeated a total of 6 times.

- Example 2



In the preceding figure, 1 indicates loop 1, and 2 indicates loop 2. To skip the FOR-NEXT loops, you can insert a CJ instruction. In this example, when X0 is OFF, loop 1 and loop 2 are executed. When X0 is ON, program execution jumps from the CJ instruction to L2, and programs between loop 1 and loop 2 are not executed.

- Example 3



In the preceding figure, 1 indicates loop 1, and 2 indicates loop 2. To skip the FOR-NEXT loop nested in a loop, you can also insert a CJ instruction. In this example, when X0 is OFF, loop 2 inside loop 1 is executed. When X0 is ON, program execution jumps from the CJ instruction to L2, and FOR-NEXT loop 2 nested in loop 1 is skipped.

5.2.10 NEXT

NEXT -- End of a loop

16-bit Instruction	NEXT: Continuous execution			
32-bit Instruction	---			
Operand	Name	Description	Range	Data Type
S	Number of loops	Number of loops	-	INT

Function and Instruction Description

Refer to the 5.2.9 FOR.

5.3 SFC Instructions

5.3.1 List of Instructions

The following table lists the SFC instructions.

Category	Instruction	Function	Language Support
SFC instruction	STL	Program jump to secondary bus	LD
	RET	Program return to primary bus	LD
	OUTSTL	Output program jump to secondary bus	LD
	SETSTL	Setting program jump to secondary bus	LD
	RSTSTL	Resetting program jump to secondary bus	LD

NOTE

The SFC instructions are only used in the main program and cannot be used in subprograms and interrupt subprograms.

5.3.2 STL

STL -- Program jump to secondary bus

16-bit Instruction	STL: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	STL number	S number of the STL statement to be executed	-	BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	-	-	-	S[1]

NOTE

[1]: Only the S element is supported.

5.3.3 RET

RET -- Program return to primary bus

16-bit Instruction	RET: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	-

Function and Instruction Description

STL splits the running process of a controlled device into several states or procedures, performs logical programming based on each state, and then switches between states based on the signal condition. STL programming simplifies logical design and makes commissioning and maintenance easier.

The STL instruction can be represented by a ladder chart, where the state (S) is considered as a control procedure used for the sequential programming of input conditions and output control. This type of control separates the ongoing procedure from the preceding procedure and implements device control by executing various procedures in sequence.

STL and ladder charts differ in programming.

An STL program starts with the STL instruction (which is different from S used in general ladder charts) and ends with the RET instruction. The intermediate programs are guided by the S state. The operation logic of the S state is switched to the next state when conditions are met.

5.3.4 OUTSTL

OUTSTL -- Output program jump to secondary bus

16-bit Instruction	OUTSTL: Continuous execution

32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	STL number	S number of the STL statement to be executed	-	BOOL

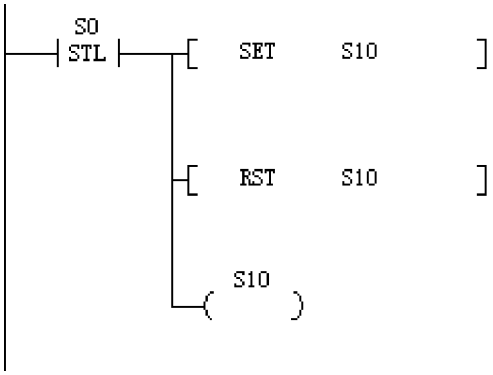
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	-	-	-	-	-	S [1]

NOTE

[1]: Only the S element is supported.

Instruction Example



5.3.5 SETSTL

SETSTL -- Setting program jump to secondary bus

16-bit Instruction	SETSTL: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	STL number	S number of the STL statement to be executed	-	BOOL

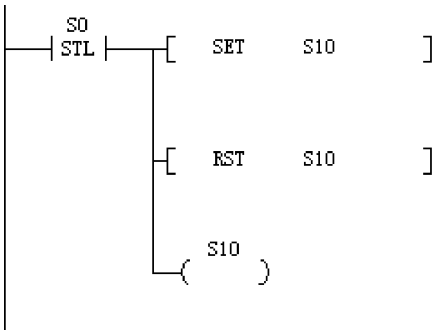
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	-	-	-	-	-	S [1]

NOTE

[1]: Only the S element is supported.

Instruction Example



5.3.6 RSTETL

RSTSTL -- Resetting program jump to secondary bus

16-bit Instruction	RSTSTL: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	STL number	S number of the STL statement to be executed	-	BOOL

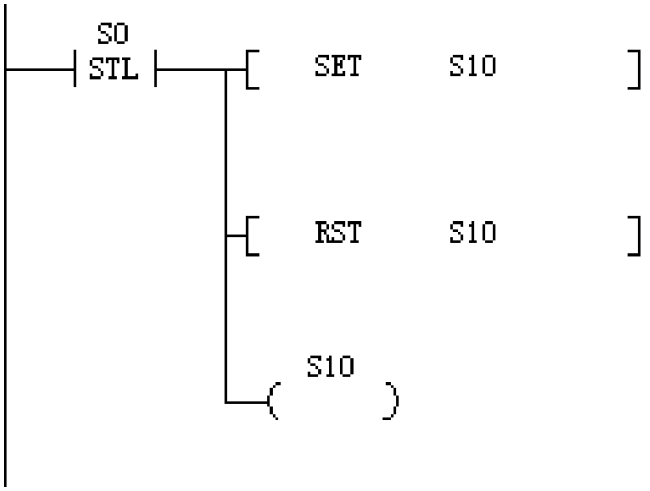
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	-	-	-	-	-	S [1]

NOTE

[1]: Only the S element is supported.

Instruction Example



5.4 Contact Operation Instructions

5.4.1 Contact Comparison

5.4.1.1 List of Instructions

The following table lists the contact comparison instructions.

Category	Instruction	Function
Contact comparison instructions	LD=	LD contact comparison equal to
	LD>	LD contact comparison greater than
	LD<	LD contact comparison less than
	LD<>	LD contact comparison not equal to
	LD>=	LD contact comparison greater than or equal to
	LD<=	LD contact comparison less than or equal to
	AND=	AND contact comparison equal to
	AND>	AND contact comparison greater than
	AND<	AND contact comparison less than
	AND<>	AND contact comparison not equal to
	AND>=	AND contact comparison greater than or equal to
	AND<=	AND contact comparison less than or equal to
	OR=	OR contact comparison equal to
	OR>	OR contact comparison greater than
	OR<	OR contact comparison less than

Category	Instruction	Function
	OR<>	OR contact comparison not equal to
	OR>=	OR contact comparison greater than or equal to
	OR<=	OR contact comparison less than or equal to
	FLDD>	Conductive when floating-point number greater than comparative state contact (S1 > S2)
	FLDD>=	Conductive when floating-point number greater than or equal to comparative state contact (S1 ≥ S2)
	FLDD<	Conductive when floating-point number less than comparative state contact (S1 < S2)
	FLDD<=	Conductive when floating-point number less than or equal to comparative state contact (S1 ≤ S2)
	FLDD=	Conductive when floating-point number equal to comparative state contact (S1 = S2)
	FLDD<>	Conductive when floating-point number not equal to comparative state contact (S1 ≠ S2)
	FANDD>	Conductive when floating-point number greater than comparative AND state contact (S1 > S2)
	FANDD>=	Conductive when floating-point number greater than or equal to comparative AND state contact (S1 ≥ S2)
	FANDD<	Conductive when floating-point number less than comparative AND state contact (S1 < S2)
	FANDD<=	Conductive when floating-point number less than or equal to comparative AND state contact (S1 ≤ S2)
	FANDD=	Conductive when floating-point number equal to comparative AND state contact (S1 = S2)
	FANDD<>	Conductive when floating-point number not equal to comparative AND state contact (S1 ≠ S2)
	FORD>	Conductive when floating-point number greater than comparative OR state contact (S1 > S2)
	FORD>=	Conductive when floating-point number greater than or equal to comparative OR state contact (S1 ≥ S2)
	FORD<	Conductive when floating-point number less than comparative OR state contact (S1 < S2)

Category	Instruction	Function
	FORD<=	Conductive when floating-point number less than or equal to comparative OR state contact (S1 ≤ S2)
	FORD=	Conductive when floating-point number equal to comparative OR state contact (S1 = S2)
	FORD<>	Conductive when floating-point number not equal to comparative OR state contact (S1 ≠ S2)
	LDZ>	Conductive when absolute value greater than comparative state contact (|S1 - S2| > |S3|)
	LDZ>=	Conductive when absolute value greater than or equal to comparative state contact (|S1 - S2| ≥ |S3|)
	LDZ<	Conductive when absolute value less than comparative state contact (|S1 - S2| < |S3|)
	LDZ<=	Conductive when absolute value less than or equal to comparative state contact (|S1 - S2| ≤ |S3|)
	LDZ=	Conductive when absolute value equal to comparative state contact (|S1 - S2| = |S3|)
	LDZ<>	Conductive when absolute value not equal to comparative state contact (|S1 - S2| ≠ |S3|)
Contact comparison instructions	ANDZ>	Conductive when absolute value greater than comparative AND state contact (|S1 - S2| > |S3|)
	ANDZ>=	Conductive when absolute value greater than or equal to comparative AND state contact (|S1 - S2| ≥ |S3|)
	ANDZ<	Conductive when absolute value less than comparative AND state contact (|S1 - S2| < |S3|)
	ANDZ<=	Conductive when absolute value less than or equal to comparative AND state contact (|S1 - S2| ≤ |S3|)
	ANDZ=	Conductive when absolute value equal to comparative AND state contact (|S1 - S2| = |S3|)
	ANDZ<>	Conductive when absolute value not equal to comparative AND state contact (|S1 - S2| ≠ |S3|)
	ORZ>	Conductive when absolute value greater than comparative OR state contact (|S1 - S2| > |S3|)
	ORZ>=	Conductive when absolute value greater than or equal to comparative OR state contact (|S1 - S2| ≥ |S3|)

Category	Instruction	Function
	ORZ<	Conductive when absolute value less than comparative OR state contact (|S1 - S2| < |S3|)
	ORZ<=	Conductive when absolute value less than or equal to comparative OR state contact (|S1 - S2| ≤ |S3|)
	ORZ=	Conductive when absolute value equal to comparative OR state contact (|S1 - S2| = |S3|)
	ORZ<>	Conductive when absolute value not equal to comparative OR state contact (|S1 - S2| ≠ |S3|)

5.4.1.2 AND#

Data comparison instructions -- The AND# instruction compares two operands and outputs the comparison result as a logical state. The variables in comparison are processed as signed numbers.

AND= -- AND contact comparison equal to
AND> -- AND contact comparison greater than
AND< -- AND contact comparison less than
AND<> -- AND contact comparison not equal to
AND>= -- AND contact comparison greater than or equal to
AND<= -- AND contact comparison less than or equal to

16-bit Instruction	AND=: Continuous execution			
32-bit Instruction	ANDD=: Continuous execution			
16-bit Instruction	AND>: Continuous execution			
32-bit Instruction	ANDD>: Continuous execution			
16-bit Instruction	AND<: Continuous execution			
32-bit Instruction	ANDD<: Continuous execution			
16-bit Instruction	AND<>: Continuous execution			
32-bit Instruction	ANDD<>: Continuous execution			
16-bit Instruction	AND<=: Continuous execution			
32-bit Instruction	ANDD<=: Continuous execution			
16-bit Instruction	AND>=: Continuous execution			
32-bit Instruction	ANDD>=: Continuous execution			
Operand	Name	Description	Range	Data Type

S1	Comparand 1	Data source to be compared or data variable unit 1	-	INT and DINT
S2	Comparand 2	Data source to be compared or data variable unit 2	-	INT and DINT

NOTE

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- The AND# instruction is preceded by other logical operations.
- This instruction compares two operands and outputs the comparison result as a logical state, which is used for a program flow operation. The variables in comparison are processed as signed numbers.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

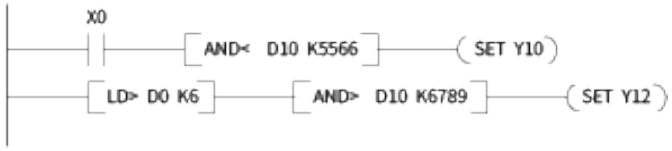
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The following table lists the AND contact comparison modes.

16-bit Instruction	FNC NO	32-bit Instruction	ON Condition	OFF Condition
AND=	232	ANDD=	S1 = S2	S1 ≠ S2
AND>	233	ANDD>	S1 > S2	S1 <= S2
AND<	234	ANDD<	S1 < S2	S1 >= S2
AND<>	236	ANDD<>	S1 <> S2	S1 = S2
AND<=	237	ANDD<=	S1 <= S2	S1 > S2
AND>=	238	ANDD>=	S1 >= S2	S1 < S2

Instruction Example



- When X0 is ON and the value of D10 is smaller than that of K5566, Y10 is ON and remains ON.
- When the value of D0 is greater than that of K6 and the value of D10 is greater than that of K6789, Y12 is ON and remains ON.

Use the 32-bit instruction ANDD# when comparing 32-bit variables. Otherwise, an error occurs.

5.4.1.3 LD#

Contact comparison instructions -- The LD# instruction compares two operands and outputs the comparison result as a logical state. The variables in comparison are processed as signed numbers.

LD= -- Contact comparison equal to
LD> -- Contact comparison greater than
LD< -- Contact comparison less than
LD<> -- Contact comparison not equal to

16-bit Instruction	LD=: Continuous execution			
32-bit Instruction	LDD=: Continuous execution			
16-bit Instruction	LD>: Continuous execution			
32-bit Instruction	LDD>: Continuous execution			
16-bit Instruction	LD<: Continuous execution			
32-bit Instruction	LDD<: Continuous execution			
16-bit Instruction	LD<>: Continuous execution			
32-bit Instruction	LDD<>: Continuous execution			
16-bit Instruction	LD<=: Continuous execution			
32-bit Instruction	LDD<=: Continuous execution			
16-bit Instruction	LD>=: Continuous execution			
32-bit Instruction	LDD>=: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Comparand 1	Data source to be compared or data variable unit 1	-	INT and DINT
S2	Comparand 2	Data source to be compared or data variable unit 2	-	INT and DINT

NOTE

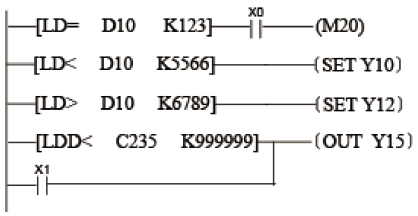
- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

Function and Instruction Description

The following table lists the LD contact comparison modes.

16-bit Instruction	FNC NO	32-bit Instruction	ON Condition	OFF Condition
LD=	224	LDD=	S1 = S2	S1 ≠ S2
LD>	225	LDD>	S1 > S2	S1 <= S2
LD<	226	LDD<	S1 < S2	S1 >= S2
LD<>	228	LDD<>	S1 ≠ S2	S1 = S2
LD<=	229	LDD<=	S1 <= S2	S1 > S2
LD>=	230	LDD>=	S1 >= S2	S1 < S2

Instruction Example



To compare 32-bit variables, use the 32-bit instruction LDD#. Otherwise, an error occur.

5.4.1.4 OR#

Data comparison instructions -- The OR# instruction compares two operands and outputs the comparison result as a logical state. The variables in comparison are processed as signed numbers.

OR= -- OR contact comparison equal to
OR> -- OR contact comparison greater than
OR< -- OR contact comparison less than
OR<> -- OR contact comparison not equal to
OR>= -- OR contact comparison greater than or equal to
OR<= -- OR contact comparison less than or equal to

16-bit Instruction	OR=: Continuous execution
32-bit Instruction	ORD=: Continuous execution

16-bit Instruction	OR>: Continuous execution
32-bit Instruction	ORD>: Continuous execution
16-bit Instruction	OR<: Continuous execution
32-bit Instruction	ORD<: Continuous execution
16-bit Instruction	OR<>: Continuous execution
32-bit Instruction	ORD<>: Continuous execution
16-bit Instruction	OR<=: Continuous execution
32-bit Instruction	ORD<=: Continuous execution
16-bit Instruction	OR>=: Continuous execution
32-bit Instruction	ORD>=: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Comparand 1	Data source to be compared or data variable unit 1	-	INT and DINT
S2	Comparand 2	Data source to be compared or data variable unit 2	-	INT and DINT

NOTE

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- This instruction compares two operands and outputs the comparison result as a logical state, which is used for a program flow operation. The variables in comparison are processed as signed numbers.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

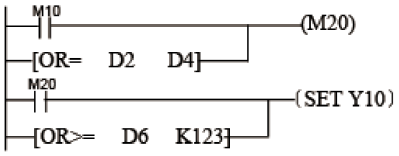
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The following table lists the OR contact comparison modes.

16-bit Instruction	FNC NO	32-bit Instruction	ON Condition	OFF Condition
OR=	240	ORD=	S1 = S2	S1 ≠ S2
OR>	241	ORD>	S1 > S2	S1 <= S2
OR<	242	ORD<	S1 < S2	S1 >= S2
OR<>	244	ORD<>	S1 <> S2	S1 = S2
OR<=	245	ORD<=	S1 <= S2	S1 > S2
OR>=	246	ORD>=	S1 >= S2	S1 < S2

Instruction Example



- When M10 is ON or the value of D2 is equal to that of D4, M20 is ON.
- When M20 is ON or the value of D6 is greater than or equal to that of K123, Y10 is ON and remains ON.

Use the 32-bit instruction ORD# when comparing 32-bit variables. Otherwise, an error occurs.

5.4.1.5 FLDD#

Floating-point contact comparison -- The FLDD# instruction compares two floating-point operands and sets a contact (a node directly connected to the left-hand bus) to ON or OFF based on the comparison result.

FLDD= -- Floating-point contact comparison equal to
FLDD> -- Floating-point contact comparison greater than
FLDD< -- Floating-point contact comparison less than
FLDD<> -- Floating-point contact comparison not equal to
FLDD>= -- Floating-point contact comparison greater than or equal to
FLDD<= -- Floating-point contact comparison less than or equal to

16-bit Instruction	-
32-bit Instruction	FLDD>: Continuous execution
16-bit Instruction	-
32-bit Instruction	FLDD>=: Continuous execution
16-bit Instruction	-
32-bit Instruction	FLDD<: Continuous execution

16-bit Instruction	-
32-bit Instruction	FLDD<=: Continuous execution
16-bit Instruction	-
32-bit Instruction	FLDD=: Continuous execution
16-bit Instruction	-
32-bit Instruction	FLDD<>: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Data 1	Soft element number of source data 1	-	REAL
S2	Data 2	Soft element number of source data 2	-	REAL

NOTE

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the FLDD*, FANDD*, and FORD* instructions, the input is FLDD*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-

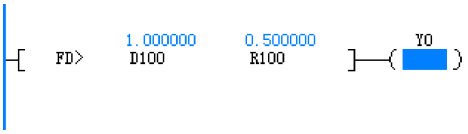
Function and Instruction Description

The FLDD# instruction compares S1 and S2. The contact becomes conductive (ON) when the conditions are met. Otherwise, it is non-conductive (OFF).

32-bit Instruction	ON Condition	OFF Condition
FLDD>	S1 > S2	S1 <= S2
FLDD>=	S1 >= S2	S1 < S2

32-bit Instruction	ON Condition	OFF Condition
FLDD<	S1 < S2	S1 >= S2
FLDD<=	S1 <= S2	S1 > S2
FLDD=	S1 = S2	S1 <> S2
FLDD<>	S1 <> S2	S1 = S2

Instruction Example



5.4.1.6 FANDD#

Floating-point AND contact comparison -- The FANDD# instruction compares two floating-point operands and sets a contact (a node connected to another node in series) to ON or OFF based on the comparison result.

FANDD= -- Floating-point AND contact comparison equal to
FANDD> -- Floating-point AND contact comparison greater than
FANDD< -- Floating-point AND contact comparison less than
FANDD<> -- Floating-point AND contact comparison not equal to
FANDD>= -- Floating-point AND contact comparison greater than or equal to
FANDD<= -- Floating-point AND contact comparison less than or equal to

16-bit Instruction	-
32-bit Instruction	FANDD>: Continuous execution
16-bit Instruction	-
32-bit Instruction	FANDD>=: Continuous execution
16-bit Instruction	-
32-bit Instruction	FANDD<: Continuous execution
16-bit Instruction	-
32-bit Instruction	FANDD<=: Continuous execution
16-bit Instruction	-
32-bit Instruction	FANDD=: Continuous execution
16-bit Instruction	-
32-bit Instruction	FANDD<>: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Data 1	Soft element number of source data 1	-	REAL
S2	Data 2	Soft element number of source data 2	-	REAL

NOTE

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the FLDD*, FANDD*, and FORD* instructions, the input is FLDD*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

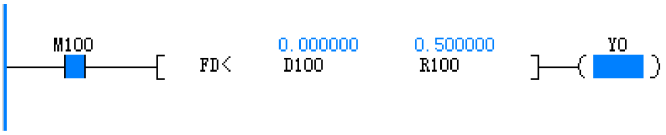
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-

Function and Instruction Description

The FANDD# instruction compares S1 and S2. The contact becomes conductive (ON) when the conditions are met. Otherwise, it is non-conductive (OFF).

32-bit Instruction	ON Condition	OFF Condition
FANDD>	S1 > S2	S1 <= S2
FANDD>=	S1 >= S2	S1 < S2
FANDD<	S1 < S2	S1 >= S2
FLD<=	S1 <= S2	S1 > S2
FANDD=	S1 = S2	S1 <> S2
FANDD<>	S1 <> S2	S1 = S2

Instruction Example



5.4.1.7 FORD#

Floating-point OR contact comparison -- The FORD# instruction compares two floating-point operands and sets a contact (a node connected to another node in parallel) to ON or OFF based on the comparison result.

FORD= -- Floating-point OR contact comparison equal to
FORD> -- Floating-point OR contact comparison greater than
FORD< -- Floating-point OR contact comparison less than
FORD<> -- Floating-point OR contact comparison not equal to
FORD>= -- Floating-point OR contact comparison greater than or equal to
FORD<= -- Floating-point OR contact comparison less than or equal to

16-bit Instruction	-
32-bit Instruction	FORD>: Continuous execution
16-bit Instruction	-
32-bit Instruction	FORD>=: Continuous execution
16-bit Instruction	-
32-bit Instruction	FORD<: Continuous execution
16-bit Instruction	-
32-bit Instruction	FORD<=: Continuous execution
16-bit Instruction	-
32-bit Instruction	FORD=: Continuous execution
16-bit Instruction	-
32-bit Instruction	FORD<>: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Data 1	Soft element number of source data 1	-	REAL
S2	Data 2	Soft element number of source data 2	-	REAL

Operand	Name	Description	Range	Data Type
S1	Data 1		-	REAL

		Soft element number of source data 1		
S2	Data 2	Soft element number of source data 2	-	REAL

NOTE

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the FLDD*, FANDD*, and FORD* instructions, the input is FLDD*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

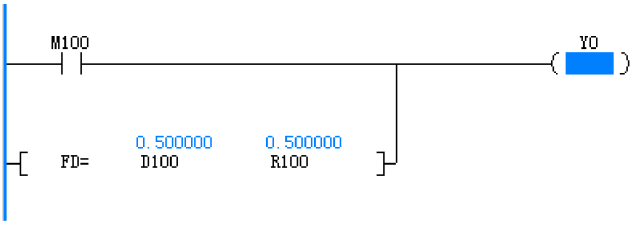
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-

Function and Instruction Description

The FORD# instruction compares S1 and S2. The contact becomes conductive (ON) when the conditions are met. Otherwise, it is non-conductive (OFF).

32-bit Instruction	ON Condition	OFF Condition
FORD>	S1 > S2	S1 <= S2
FORD>=	S1 >= S2	S1 < S2
FORD<	S1 < S2	S1 >= S2
FORD<=	S1 <= S2	S1 > S2
FORD=	S1 = S2	S1 <> S2
FORD<>	S1 <> S2	S1 = S2

Instruction Example



5.4.1.8 LDZ#

Absolute value comparison contact -- The LDZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3 and sets a contact (a node directly connected to the left-hand bus) to ON or OFF based on the comparison result.

LDZ= -- Absolute value contact comparison equal to
LDZ> -- Absolute value contact comparison greater than

16-bit Instruction	LDZ>: Continuous execution
32-bit Instruction	LDDZ>: Continuous execution
16-bit Instruction	LDZ>=: Continuous execution
32-bit Instruction	LDDZ>=: Continuous execution
16-bit Instruction	LDZ<: Continuous execution
32-bit Instruction	LDDZ<: Continuous execution
16-bit Instruction	LDZ<=: Continuous execution
32-bit Instruction	LDDZ<=: Continuous execution
16-bit Instruction	LDZ=: Continuous execution
32-bit Instruction	LDDZ=: Continuous execution
16-bit Instruction	LDZ<>: Continuous execution
32-bit Instruction	LDDZ<>: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Subtrahend	Source element of the subtrahend	-	INT and DINT
S2	Minuend	Source element of the minuend	-	INT and DINT
S3	Comparand	Source element of the comparand	-	INT and DINT

Operand	Name	Description	Range	Data Type
S1	Subtrahend		-	INT and DINT

Operand	Name	Description	Range	Data Type
		Source element of the subtrahend		
S2	Minuend	Source element of the minuend	-	INT and DINT
S3	Comparand	Source element of the comparand	-	INT and DINT

NOTE

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the LDZ*/LDDZ*, ANDZ*/ANDDZ*, and ORZ*/ORDZ* instructions, the input is LDZ*/LDDZ*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

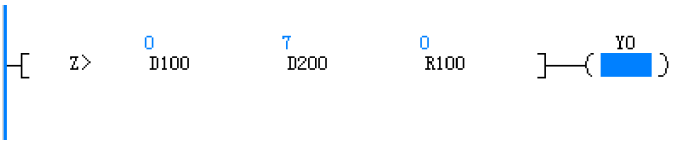
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The LDZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3. The contact becomes conductive (ON) when the conditions are met. Otherwise, it is non-conductive (OFF).

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
LDZ>	LDDZ>	|S1 - S2| > |S3|	|S1 - S2| <= |S3|
LDZ>=	LDDZ>=	|S1 - S2| >= |S3|	|S1 - S2| < |S3|
LDZ<	LDDZ<	|S1 - S2| < |S3|	|S1 - S2| >= |S3|
LDZ<=	LDDZ<=	|S1 - S2| <= |S3|	|S1 - S2| > |S3|
LDZ=	LDDZ=	|S1 - S2| = |S3|	|S1 - S2| <> |S3|
LDZ<>	LDDZ<>	|S1 - S2| <> |S3|	|S1 - S2| = |S3|

Instruction Example



5.4.1.9 ANDZ#

Absolute value comparison AND contact -- The ANDZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3 and sets a contact (a node connected to another node in series) to ON or OFF based on the comparison result.

ANDZ= -- Absolute value comparison AND contact equal to
ANDZ> -- Absolute value comparison AND contact greater than
ANDZ< -- Absolute value comparison AND contact less than
ANDZ<> -- Absolute value comparison AND contact not equal to
ANDZ>= -- Absolute value comparison AND contact greater than or equal to

16-bit Instruction	ANDZ>: Continuous execution
32-bit Instruction	ANDDZ>: Continuous execution
16-bit Instruction	ANDZ>=: Continuous execution
32-bit Instruction	ANDDZ>=: Continuous execution
16-bit Instruction	ANDZ<: Continuous execution
32-bit Instruction	ANDDZ<: Continuous execution
16-bit Instruction	ANDZ<=: Continuous execution
32-bit Instruction	ANDDZ<=: Continuous execution
16-bit Instruction	ANDZ=: Continuous execution
32-bit Instruction	ANDDZ=: Continuous execution
16-bit Instruction	ANDZ<>: Continuous execution
32-bit Instruction	ANDDZ<>: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Subtrahend	Source element of the subtrahend	-	INT and DINT
S2	Minuend	Source element of the minuend	-	INT and DINT
S3	Comparand	Source element of the comparand	-	INT and DINT

NOTE

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the LDZ*/LDDZ*, ANDZ*/ANDDZ*, and ORZ*/ORDZ* instructions, the input is LDZ*/LDDZ*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

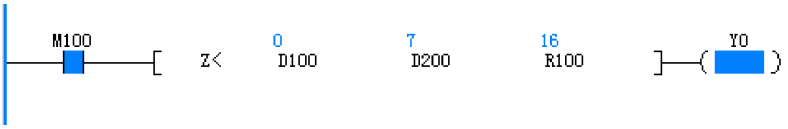
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The ANDZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3. The contact becomes conductive (ON) when the conditions are met. Otherwise, it is non-conductive (OFF).

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
ANDZ>	ANDDZ>	|S1 - S2| > |S3|	|S1 - S2| <= |S3|
ANDZ>=	ANDDZ>=	|S1 - S2| >= |S3|	|S1 - S2| < |S3|
ANDZ<	ANDDZ<	|S1 - S2| < |S3|	|S1 - S2| >= |S3|
ANDZ<=	ANDDZ<=	|S1 - S2| <= |S3|	|S1 - S2| > |S3|
ANDZ=	ANDDZ=	|S1 - S2| = |S3|	|S1 - S2| <> |S3|
ANDZ<>	ANDDZ<>	|S1 - S2| <> |S3|	|S1 - S2| = |S3|

Instruction Example



5.4.1.10 ORZ#

Absolute value comparison OR contact -- The ORZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3 and sets a contact (a node connected to another node in parallel) to ON or OFF based on the comparison result.

ORZ= -- Absolute value comparison OR contact equal to
ORZ> -- Absolute value comparison OR contact greater than
ORZ< -- Absolute value comparison OR contact less than
ORZ<> -- Absolute value comparison OR contact not equal to
ORZ>= -- Absolute value comparison OR contact greater than or equal to
ORZ<= -- Absolute value comparison OR contact less than or equal to

16-bit Instruction	ORZ>: Continuous execution
32-bit Instruction	ORDZ>: Continuous execution
16-bit Instruction	ORZ>=: Continuous execution
32-bit Instruction	ORDZ>=: Continuous execution
16-bit Instruction	ORZ<: Continuous execution
32-bit Instruction	ORDZ<: Continuous execution
16-bit Instruction	ORZ<=: Continuous execution
32-bit Instruction	ORDZ<=: Continuous execution
16-bit Instruction	ORZ=: Continuous execution
32-bit Instruction	ORDZ=: Continuous execution
16-bit Instruction	ORZ<>: Continuous execution
32-bit Instruction	ORDZ<>: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Subtrahend	Source element of the subtrahend	-	INT and DINT
S2	Minuend	Source element of the minuend	-	INT and DINT
S3	Comparand	Source element of the comparand	-	INT and DINT

NOTE

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the LDZ*/LDDZ*, ANDZ*/ANDDZ*, and ORZ*/ORDZ* instructions, the input is LDZ*/LDDZ*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

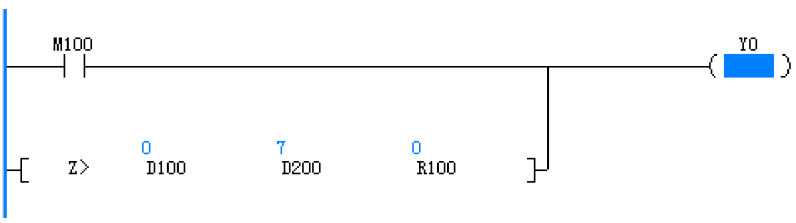
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The ORZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3. The contact becomes conductive (ON) when the conditions are met. Otherwise, it is non-conductive (OFF).

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
ORZ>	ORDZ>	|S1 - S2| > |S3|	|S1 - S2| <= |S3|
ORZ>=	ORDZ >=	|S1 - S2| >= |S3|	|S1 - S2| < |S3|
ORZ<	ORDZ<	|S1 - S2| < |S3|	|S1 - S2| >= |S3|
ORZ<=	ORDZ<=	|S1 - S2| <= |S3|	|S1 - S2| > |S3|
ORZ=	ORDZ=	|S1 - S2| = |S3|	|S1 - S2| <> |S3|
ORZ<>	ORDZ<>	|S1 - S2| <> |S3|	|S1 - S2| = |S3|

Instruction Example



5.4.2 Contact Logical Operation

5.4.2.1 List of Instructions

The following table lists the contact logical operation instructions.

Instruction category	Name	Function
Contact logical operation instructions	LD&	LD logical AND operation
	LD|	LD logical OR operation
	LD^	LD logical XOR operation
	AND&	LD logical AND operation
	AND|	AND logical OR operation
	AND^	AND logical XOR operation
	OR&	OR logical AND operation
	OR|	OR logical OR operation
	OR^	OR logical XOR operation

5.4.2.2 LD*

LD logic operation instructions -- The bit logic operation result is used to determine whether the contact (a node directly connected to the left-hand bus) is conductive.

LD| -- LD logic OR operation
LD^ -- LD logic XOR operation

16-bit Instruction	LD&: Continuous execution			
32-bit Instruction	LDD&: Continuous execution			
16-bit Instruction	LD|: Continuous execution			
32-bit Instruction	LDD|: Continuous execution			
16-bit Instruction	LD^: Continuous execution			
32-bit Instruction	LDD^: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Soft element number of source data 1	-	INT and DINT
S2	Data 2	Soft element number of source data 2	-	INT and DINT

NOTE

- * indicates &, |, or ^.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

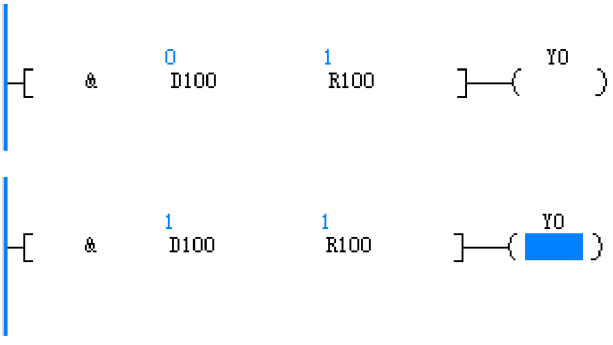
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The LD* instruction performs a logic operation (AND: &; NOT: |; XOR: ^) on S1 and S2. It is conductive (ON) if the operation result is not 0 and non-conductive (OFF) if the operation result is 0. The execution results are as follows.

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
LD&	LDD&	S1&S2 ≠ 0	S1&S2 = 0
LD|	LDD|	S1|S2 ≠ 0	S1|S2 = 0
LD^	LDD^	S1^S2 ≠ 0	S1^S2 = 0

Instruction Example



5.4.2.3 AND*

The bit logic operation result is used to determine whether the contact (a node connected to another node in series) is conductive.

AND& -- AND logic AND operation
AND| -- AND logic OR operation
AND^ -- AND logic XOR operation

16-bit Instruction	AND&: Continuous execution			
32-bit Instruction	ANDD&: Continuous execution			
16-bit Instruction	AND|: Continuous execution			
32-bit Instruction	ANDD|: Continuous execution			
16-bit Instruction	AND^: Continuous execution			
32-bit Instruction	ANDD^: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Soft element number of source data 1	-	INT and DINT
S2	Data 2	Soft element number of source data 2	-	INT and DINT



- * indicates &, |, or ^.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

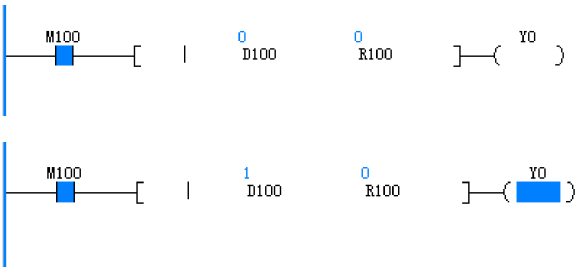
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The AND instruction performs a logic operation (AND: &; NOT: |; XOR: ^) on S1 and S2. It is conductive (ON) if the operation result is not 0 and non-conductive (OFF) if the operation result is 0. The execution results are as follows.

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
AND&	ANDD&	S1&S2 ≠ 0	S1&S2 = 0
AND|	ANDD|	S1|S2 ≠ 0	S1|S2 = 0
AND^	ANDD^	S1^S2 ≠ 0	S1^S2 = 0

Instruction Example



5.4.2.4 OR*

The bit logical operation result is used to determine whether the contact (a node connected to another node in parallel) is conductive.

OR& -- OR logic AND operation
OR| -- OR logic OR operation
OR^ -- OR logic XOR operation

16-bit Instruction	OR&: Continuous execution			
32-bit Instruction	ORD&: Continuous execution			
16-bit Instruction	OR|: Continuous execution			
32-bit Instruction	ORD|: Continuous execution			
16-bit Instruction	OR^: Continuous execution			
32-Bit Instruction	ORD^: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Soft element number of source data 1	-	INT and DINT
S2	Data 2	Soft element number of source data 2	-	INT and DINT

NOTE

- * indicates &, |, or ^.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

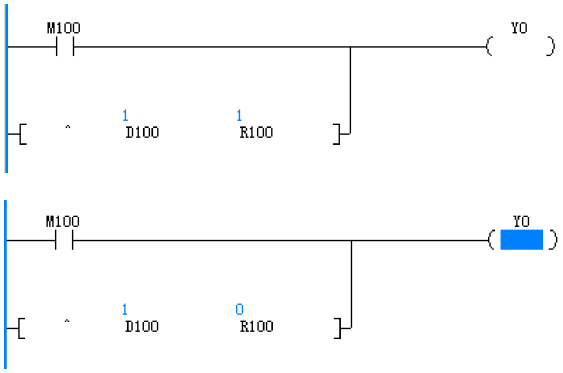
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The OR* instruction performs a logic operation (AND: &; NOT: |; XOR: ^) on S1 and S2. It is conductive (ON) if the operation result is not 0 and non-conductive (OFF) if the operation result is 0. The execution results are as follows.

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
OR&	ORD&	S1&S2 ≠ 0	S1&S2 = 0
OR|	ORD|	S1|S2 ≠ 0	S1|S2 = 0
OR^	ORD^	S1^S2 ≠ 0	S1^S2 = 0

Instruction Example



5.5 Data Operation Instructions

5.5.1 Arithmetic Operation

5.5.1.1 List of Instructions

The following table lists the arithmetic operation instructions.

Category	Instruction	Function	Language Support
Arithmetic operation instructions	ADD	Binary data addition	LD
	SUB	Binary data subtraction	LD
	MUL	Binary data multiplication	LD

Category	Instruction	Function	Language Support
	DIV	Binary data division	LD
	ABS	Absolute value	LD and LiteST
	DEABS	Absolute value of floating-point number	LD
	MOD	Remainder by binary data division	LD and LiteST
	EADD	Binary floating-point addition	LD
	ESUB	Binary floating-point subtraction	LD
	EMUL	Binary floating-point multiplication	LD
	EDIV	Binary floating-point division	LD
	INC	Binary data increment by 1	LD
	DEC	Binary data decrement by 1	LD

5.5.1.2 ADD

ADD -- Binary data addition

16-bit Instruction	ADD: Continuous execution/ADDP: Pulse execution			
32-bit instruction	DADD: Continuous execution/DADDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Augend	Data, or address of the word soft element that stores the data	-	INT, DINT, and POINTER
S2	Addend	Data, or address of the word soft element that stores the data	-	INT, DINT, and POINTER
D	Sum	Address of the word soft element that stores the data	-	INT, DINT, and POINTER

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

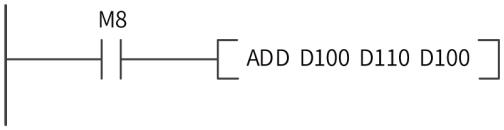
Function and Instruction Description

The ADD instruction requires contact driving and has three operands. It adds S1 and S2 algebraically in binary format, and stores the addition result in D. The variables in the algebraic operation are processed as signed numbers. The most significant bit is the sign bit. The value 0 indicates a positive number, whereas the value 1 indicates a negative number.

- The zero flag (M8020) is set if the operation result is 0.
- The carry flag (M8022) is set if the operation result is greater than 32,767 (in 16-bit operation) or 2,147,483,647 (in 32-bit operation).
- The borrow flag (M8021) is set if the operation result is less than -32,768 (in 16-bit operation) or -2,147,483,648 (in 32-bit operation).
- In 32-bit operation, the variable address in the ADD instruction is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming.

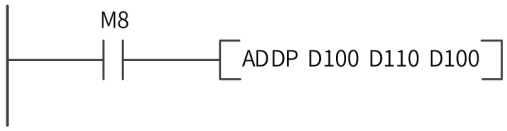
Instruction Example

Example 1



When M8 is set, D110 (addend) is added to D100 (augend), and the addition result is stored to D100. Assume that D100 is K8 and D110 is K-12, then D100 = 8 + (-12) = -4.

Example 2



D110 (addend) is added to D100 (augend) on the rising edge of M8, and the addition result is stored to D100.

5.5.1.3 SUB

SUB: Binary data subtraction

16-bit instruction	SUB: Continuous execution SUBP: Pulse execution			
32-bit instruction	DSUB: Continuous execution DSUBP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Subtrahend	Data, or address of the word soft element that stores the data	-	INT/DINT
S2	Minuend	Data, or address of the word soft element that stores the data	-	INT/DINT
D	Difference	Address of the word soft element that stores the data	-	INT/DINT

For soft elements, see the following table.

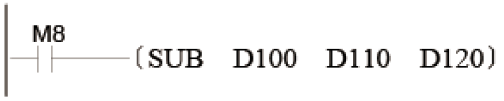
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function

The SUB instruction requires contact driving and has three operands. It performs subtraction on S1 and S2 algebraically in binary format, and stores the subtraction result in D. The variables in the algebraic operation are processed as signed numbers. The most significant bit is the sign bit. The value 0 indicates a positive number, whereas the value 1 indicates a negative number.

- The zero flag M8020 is set if the result of the calculation is 0.
- The carry flag (M802) is set if the operation result is greater than 32,767 (in 16-bit operation) or -2,147,483,647 (in 32-bit operation).
- The borrow flag (M8021) is set if the operation result is less than -32,768 (in 16-bit operation) or -2,147,483,648 (in 32-bit operation).
- In 32-bit operation, the variable addresses in the SUB instruction contain the lower 16 bits, and the adjacent high-numbered address unit contains the upper 16 bits. Avoid repeated or overlapping addresses during programming.

Example



When M8 is set, D110 (minuend) is subtracted from D100 (subtrahend), and the subtraction result is stored to D120. Assume that D100 is K10 and D110 is K8, then D120 = 10 – 8 = K2.

5.5.1.4 MUL

MUL: Binary data multiplication

16-bit instruction	MUL: Continuous execution MULP: Pulse execution			
32-bit instruction	DMUL: Continuous execution DMULP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Multiplicand	Data, or address of the word soft element that stores the data	-	INT/DINT
S2	Multiplier	Data, or address of the word soft element that stores the data	-	INT/DINT
D	Product	Address of the word soft element that stores the data	-	DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function

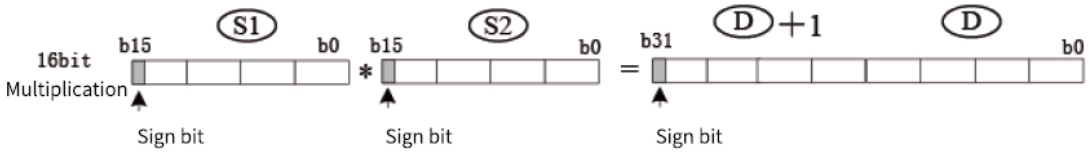
The MUL instruction requires contact driving and has three operands. It multiplies S1 by S2 algebraically in binary format, and stores the multiplication result in D. The variables in the algebraic operation are processed as signed numbers. The most significant bit is the sign bit. The value 0 indicates a positive number, whereas the value 1 indicates a negative number.

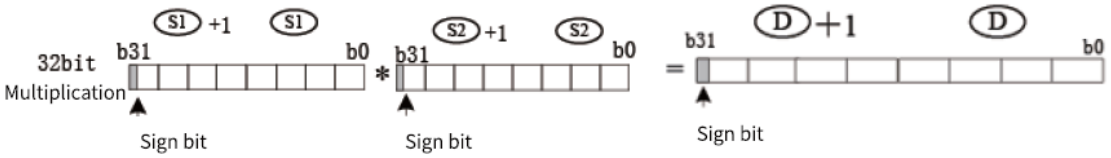
In 32-bit operation, the variable addresses in the MUL instruction contain the lower 16 bits, and the adjacent high-numbered address unit contains the upper 16 bits. Avoid repeated or overlapping addresses during

programming. The result of the calculation contains only 32 bits. If this limit is exceeded, use the floating point operation instruction EMUL.

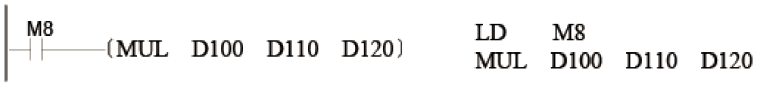
For 16-bit multiplication, the product is 32-bit data.

For 32-bit multiplication, the product is 32-bit data.





Example



When M8 is set, D100 (multiplicand) is multiplied by D110 (multiplier), and the multiplication result is stored in D120.

If D100 is K5 and D110 is K9, then D120 = 5 x 9 = K45.

If D100 is K1234 and D110 is K5678, then D120 and D121 = 1234 x 5678 = K7006652. Note that the product is greater than 16 bits at this time, and both D121 and D120 are used.

5.5.1.5 DIV

DIV: Binary data division

16-bit instruction	DIV: Continuous execution DIVP: Pulse execution			
32-bit instruction	DDIV: Continuous execution DDIVP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Dividend	Data, or address of the word soft element that stores the data	-	INT/ DINT
S2	Divider	Data, or address of the word soft element that stores the data	-	INT/ DINT
D	Quotient	Address of the word soft element that stores the data	-	INT/ DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

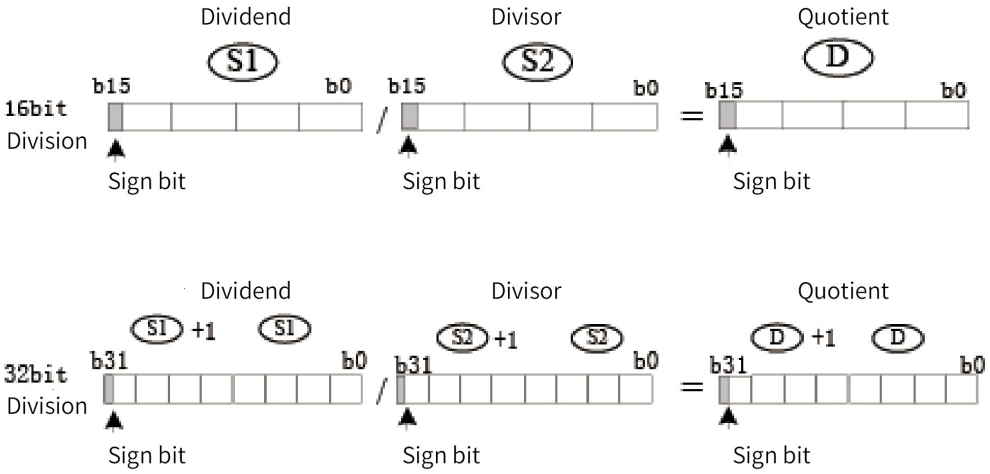
Function

The DIV instruction requires contact driving and has three operands. Use this instruction to algebraically divide the value in S1 by that in S2 in BIN mode. The result is stored in D. The variables in the algebraic operation are processed as signed numbers. The most significant bit is the sign bit. The value 0 indicates a positive number, whereas the value 1 indicates a negative number.

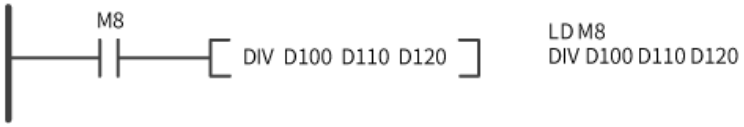
In 32-bit operation, the variable address (S1 and S2) in the DIV instruction is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming. The quotient is stored in the units pointed to by D and D+1.

The DIV operation result involves only the quotient. To obtain the remainder, use the MOD instruction.

If the divisor S2 is 0, a calculation error occurs.



Example



When M8 is set, D100 (dividend) is divided by D110 (divisor), and the quotient is stored in D120.

5.5.1.6 ABS

The LD expression of the ABS instruction is different from its LiteST expression.

- **LD**

The ABS instruction calculates the absolute value of an integer.

ABS -- Absolute value of integer

16-bit Instruction	ABS: Continuous execution/ABSP: Pulse execution			
32-bit Instruction	DABS: Continuous execution/DABSP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Source data for which the absolute value is calculated	--	INT, DINT, and POINTER
D	Absolute value	Obtained absolute value	--	INT, DINT, and POINTER

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

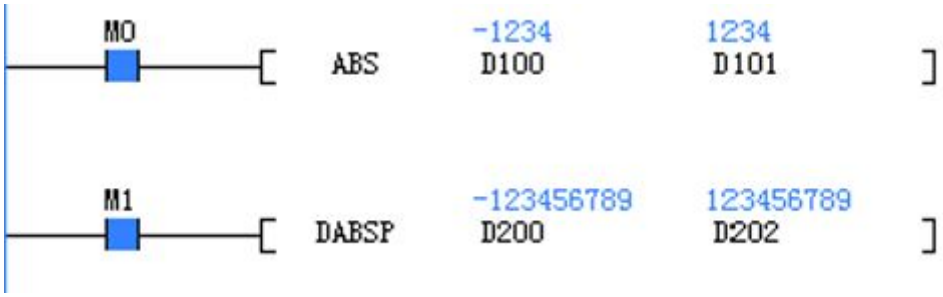
Function and Instruction Description

The ABS instruction requires contact driving and has two operands. It assigns the absolute value of the integer in S to D.

When the 32-bit instruction (DABS) is executed, the operation involves the variable units (S+1, S) and (D+1, D).

S: Integer for which the absolute value is to be calculated

D: Obtained absolute value



- **LiteST**

Return Value Type	Obtain the absolute value of a variable.

BYTE, INT, DINT, and REAL			
Operand	Description	Range	Data Type
S	Variable, of which the absolute value is obtained	-	BYTE, INT, DINT, and REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

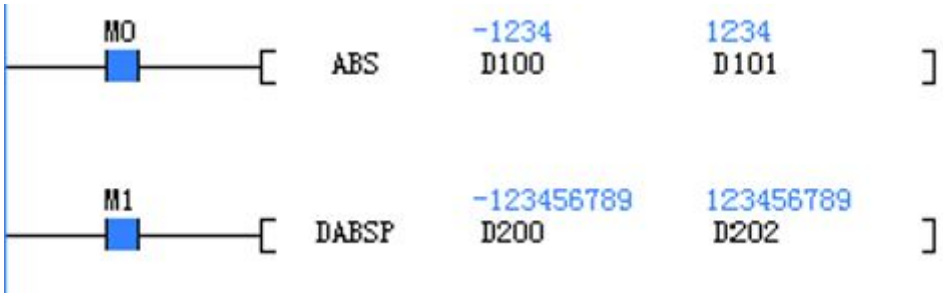
This instruction is used to obtain the absolute value of a variable, In this instruction:

- S is the variable, of which the absolute value is obtained.
- The return value is the absolute value of S.
- The return value type is consistent with the input type.

Instruction Example

```
int0 := ABS(int1);
```

Obtain the absolute value of int1 and store the result in int0.



5.5.1.7 DEABS

The EABS instruction calculates the absolute value of a floating-point number.

EABS: Absolute value of floating-point number

16-bit instruction	-			
32-bit instruction	DEABS: Continuous execution DEABSP: Pulse execution			
Operands	Name	Description	Value range	Data type

S	Source data	Source data for absolute value calculation	-	REAL
D	Absolute value	Obtained absolute value	-	REAL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The EABS instruction calculates the absolute value a single-precision floating-point number. It requires contact driving. When it is executed, the absolute value of the floating-point number in S is assigned to D.

S: Floating-point number for which the absolute value is to be calculated

D: Obtained absolute value

Example

When the flow is active, the absolute value of the source floating-point number is calculated and then assigned to the target register or variable.



5.5.1.8 MOD

The MOD instruction calculates the remainder produced by a division of two integers.

MOD -- Remainder by division

Instruction	Name	LD Expression	LiteST Expression
MOD	Remainder by division	MOD ??? ??? ???	-

16-bit Instruction	MOD: Continuous execution/MODP: Pulse execution

32-bit Instruction	DMOD: Continuous execution/DMODP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Soft element number of source data 1	-	INT, DINT, and REAL
S2	Data 2	Soft element number of source data 2	-	INT, DINT, and REAL
D	Operation result	Start number of elements for storing the operation result	-	INT, DINT, and REAL

NOTE

The data type REAL is supported by LiteST but not LD.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The MOD instruction requires contact driving and has three operands. It divides S1 by S2 algebraically in binary format and stores the remainder in D. The variables in the algebraic operation are processed as signed numbers. The most significant bit is the sign bit. The value 0 indicates a positive number, whereas the value 1 indicates a negative number.

In 32-bit operation, the variable address in the MOD instruction is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming. The remainder is stored in the units pointed to by D and D+1.

If the divisor S2 is 0, a calculation error occurs. If the dividend is negative, the remainder is negative.

Instruction Example



Program running flag

- Running: ON
- Stopped: OFF

	Element Name	Data Type	Display Format	Current Value
1	D100	INT	Decimal	10000
2	D200	INT	Decimal	9000
3	D300	INT	Decimal	1000
4				
5				

5.5.1.9 EADD

The EADD instruction adds two binary floating-point numbers together.

EADD: Floating-point addition

16-bit instruction	-			
32-bit instruction	DEADD: Continuous execution DEADDP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Augend	Augend of a binary float addition	-	REAL/POINTER
S2	Addend	Addend of a binary float addition	-	REAL/POINTER
D	AND	Unit that stores the result of binary float addition	-	REAL/POINTER

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

Two binary floating-point numbers are added together. Specifically:

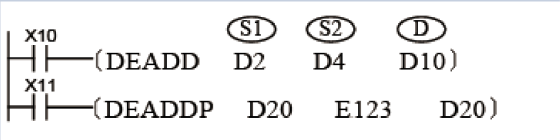
- S1 and S2 are respectively the binary floating-point augend and addend.
- D is the unit that stores the sum of S1 and S2.

The zero flag M8020 is set if the result of the calculation is 0.

The carry flag M8022 is set if the absolute value of the calculation result is greater than the maximum floating-point value.

The borrow flag M8021 is set if the absolute value of the calculation result is less than the minimum floating-point value.

Example



Introduction

When X10 is ON, two binary floating-point numbers in (D3, D2) and (D5, D4) are added together and the sum is stored in (D11, D10).

When X11 switches from OFF to ON, the floating-point number in (D21, D20) is incremented by 123.

If the sum is stored in the same unit as the augend or addend, use the DEADDP instruction of the pulse execution type. If the instruction of the continuous execution type is used, calculation is performed upon every program scan.

5.5.1.10 ESUB

The ESUB instruction performs subtraction on two binary floating-point numbers.

ESUB: Floating-point subtraction

16-bit instruction	-			
32-bit instruction	DESUB: Continuous execution DESUBP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Subtrahend	Subtrahend of a binary floating point subtraction	-	REAL
S2	Minuend	Minuend of a binary floating point subtraction	-	REAL
D	Difference	Unit that stores the difference in a binary floating point subtraction	-	REAL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S2	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

One binary floating-point number is subtracted from another. Specifically:

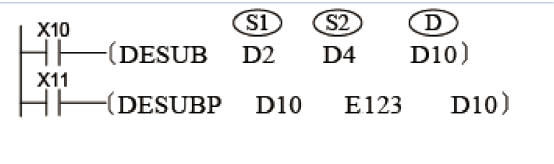
- S1 and S2 are respectively the subtrahend and minuend of a binary floating point subtraction.
- D is the unit that stores the difference in a binary floating point subtraction.

The zero flag M8020 is set if the result of the calculation is 0.

The carry flag M8022 is set if the absolute value of the calculation result is greater than the maximum floating-point value.

The borrow flag M8021 is set if the absolute value of the calculation result is less than the minimum floating-point value.

Example



Introduction

When X10 is ON, the binary floating-point number in (D5, D4) is subtracted from that in (D3, D2) and the binary floating-point difference is stored in (D11, D10).

When X11 switches from OFF to ON, the floating-point number in (D11, D10) is decremented by 123.

If the difference is stored in the same unit as the subtrahend or minuend, use the DESUBP instruction of the pulse execution type. If the instruction of the continuous execution type is used, calculation is performed upon every program scan.

5.5.1.11 EMUL

The EMUL instruction multiplies two binary floating-point numbers together.

EMUL: Binary floating point multiplication

16-bit instruction	-			
32-bit instruction	DEMUL: Continuous execution DEMULP: Pulse execution			
Operands	Name	Description		

			Value range	Data type
S1	Multiplicand	Multiplicand of a binary floating point multiplication	-	REAL
S2	Multiplier	Multiplier of a binary floating point multiplication	-	REAL
D	Product	Unit that stores the product of a binary floating point multiplication	-	REAL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function

Two binary floating-point numbers are multiplied together. Specifically:

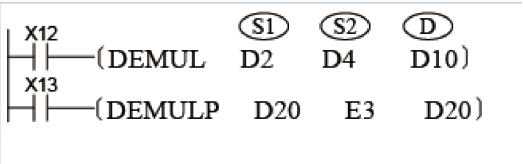
- S1 and S2 are respectively the multiplicand and multiplier of a binary floating point multiplication.
- D is the unit that stores the product of a binary floating point multiplication.

The zero flag M8020 is set if the result of the calculation is 0.

The carry flag M8022 is set if the absolute value of the calculation result is greater than the maximum floating-point value.

The borrow flag M8021 is set if the absolute value of the calculation result is less than the minimum floating-point value.

Example



Introduction

When X12 is ON, the binary floating-point number in (D3, D2) is multiplied by that in (D5, D4) and the binary floating-point product is stored in (D11, D10).

When X13 switches from OFF to ON, the binary floating-point number in (D21, D20) is multiplied by 3 and the result is stored in (D21, D20)

If the unit that stores the product is the same as the multiplicand or multiplier storage unit, use the DEMULP instruction of the pulse execution type. If the continuous execution type is used, calculation is performed upon every program scan.

5.5.1.12 EDIV

The EDIV instruction divides one binary floating-point number by another.

EDIV: Floating-point division

16-bit instruction	-			
32-bit instruction	DEDIV: Continuous execution DEDIVP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Dividend	Dividend of a binary floating point division	-	REAL
S2	Divider	Divider of a binary floating point division	-	REAL
D	Quotient	Start address of units that store the quotient of a binary floating point division	-	REAL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function

One binary floating-point number is divided by another. Specifically:

- S1 and S2 are respectively the dividend and divider of a a binary floating point division.
- D is the start address for storing the quotient of a binary floating point division.

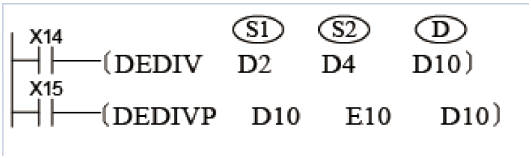
The zero flag M8020 is set if the result of the calculation is 0.

The carry flag M8022 is set if the absolute value of the calculation result is greater than the maximum floating-point value.

The borrow flag M8021 is set if the absolute value of the calculation result is less than the minimum floating-point value.

If the divisor is 0, a calculation error occurs.

Example



Introduction

When X14 = ON, the quotient of dividing the binary floating-point number in (D3, D2) by that in (D5, D4) is stored in (D11, D10).

When X15 switches from OFF to ON, the binary floating-point number in (D11, D10) is divided by 10 and the result is stored in (D11, D10).

If the unit that stores the quotient is the same as the dividend or divider storage unit, use the DEDIVP instruction of the pulse execution type. If the continuous execution type is used, calculation is performed upon every program scan.

5.5.1.13 INC

The INC instruction increments the binary data by 1.

INC -- Increment by 1

16-bit Instruction	INC: Continuous execution/INCP: Pulse execution			
32-bit Instruction	DINC: Continuous execution/DINCP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Cumulative result	Address of the word soft element that stores the data	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

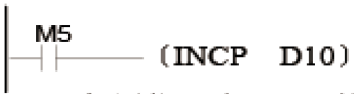
The INC instruction increases the value in D by 1 each time it is executed.

In 16-bit operation, when 32,767 increases by 1, the result is -32,768. In 32-bit operation, when 2,147,483,647 increases by 1, the result is -2,147,483,648.

This instruction does not refresh the zero flag, carry flag, and borrow flag.

In 32-bit operation, the variable address in D is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming.

Instruction Example



The value in D10 increases by 1 each time M5 is set to ON.

5.5.1.14 DEC

The DEC instruction decrements the binary data by 1.

DEC -- Decrement by 1

16-bit Instruction	DEC: Continuous execution/DECP: Pulse execution			
32-bit Instruction	DDEC: Continuous execution/DDECP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Regressive result	Address of the word element that stores the regressive result	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	√	-	-

Function and Instruction Description

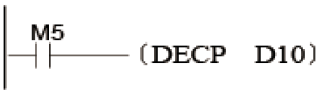
The DEC instruction decreases the value in D by 1 each time it is executed.

This instruction does not refresh the zero flag, carry flag, and borrow flag.

In 16-bit operation, when -32,768 decreases by 1, the result is 32,767. In 32-bit operation, when -2,147,483,648 decreases by 1, the result is 2,147,483,647.

In 32-bit operation, the variable address in D is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming.

Instruction Example



The value in D10 decreases by 1 each time M5 is set to ON.

5.5.2 Data Logic Operation

5.5.2.1 List of Instructions

The following table lists the data logic operation instructions.

Category	Instruction	Function
Data logic operation instruction	WAND	Binary data logic AND
	WOR	Binary data logic OR
	WXOR	Binary data logic XOR
	NEG	Binary data negation
	ENEG	Binary floating-point sign negation

5.5.2.2 WAND

When the driving conditions are met, the WAND instruction performs a logical AND on S1 and S2 by bit and stores the result in D.

WAND: Logic AND instruction

16-bit instruction	WAND: Continuous execution WANDP: Pulse execution			
32-bit instruction	DWAND: Continuous execution DWANDP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Data 1	Data in an AND operation, or address of the word soft element that stores the data	-	INT/DINT
S2	Data 2	Data in an AND operation, or address of the word soft element that stores the data	-	INT/DINT
D	Operation result	Address of the word soft element that stores the operation result	-	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

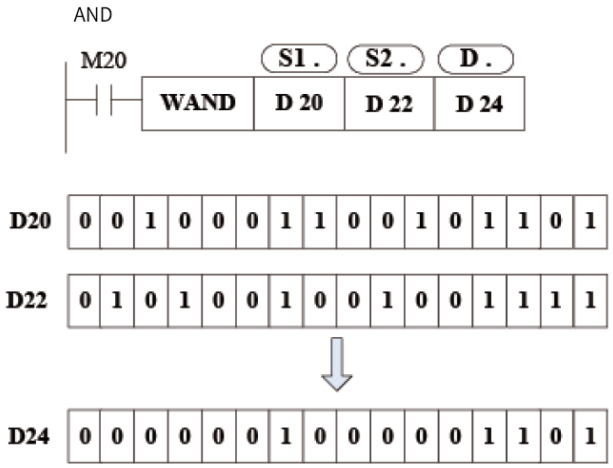
Function

The WAND instruction performs logical AND on the binary values in S1 and S2 by bit and stores the operation result in the variable D.

The result of a logical AND operation is 0 if the value of either S1 or S2 is 0.

1·1=1 1·0=0 0·1=0 0·0=0

Example



5.5.2.3 WOR

When the driving conditions are met, the WOR instruction performs a logical OR on S1 and S2 by bit and stores the result in D.

WOR: Word logic instruction

16-bit instruction	WOR: Continuous execution WORP: Pulse execution			
32-bit instruction	DWOR: Continuous execution DWORP: Pulse execution			
Operands	Name	Description	Value range	Data type

S1	Data 1	Data in the OR operation, or address of the word soft element that stores the data	-	INT/DINT
S2	Data 2	Data in the OR operation, or address of the word soft element that stores the data	-	INT/DINT
D	Operation result	Address of the word soft element that stores the operation result	-	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

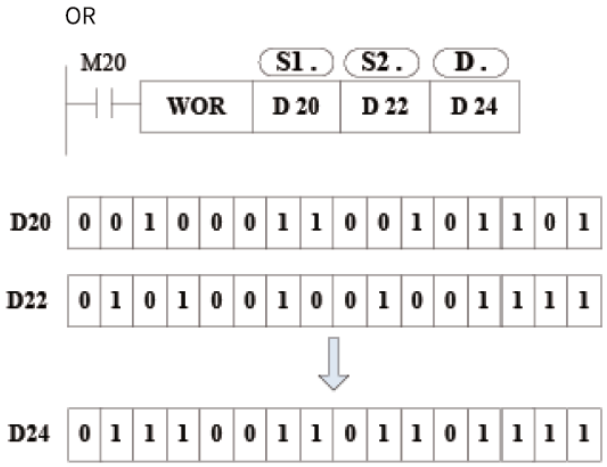
Function

The bit patterns of the BIN values in S1 and S2 are analyzed. The result of the logical OR analysis is stored in the variable D.

The result of a logical OR operation is 1 if the value of either S1 or S2 is 1.

1 + 1=1 1 + 0=1 0 + 1=1 0 + 0=0

Example



5.5.2.4 WXOR

When the driving conditions are met, the WXOR instruction performs a logical XOR operation on S1 and S2 by bit and stores the result in D.

WXOR: Logical XOR instruction

16-bit instruction	WXOR: Continuous execution WXORP: Pulse execution			
32-bit instruction	DWXOR: Continuous execution DWXORP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Data 1	Data in an XOR operation, or address of the word soft element that stores the data	-	INT/DINT
S2	Data 2	Data in an XOR operation, or address of the word soft element that stores the data	-	INT/DINT
D	Operation result	Address of the word soft element that stores the operation result	-	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

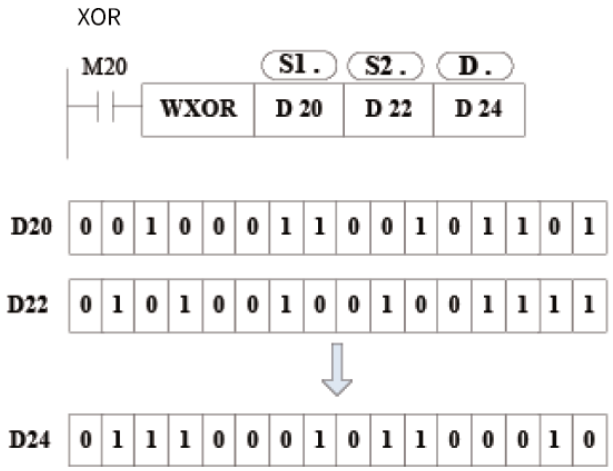
Function and Instruction Description

The bit patterns of the BIN values in S1 and S2 are analyzed. The result of the logical XOR analysis is stored in the variable D.

The result of a logical XOR operation is 0 if S1 and S2 are the same and 1 if they are different.

11=0 10=1 01=1 00=0

Example



5.5.2.5 NEG

When the driving conditions are met, the NEG instruction inverts each bit of D, adds 1, and then writes the result to D.

NEG -- Negation

16-bit Instruction	NEG: Continuous execution/NEGP: Pulse execution			
32-bit Instruction	DNEG: Continuous execution/DNEGP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Operation result	Address of the word soft element that stores the data	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

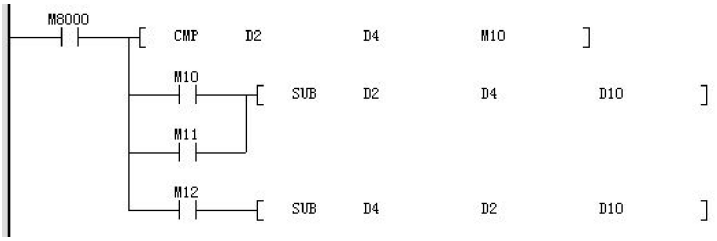
The NEG instruction requires contact driving and has one operand. It inverts each bit of D, adds 1, and then writes the result to D.

The instruction of the pulse execution type is used in normal cases.

The NEG instruction can be used to obtain the absolute value of a negative binary number.

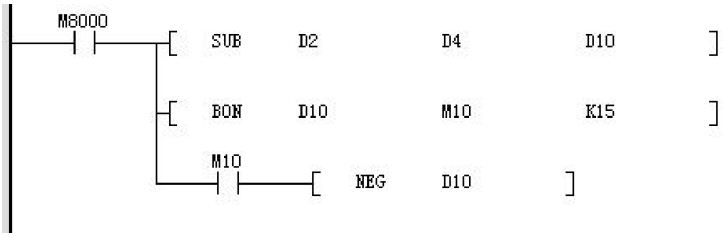
Instruction Example

The following example illustrates how to obtain the absolute value of the difference in a subtraction.



When the value of D2 is greater than that of D4, M10 is ON. When the value of D2 is equal to that of D4, M11 is ON. When the value of D2 is less than that of D4, M12 is ON. This ensures that the value in D10 is positive.

The preceding program is represented as follows.



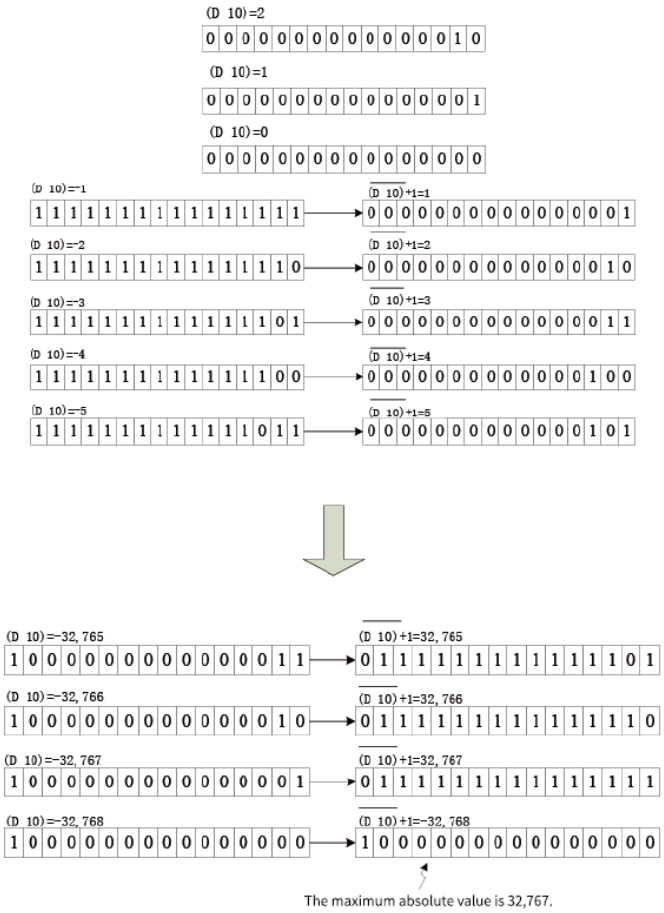
When bit 15 of D10 is 1 (indicating that the value in D10 is negative), M10 is ON. The NEG instruction can be used to obtain the absolute value of D10.

In the preceding examples, D10 = K4 if D2 = K4 and D4 = K8 or D2 = K8 and D4 = K4.

Additional Information

The most significant bit (leftmost bit) of the register indicates whether a number is positive or negative. 0 indicates positive and 1 indicates negative.

When the most significant bit is 1, the NEG instruction can be used to obtain absolute value.



5.5.2.6 ENEG

The ENEG instruction inverts the sign of a binary floating-point number (real number).

ENEG: Floating-point sign negation instruction

16-bit instruction	-			
32-bit instruction	DENEG: Continuous execution DENEGP: Pulse execution			
Operands	Name	Description	Value range	Data type
D	Operands	Start number of soft elements that store the binary floating-point number subjected to a sign change	-	REAL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-

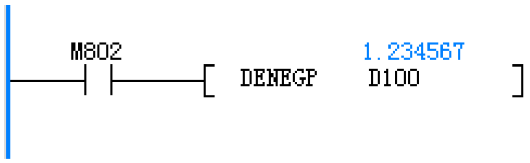
Function

The sign of the binary floating-point number in [D+1, D] is inverted. The result is stored in [D+1, D]. The pulse execution type is generally used.

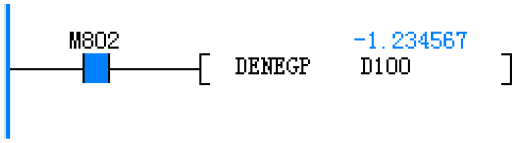
Example

Data in D100 and D101 is inverted, and the result is stored in D100 and D101.

- Before execution of the instruction



- After execution of the instruction



5.5.3 Word Bit Operation

5.5.3.1 List of Instructions

The following table lists the word bit operation instructions.

Instruction category	Name	Function
Word bit operation instruction	BLD	Word or dword bit contact instruction
	BLDI	Word or dword bit inversion contact instruction
	BAND	Word or dword bit AND contact instruction
	BANDI	Word or dword bit AND inversion contact instruction
	BOR	Word or dword bit OR contact instruction
	BORI	Word or dword bit OR inversion contact instruction
	BOUT	Word or dword bit data output instruction
	BSET	Word or dword bit data setting instruction
	BRST	Word or dword bit data reset instruction

5.5.3.2 BLD

The execution result (ON or OFF) of the BLD instruction is determined based on the state (ON or OFF) of the specified bit of the source data.

BLD: Word or dword bit contact instruction

16-bit instruction	BLD: continuous execution			
32-bit instruction	DBLD: Continuous execution			
Operands	Name	Description	Value range	Data type
S	Source data	Soft element number of the source data	-	INT/ DINT
n	Loaded bit	Designated loaded bit; value range: 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/ DINT

NOTE

For BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the software tool.

For soft elements, see the following table.

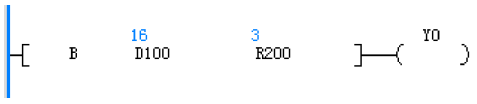
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

This instruction uses the state of the specified bit of a word variable as the contact output.

Example

- n = 3:



- n = 4:



5.5.3.3 BLDI

The execution result (ON or OFF) of the BLDI instruction is determined based on the state (ON or OFF) of the specified bit of the source data.

BLDI: Word or dword bit inversion contact instruction

16-bit instruction	BLDI: continuous execution			
32-bit instruction	DBLDI: continuous execution			
Operands	Name	Description	Value range	Data type
S	Source data	Soft element number of the source data	-	INT/ DINT
n	Loaded bit	Designated loaded bit; value range: 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/ DINT

NOTE

For BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the software tool.

For soft elements, see the following table.

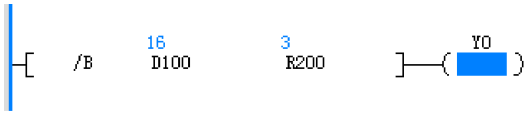
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

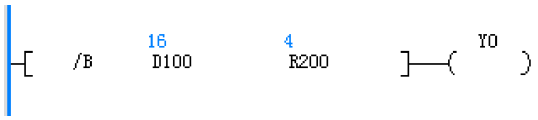
This instruction inverts the state of the specified bit of a word variable and use the inversion result as the contact output.

Example

- n = 3:



- n = 4:



5.5.3.4 BAND

The execution result (ON or OFF) of the BAND instruction is determined based on the state (ON or OFF) of the specified bit of the source data.

BAND: Word or dword bit AND contact instruction

16-bit instruction	BAND: Continuous execution			
32-bit instruction	DBAND: Continuous execution			
Operands	Name	Description	Value range	Data type
S	Source data	Soft element number of the source data	-	INT/ DINT
n	Loaded bit	Designated loaded bit; value range: 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/ DINT

NOTE

For BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the software tool.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

This instruction uses the state of the specified bit of a word variable as the contact output.

Example



5.5.3.5 BANDI

The BANDI instruction determines the node state (OFF or ON) based on the state (ON or OFF) of the specified bit of the source data.

BANDI: Word or dword data bit NAND contact

16-bit Instruction	BANDI: Continuous execution			
32-bit Instruction	DBANDI: Continuous execution			
Operand	Name	Description	Range	Data Type
S	Source data	Element number of the source data	-	INT and DINT
n	Load bit	Specified load bit, ranging from 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15 or 31	INT and DINT

NOTE

For the BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the background.

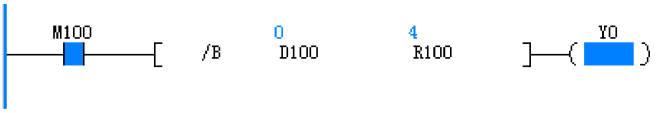
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The BANDI instruction inverts the state of the specified bit of a word variable and uses the inversion result as the contact output.

Instruction Example



5.5.3.6 BOR

The execution result (ON or OFF) of the BAND instruction is determined based on the state (ON or OFF) of the specified bit of the source data.

BOR: Word or dword bit OR contact instruction

16-bit instruction	BOR: Continuous execution			
32-bit instruction	DBOR: Continuous execution			
Operands	Name	Description	Value range	Data type
S	Source data	Soft element number of the source data	-	INT/ DINT
n	Loaded bit	Designated loaded bit; value range: 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0-15/31	INT/ DINT

NOTE

For BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the software tool.

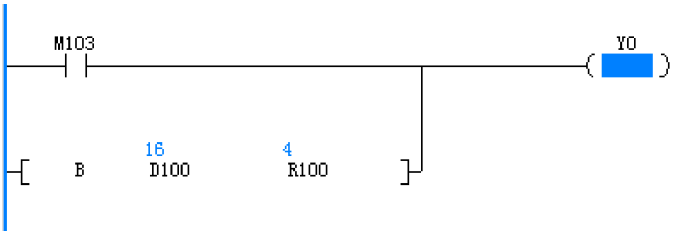
For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

This instruction uses the state of the specified bit of a word variable as the contact output.

Example



5.5.3.7 BORI

The execution result (ON or OFF) of the BORI instruction is determined based on the state (ON or OFF) of the specified bit of the source data.

BORI: Word or dword bit OR inversion contact instruction

16-bit instruction	BORI: continuous execution			
32-bit instruction	DBORI: continuous execution			
Operands	Name	Description	Value range	Data type
S	Source data	Soft component number of the source data	-	INT/ DINT
n	Loaded bit	Designated loaded bit; value range: 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0-15/31	INT/ DINT

NOTE

For BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the software tool.

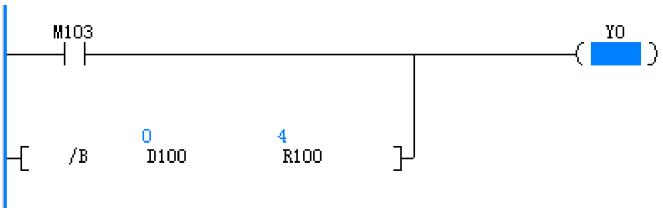
For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

This instruction inverts the state of the specified bit of a word variable and use the inversion result as the contact output.

Example



5.5.3.8 BOUT

The BOUT instruction outputs the logical operation result prior to this instruction to the designated element.

BOUT: Word or dword bit data output instruction

16-bit instruction	BOUT: continuous execution			
32-bit instruction	DBOUT: continuous execution			
Operands	Name	Description	Value range	Data type
D	Output data	Soft element number of the output data	-	INT/ DINT
n	Output bit	Designated output bit; value range: 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0-15/31	INT/ DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

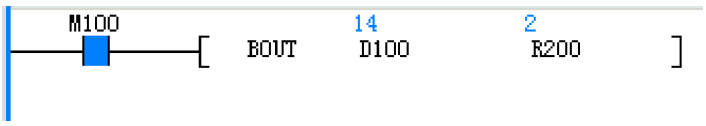
Function

The BOUT instruction outputs the flow prior to this instruction to the specified bit of a word variable.

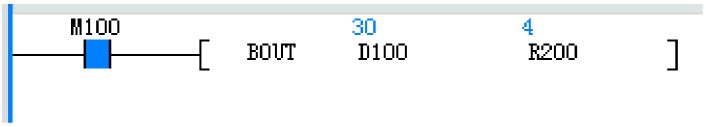
Example

The initial value of D100 is 2#1010 (decimal K10).

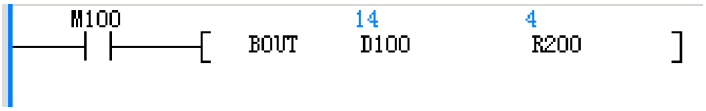
1. When R200 = 2 and M100 = ON, bit 2 of D100 is set to get the result D100 = 2#1110 (decimal K14).



2. When R200 = 4 and M100 = ON, bit 4 of D100 is set to get the result D100 = 2#11110 (decimal K30).



3. When M100 is OFF, bit 4 of D100 is reset and the value of D100 becomes 2#1110 (decimal K14).



5.5.3.9 BSET

When the BSET instruction is driven, the bit designated by this instruction is set to ON.

BSET: Word or dword bit data setting instruction

16-bit instruction	BSET: Continuous execution			
32-bit instruction	DBSET: Continuous execution			
Operands	Name	Description	Value range	Data type
D	Output data	Soft element number of the output data	-	INT/DINT
n	Output bit	Designated output bit; value range: 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0-15/31	INT/DINT

For soft elements, see the following table.

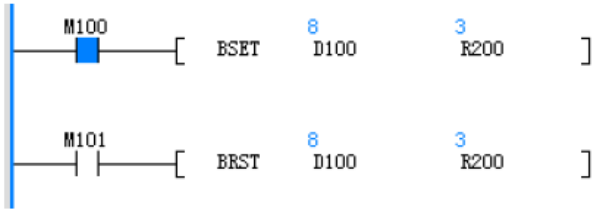
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

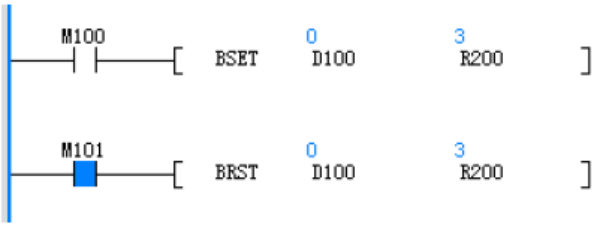
The BSET instruction sets the specified bit of a word variable to ON.

Example

- When M100 is ON:



- When M100 is OFF:



5.5.3.10 BRST

When the BRST instruction is driven, the bit designated by this instruction is set to OFF.

BRST: Word or dword bit data reset instruction

16-bit instruction	BRST: continuous execution			
32-bit instruction	DBRST: continuous execution			
Operands	Name	Description	Value range	Data type
D	Output data	Soft element number of the output data	-	INT/ DINT
n	Output bit	Designated output bit; value range: 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0-15/31	INT/ DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

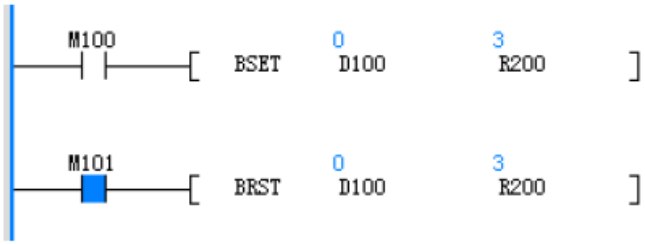
The BRST resets the specified bit of a word variable.

Example

- When M100 is OFF:



- When M101 = ON:



5.5.4 Trigonometric Function

5.5.4.1 List of Instructions

The following table lists the trigonometric function instructions.

Instruction category	Name	Function
Trigonometric functions	SIN	Floating point SIN operation
	COS	Floating point COS operation
	TAN	Floating point TAN operation
	ASIN	Binary floating point ARCSIN operation
	ACOS	Binary floating point ARCCOS operation
	ATAN	Binary floating point ARCTAN operation
	RAD	Binary floating point degree-to-radian conversion
	DEG	Binary floating point radian-to-degree conversion
	SINH	Binary floating point SINH operation
	COSH	Binary floating point COSH operation
	TANH	Binary floating point TANH operation

5.5.4.2 SIN

The LD expression of the SIN instruction is different from its LiteST expression.

- **LD**

The SIN instruction calculates the sine of the specified angle (in radians). The variable is a binary floating-point number.

SIN -- Floating-point SIN operation

16-bit Instruction	-			
32-bit Instruction	DSIN: Continuous execution/DSINP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Angle variable (binary floating-point number, in radians) for which the sine is to be calculated. The value range is from 0 to 2π.	-	REAL
D	Operation result	Memory unit for storing the sine value (binary floating-point)	-	REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The SIN instruction obtains the sine of the specified angle (in radians). The variable is a binary floating-point number. In this instruction:

- S is the angle variable (binary floating-point number) for which the sine value is to be calculated, in radians. The value range is from 0 to 2π.
- D is the memory unit for storing the SIN operation result (binary floating-point number).

Instruction Example

Example 1

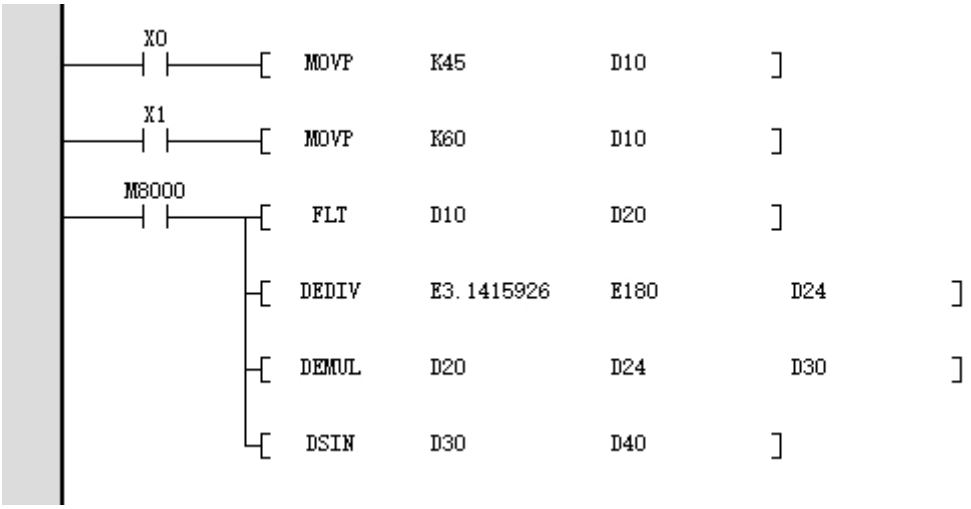


The angle in radians specified in (D21, D20) is converted into the sine value and stored in (D31, D30).

Both the source data and SIN operation result are binary floating-point numbers.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

Example 2



- 1. X0 and X1 determine the angle in degrees, which is 45° or 60°. The value is stored in D10.
- 2. The decimal value in D10 is converted into a binary floating-point number and stored in (D21, D20).
- 3. The floating-point number of (π/180) is calculated and stored in (D25, D24).
- 4. The floating-point angle in degrees in (D21, D20) is converted to the floating-point angle in radians and stored in (D31, D30).
- 5. The sine of the floating-point angle in radians (D31, D30) is calculated and stored in (D41, D40) as a floating-point number.

- **LiteST**

Return Value Type	Sine operation		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the sine value needs to be evaluated	-	BYTE, INT, DINT, and REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

The SIN instruction is used to evaluate the sine value of the specified angle (in RAD). In this instruction:

- S is the angle variable (in RAD), of which the SIN value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the SIN calculation result after conversion.

Instruction Example

```
real0 := SIN(real1);
```

Evaluate the SIN value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

5.5.4.3 TAN

The LD expression of the TAN instruction is different from its LiteST expression.

- **LD**

The TAN instruction calculates the tangent of the specified angle (in radians). The variable is a binary floating-point number.

TAN -- Floating-point TAN operation

16-bit Instruction	---			
32-bit Instruction	DTAN: Continuous execution/DTANP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Angle variable (binary floating-point number, in radians) for which the tangent is to be calculated. The value range is from 0 to 2π.	-	REAL

D	Operation result	Storage unit for storing the tangent value (binary floating-point)	-	REAL

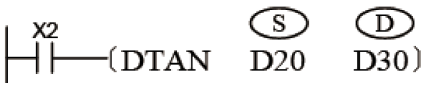
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The TAN instruction obtains the tangent of the specified angle (in radians). The variable is a binary floating-point number.

Instruction Example



The angle in radians specified in (D21, D20) is converted into the tangent value and stored in (D31, D30).

Both the source data and TAN operation result are binary floating-point numbers.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

- **LiteST**

Return Value Type	Tangent operation		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the tangent value needs to be evaluated	-	BYTE, INT, DINT, and REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to evaluate the tangent value of the specified angle (in RAD).

- S is the angle variable (in RAD), of which the tangent value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the tangent calculation result after conversion.

Instruction Example

```
real0 := TAN(real1);
```

Evaluate the tangent value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

5.5.4.4 COS

The LD expression of the COS instruction is different from its LiteST expression.

- **LD**

The COS instruction calculates the cosine of the specified angle (in radians). The variable is a binary floating-point number.

COS -- Floating-point COS operation

16-bit Instruction	-			
32-bit Instruction	DCOS: Continuous execution/DCOSP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Angle variable (binary floating-point number, in radians) for which the cosine is to be calculated; The value range is from 0 to 2π.	-	REAL
D	Operation result	Storage unit for storing the cosine	-	REAL

		value (binary floating-point)		

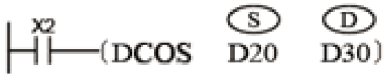
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The COS instruction obtains the cosine of the specified angle (in radians). The variable is a binary floating-point number.

Instruction Example



The angle in radians specified in (D21, D20) is converted into the cosine value and stored in (D31, D30).

Both the source data and COS operation result are binary floating-point numbers.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

- **LiteST**

Return Value Type	Cosine operation instruction		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the cosine value needs to be evaluated	-	BYTE, INT, DINT, and REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to evaluate the cosine value of the specified angle (in RAD). In this instruction:

- S is the angle variable (in RAD), of which the cosine value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the cosine calculation result after conversion.

Instruction Example

```
real0 := COS(real1);
```

Evaluate the cosine value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

5.5.4.5 ASIN

The LD expression of the ASIN instruction is different from its LiteST expression.

- **LD**

The ASIN instruction calculates the angle in radians based on a SIN value.

ASIN -- Floating-point SIN^{-1} operation

16-bit Instruction	---			
32-bit Instruction	DASIN: Continuous execution/DASINP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which ARCSIN is to be calculated	-	REAL
D	Operation result	Storage unit for storing the operation result (-n/2 to +n/2)	-	REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

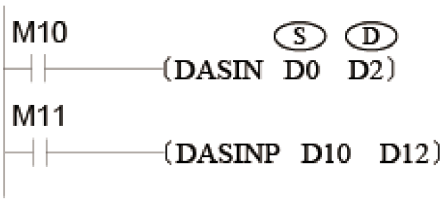
The ASIN instruction calculates the angle in radians based on a sine value.

NOTE

An operation error will occur if the value in S falls beyond the range of -1.0 to +1.0.

Instruction Example

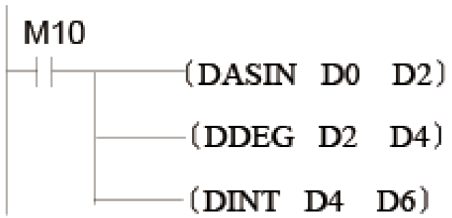
Example 1



When M10 is ON, the SIN-1 value of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

$$SIN^{-1}(D1、D0) \Longrightarrow (D3、D2)$$

Example 2



Assume that the value in (D1, D0) is 0.707106781. When M10 switches from OFF to ON, the value in (D3, D2) is 0.78539815, that in (D5, D4) is 45, and that in (D7, D6) is 45.

- **LiteST**

Return Value Type	Arcsine operation instruction		
REAL			
Operand	**Description**	**Range**	**Data Type**
S	Angle variable (in RAD), of which the arsine value needs to be evaluated	-	BYTE, INT, DINT, and REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to evaluate the arcsine value of the specified angle (in RAD), In this instruction:

- S is the angle variable (in RAD), of which the arcsine value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the arcsine calculation result after conversion.

NOTE

An operation error will occur if the value in S falls beyond the range of -1.0 to +1.0.

Instruction Example

```
real0 := ASIN(real1);
```

Evaluate the ASIN value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

5.5.4.6 ACOS

The LD expression of the ACOS instruction is different from its LiteST expression.

- **LD**

The ACOS instruction calculates the angle in radians based a COS value.

ACOS -- Floating-point COS^{-1} operation

16-bit Instruction	---

32-bit Instruction	DACOS: Continuous execution/DACOSP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which ACOS is to be calculated	-	REAL
D	Operation result	Storage unit for storing the operation result (0 to n)	-	REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

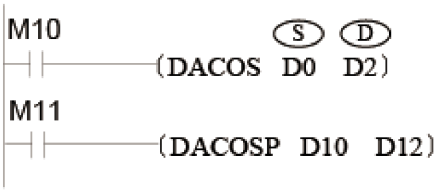
The ACOS instruction calculates the angle in radians based a COS value.

NOTE

An operation error will occur if the value in S falls beyond the range of -1.0 to +1.0.

Instruction Example

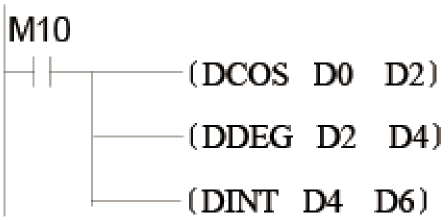
Example 1



When M10 is ON, the COS^{-1} value of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

$$COS^{-1}(D1、D0) \Longrightarrow (D3、D2)$$

Example 2



Assume that the value in (D1, D0) is 0.866025404. When M10 switches from OFF to ON, the value in (D3, D2) is 0.52359877, that in (D5, D4) is 30, and that in (D7, D6) is 30.

- **LiteST**

Return Value Type	ACOS operation		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the ACOS value needs to be evaluated	-	BYTE, INT, DINT, and REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

The ACOS instruction is used to evaluate the ACOS value of the specified angle (in RAD). In this instruction:

- S is the angle variable (in RAD), of which the ACOS value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the ACOS calculation result after conversion.

NOTE

An operation error will occur if the value in S falls beyond the range of -1.0 to +1.0.

Instruction Example

```
real0 := ACOS(real1);
```

Evaluate the ACOS value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

5.5.4.7 ATAN

The LD expression of the ATAN instruction is different from its LiteST expression.

- **LD**

The ATAN instruction calculates the angle in radians based a TAN value.

ATAN -- Floating-point TAN^{-1} operation

16-bit Instruction	---			
32-bit Instruction	DATAN: Continuous execution/DATANP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which arctangent is to be calculated	-	REAL
D	Operation result	Storage unit for storing the operation result (-n/2 to +n/2)	-	REAL

The following table lists soft elements in the instruction.

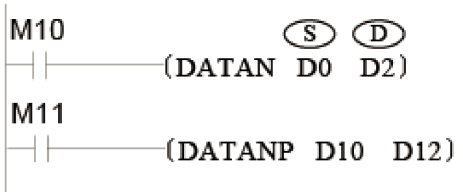
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The ATAN instruction calculates the angle in radians based a TAN value.

Instruction Example

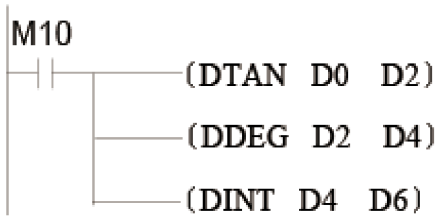
Example 1



When M10 is ON, the TAN-1 value of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

$$TAN^{-1}(D1、D0) \Longrightarrow (D3、D2)$$

Example 2



Assume that the value in (D1, D0) is 1.732050808. When M10 switches from OFF to ON, the value in (D3, D2) is 1.04719753, that in (D5, D4) is 60, and that in (D7, D6) is 60.

- **LiteST**

Return Value Type	Arctangent operation		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the arctangent value needs to be evaluated	-	BYTE, INT, DINT, and REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

The ATAN instruction is used to evaluate the arctangent value of the specified angle (in RAD). In this instruction:

- S is the angle variable (in RAD), of which the arctangent value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the arctangent calculation result after conversion.

Instruction Example

```
real0 := ATAN(real1);
```

Evaluate the arctangent value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

5.5.4.8 RAD

The RAD instruction converts a binary floating-point value in degrees into a value in radians.

RAD: Floating-point degree-to-radian conversion instruction

16-bit instruction	-			
32-bit instruction	DRAD: Continuous execution DRADP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Binary floating-point degrees to be converted to radians	-	REAL
D	Operation result	Operation result storage unit	-	REAL

The list of soft elements is shown as follows.

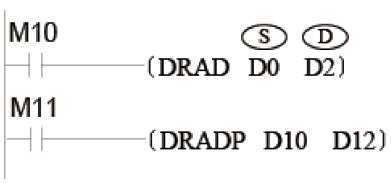
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

Binary floating-point degrees are converted to radians. The formula is as follows: Value in radians = Value in degrees x n/180.

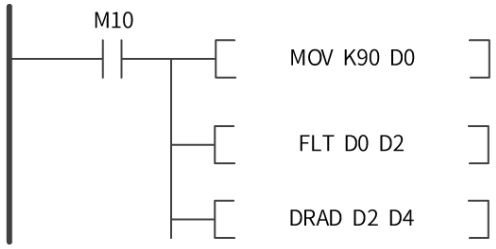
Example

Example 1



When M10 is ON, the binary floating-point value in degrees in (D1, D0) is converted into a value in radians and stored in (D3, D2).

Example 2



When M10 switches from OFF to ON, the value 90 is assigned to D0. The integer in D0 is converted into a floating-point number, which is then assigned to (D3, D2). Degree-to-radian conversion is performed on (D3, D2) and the result is assigned to (D5, D4). The final value in (D5, D4) is π/2, that is, 1.570796.

5.5.4.9 DEG

The DEG instruction converts a binary floating-point value in radians into a value in degrees.

DEG: Floating-point radian-to-degree conversion instruction

16-bit instruction	-			
32-bit instruction	DDEG: Continuous execution DDEGP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Binary floating-point radian variable to be converted to degrees	-	REAL
D	Operation result	Operation result storage unit	-	REAL

The list of soft elements is shown as follows.

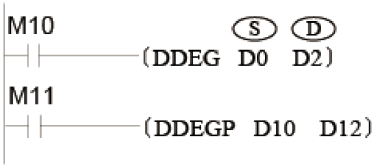
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function

Binary floating-point radians are converted to degrees. The formula is as follows: Value in degrees = Value in radians x 180/n.

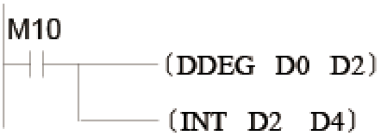
Example

Example 1



When M10 is ON, the binary floating-point value in radians in (D1, D0) is converted into a value in degrees and stored in (D3, D2).

Example 2



Assume that the value in (D1, D0) is 3.1415926. When M10 switches from OFF to ON, the value in (D3, D2) is 180. After the floating-point number is converted into an integer, the value in (D5, D4) is 180.

5.5.4.10 SINH

The SINH instruction calculates the sinh of a binary floating-point number.

SINH: Floating point SINH operation

16-bit instruction	-			
32-bit instruction	DSINH: Continuous execution DSINHP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Binary floating-point variable whose SINH is to be calculated	-	REAL
D	Operation result	Operation result storage unit (Error 6706 is returned if the operation result D exceeds the floating-point range.)	-	REAL

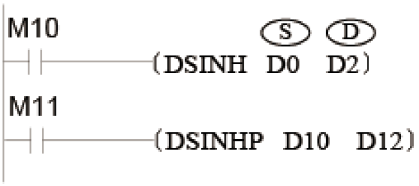
The list of soft elements is shown as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function

The sinh of a binary floating-point number is calculated. The formula is as follows: sinh = (es - e-s)/2.

Example



When M10 is ON, the hyperbolic sine of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

5.5.4.11 COSH

The COSH instruction calculates the hyperbolic cosine of a binary floating-point number.

COSH: Floating point COSH operation

16-bit instruction	-			
32-bit instruction	DCOSH: Continuous execution DCOSHP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Binary floating-point variable whose cosh is to be calculated	-	REAL
D	Operation result	Operation result storage unit	-	REAL

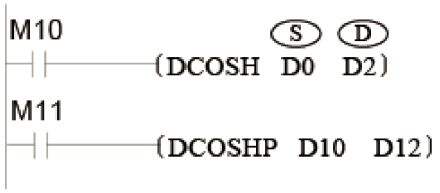
The list of soft elements is shown as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function

The cosh of a binary floating-point number is calculated. The formula is as follows: cosh = (es + e-s)/2.

Example



When M10 is ON, the hyperbolic cosine of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

5.5.4.12 TANH

The TANH instruction calculates the hyperbolic tangent of a binary floating-point number.

TANH: Floating point TANH operation

16-bit instruction	-			
32-bit instruction	DTANH: Continuous execution DTANHP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Binary floating-point variable whose tanh is to be calculated	-	REAL
D	Operation result	Operation result storage unit	-	REAL

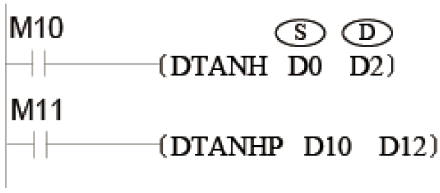
The list of soft elements is shown as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The TANH instruction calculates the hyperbolic tangent of a binary floating-point number. The formula is as follows: tanh = (es - e-s)/(es + e-s).

Example



When M10 is ON, the hyperbolic tangent of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

5.5.5 Table Operation

5.5.5.1 List of Instructions

The following table lists the table operation instructions.

Category	Instruction	Function
Table operation instruction	WSUM	Data sum calculation
	MEAN	Mean calculation
	LIMIT	Upper and lower limit control
	BZAND	Dead zone control
	ZONE	Zone control
	SCL	Coordinate determination (coordinate data of different points)
	SCL2	Coordinate determination 2 (X and Y coordinates)

5.5.5.2 WSUM

The WSUM instruction calculates the sum of consecutive 16-bit or 32-bit data entries.

WSUM: Data sum calculation

16-bit instruction	WSUM: Continuous execution WSUMP: Pulse execution			
32-bit instruction	DWSUM: Continuous execution DWSUMP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Source data	Start number of soft elements that store the data entries whose sum is to be calculated	-	INT/DINT, array*n
D	Result	Start number of soft elements that store the sum	-	INT/DINT, array*2
n	Data count	Data count	2 to 256	INT/DINT

The list of soft elements is show as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

- 16-bit instruction
  The WSUM instruction calculates the sum of n 16-bit data entries starting from [S] and stores the result as 32-bit data in [D+1, D].
- 32-bit instruction
  The WSUM instruction calculates the sum of n 32-bit data entries starting from [S+1, S] and stores the result as 64-bit data in [D+3, D+2, D+1, D].

Errors

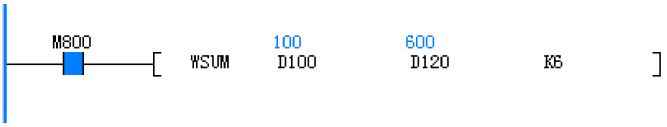
An error is returned in the following conditions:

The n elements starting from [S] are out of range.

[D] for data storage is out of range.

The operand n is less than or equal to 0.

Instruction Example



5.5.5.3 MEAN

When the driving conditions are met, the MEAN instruction calculates the mean value of n data entries starting from S and stores the result in D.

MEAN: Mean calculation

16-bit instruction	MEAN: Continuous execution MEANP: Pulse execution			
32-bit instruction	DMEAN: Continuous execution DMEANP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data start address	Start address of word soft elements that store the data entries whose	-	INT/DINT, array*n

		mean value is to be calculated		
D	Mean value	Address of the word soft element that stores the mean value	-	INT/DINT
n	Data length	Immediate operand	1 to 256	INT/DINT

The list of soft elements is shown as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The MEAN instruction calculates the mean value of n variables starting from S by dividing the sum of the variables by n and then stores the result in D.

The remainder (if any) of the calculation result is rounded.

A calculation error occurs when n falls beyond the range of 1 to 256.

Example



(D10+D11+D12+D13)/2=D20

Assume that D10 is K5, D11 is K5, D12 is K15, and D13 is K52. Then D20 is K19, and the remainder 1 is discarded.

5.5.5.4 LIMIT

The LIMIT instruction sets the upper and lower limits of an input value to control the output.

LIMIT -- Upper and lower limit control

16-bit Instruction	LIMIT: Continuous execution/LIMITP: Pulse execution			
32-bit Instruction	DLIMIT: Continuous execution/DLIMITP: Pulse execution			
Operand	Name	Description	Range	Data Type

S1	Lower limit	Minimum output limit	-	INT and DINT
S2	Upper limit	Maximum output limit	-	INT and DINT
S3	Input value	Input value to be controlled by lower and upper limits	-	INT and DINT
D	Output value	Start number of elements that store an output value under lower and upper limit control	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	-	-
S3	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

- 16-bit Instruction
  The LIMIT instruction sets the upper and lower limits in [S1] and [S2] for the input in [S3] to control the output in [D].

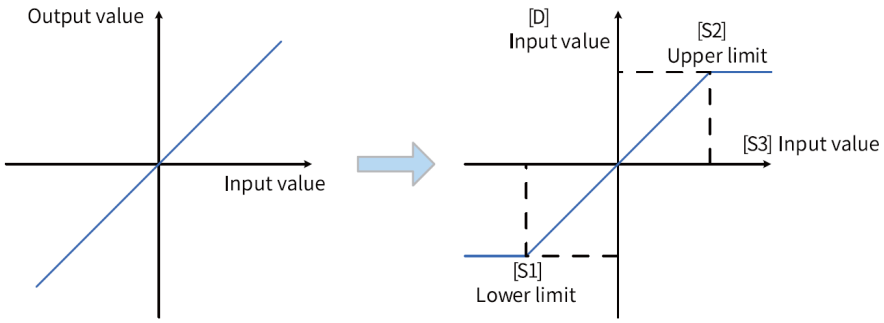
  When the input is less than the lower limit ([S1] > [S3]), [S1] is used as the output ([D]).

  When the input is greater than the upper limit ([S2] < [S3]), [S2] is used as the output ([D]).

  When the input falls between the upper and lower limits ([S1] ≤ [S3] ≤ [S2]), [S3] is used as the output ([D]).

  If only the upper limit is controlled, set the lower limit specified by [S1] to the minimum 16-bit signed value, that is, -32768.

  If only the lower limit is controlled, set the upper limit specified by [S2] to the maximum 16-bit signed value, that is, 32767.



- 32-bit Instruction
  The DLIMIT instruction sets the upper and lower limits in [S1+1, S1] and [S2+1, S2] for the input in [S3+1, S3] to control the output in [D+1, D].

  When the input is less than the lower limit ([S1+1, S1] > [S3+1, S3]), [S1+1, S1] is used as the output ([D+1, D]).

  When the input is greater than the upper limit ([S2+1, S2] < [S3+1, S3]), [S2+1, S2] is used as the output ([D+1, D]).

  When the input falls between the upper and lower limits ([S1+1, S1] ≤ [S3+1, S3] ≤ [S2+1, S2]), [S3+1, S3] is used as the output ([D+1, D]).

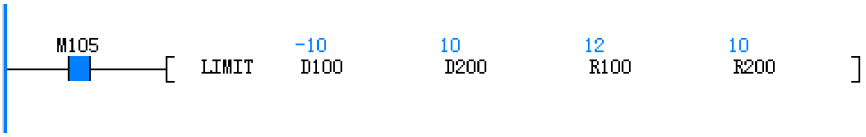
  If only the upper limit is controlled, set the lower limit specified by [S1+1, S1] to the minimum 32-bit signed value, that is, -2,147,483,648.

  If only the lower limit is controlled, set the upper limit specified by [S2+1, S2] to the maximum 32-bit signed value, that is, 2,147,483,647.

An error is returned in the following cases:

The lower limit is greater than the upper limit in the 16-bit or 32-bit instruction.

Instruction Example



5.5.5.5 BZAND

The BZAND instruction controls an output value based on whether the input value is within the specified dead zone range (defined by upper and lower limits).

BZAND -- Dead zone control

16-bit Instruction	BZAND: Continuous execution/BZANDP: Pulse execution			
32-bit Instruction	DBZAND: Continuous execution/DBZANDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Lower limit	Lower limit of a dead zone (with no output)	-	INT and DINT
S2	Upper limit		-	INT and DINT

		Upper limit of a dead zone (with no output)		
S3	Input value	Input value subject to dead zone control	-	INT and DINT
D	Output value	Number of the element that stores an output value under dead zone control	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	-	-
S3	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

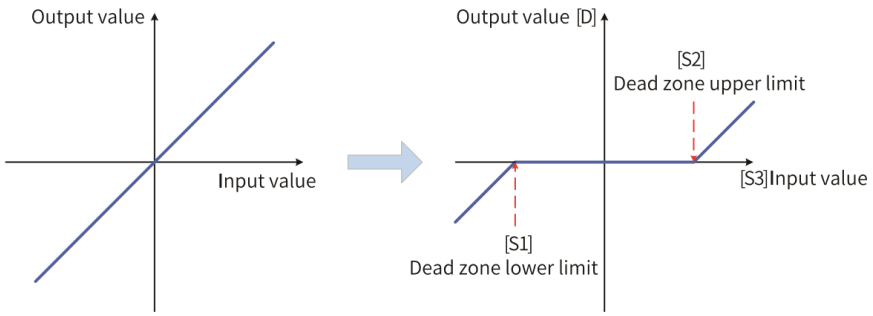
- 16-bit instruction
  The BZAND instruction sets a dead zone range in [S1] and [S2] for an input value in [S3] to control the output in [D].

  The output value is controlled as follows:

  When the input is less than the lower limit of the dead zone ([S1] > [S3]), ([S3] – [S1]) is used as the output ([D]).

  When the input is greater than the upper limit of the dead zone ([S2] < [S3]), ([S3] – [S2]) is used as the output ([D]).

  When the input falls between the upper and lower limits of the dead zone ([S1] ≤ [S3] ≤ [S2]), 0 is used as the output ([D]).



- 32-bit instruction
  The DBZAND instruction sets a dead zone range in [S1+1, S1] and [S2+1, S2] for an input value in [S3+1, S3] to control the output in [D+1, D].

  When the input is less than the lower limit of the dead zone ([S1+1, S1] > [S3+1, S3]), ([S3+1, S3] - [S1+1, S1]) is used as the output ([D+1, D]).

  When the input is greater than the upper limit of the dead zone ([S2+1, S2] < [S3+1, S3]), ([S3+1, S3] - [S2+1, S2]) is used as the output ([D+1, D]).

  When the input falls between the upper and lower limits of the dead zone ([S1+1, S1] ≤ [S3+1, S3] ≤ [S2+1, S2]), 0 is used as the output ([D+1, D]).

  Data overflow conforms to cyclical processing during instruction execution. That is, the minimum value is reached when the maximum value increases by 1. The maximum value is reached when the minimum value decreases by 1.

An error is reported if the lower limit is greater than the upper limit in the 16-bit or 32-bit instruction.

Instruction Example



5.5.5.6 ZONE

The ZONE instruction controls an output value by using the specified deviation based on whether the input value is positive or negative.

ZONE -- Zone control

16-bit Instruction	ZONE: Continuous execution/ZONEP: Pulse execution			
32-bit Instruction	DZONE: Continuous execution/DZONEP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Negative deviation	Negative deviation added to an input value. The value can be a positive number, negative number, or 0.	-	INT and DINT
S2	Positive deviation	Positive deviation added to an input value. The value can be a positive number, negative number, or 0.	-	INT and DINT
S3	Input value	Input value subject to zone control	-	INT and DINT

D	Output value	Start number of elements that store an output value under zone control	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	-	-
S3	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

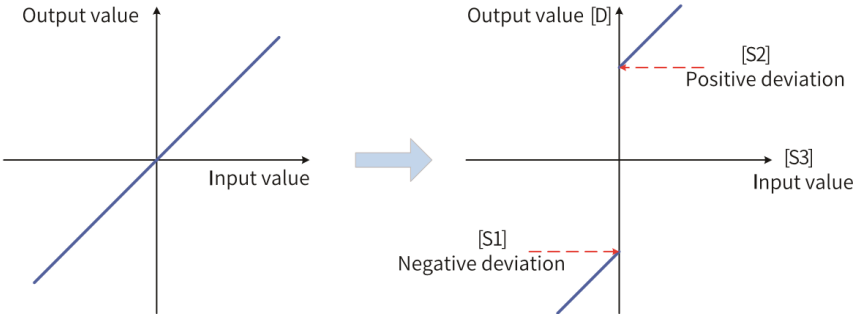
- 16-bit instruction
  The ZONE instruction adds the value in [S2] or [S1] to the input value in [S3] based on whether the input is positive or negative and stores the result in [D].

  When the input is less than 0 ([S3] < 0), ([S3] + [S1]) is used as the output ([D]).

  When the input is greater than 0 ([S3] > 0), ([S3] + [S2]) is used as the output ([D]).

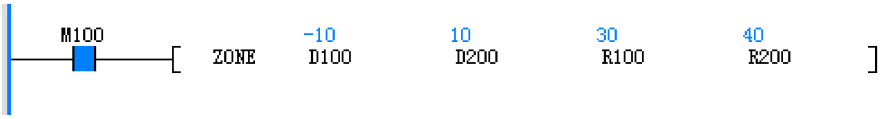
  When the input is 0 ([S3] = 0), 0 is used as the output ([D]).

  The instruction is executed as follows.



- 32-bit instruction
  The DZONE instruction adds the value in [S2+1, S2] or [S1+1, S1] to the input value in [S3+1, S3] based on whether the input is positive or negative and stores the result in [D+1, D].

Instruction Example



5.5.5.7 SCL

The SCL2 instruction determines the coordinates of an input value based on the designated data table and outputs the resulting value.

SCL: Coordinate determination (coordinates of different points)

16-bit instruction	SCL: Continuous execution SCLP: Pulse execution			
32-bit instruction	DSCL: Continuous execution DSCLP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Input value	Input value subjected to coordinate determination, or number of the soft element that stores the input value	-	INT/DINT
S2	Table data	Start number of soft elements that store the conversion table used for coordinate determination	1 to 256	INT/DINT, array*indeterminate
D	Output value	Number of the soft element that stores the output value under coordinate control	-	INT/DINT

The following table shows the soft elements.

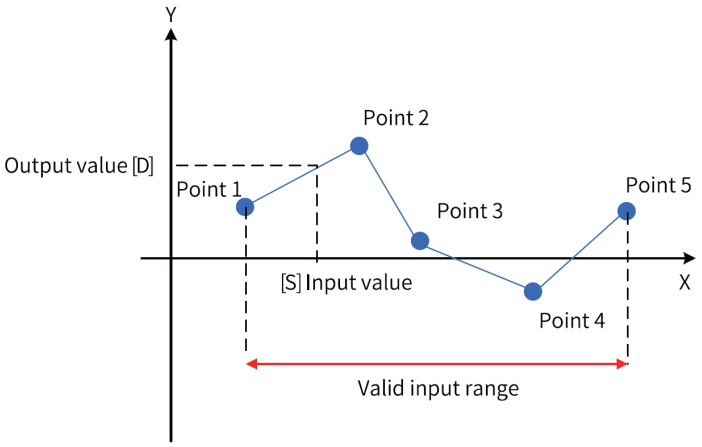
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function

- 16-bit Instruction

The SCL2 instruction determines the output value ([D]) corresponding to the input value in [S1] based on the graph determined by the table data in [S2]. If the output value is not an integer, the digit in the first decimal place is rounded.

For instruction implementation, see the following figure.



The [S2] data organizational form in the 16-bit instruction is as follows:

Setting		Soft Component Assignment for the Data Table
Assume that the number of coordinate points is 5.		[S2]
Point 1	X coordinate	[S2+1]
	Y coordinate	[S2+2]
Point 2	X coordinate	[S2+3]
	Y coordinate	[S2+4]
Point 3	X coordinate	[S2+5]
	Y coordinate	[S2+6]
Point 4	X coordinate	[S2+7]
	Y coordinate	[S2+8]
Point 5	X coordinate	[S2+9]
	Y coordinate	[S2+10]

- 32-bit instruction
  The SCL2 instruction determines the output value ([D+1, D]) corresponding to the input value in [S1+1, S1] based on the graph determined by the table data in [S2+1, S2]. If the output value is not an integer, the digit in the first decimal place is rounded.

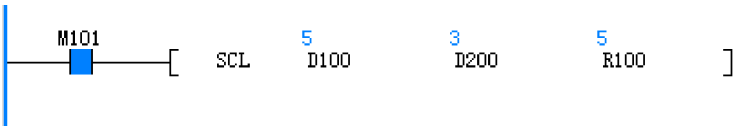
  The [S2] data organizational form in the 32-bit instruction is as follows:

Setting		Soft component assignment for the data table
Assume that the number of coordinate points is 5.		[S2+1, S]
Point 1	X coordinate	[S2+3, S2+2]
	Y coordinate	[S2+5, S2+4]
Point 2	X coordinate	[S2+7, S2+6]
	Y coordinate	[S2+9, S2+8]
Point 3	X coordinate	[S2+11, S2+10]
	Y coordinate	[S2+13, S2+12]
Point 4	X coordinate	[S2+15, S2+14]
	Y coordinate	[S2+17, S2+16]
Point 5	X coordinate	[S2+19, S2+18]
	Y coordinate	[S2+21, S2+20]

An error is returned in the following conditions:

- The x coordinates of table data are not sorted in ascending order.
- The value in [S1] is beyond the range of the table data.

Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	5
D200	16-bit INT	Decimal	3
D201	16-bit INT	Decimal	0
D202	16-bit INT	Decimal	0
D203	16-bit INT	Decimal	10
D204	16-bit INT	Decimal	10
D205	16-bit INT	Decimal	20
D206	16-bit INT	Decimal	0
	16-bit INT	Decimal	
R100	16-bit INT	Decimal	5
	16-bit INT	Decimal	

5.5.5.8 SCL2

The SCL2 instruction determines the coordinates of an input value based on the specified data table and outputs the result.

SCL2: Coordinate determination 2 (X and Y coordinates)

16-bit instruction	SCL2: Continuous execution/SCL2P: Pulse execution			
32-bit instruction	DSCL2: Continuous execution/DSCL2P: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Input value	Input value subjected to coordinate determination, or number of the soft element that stores the input value	-	INT/DINT
S2	Table data	Start number of soft elements that store the conversion table used for coordinate determination	1 to 256	INT/DINT, array*indeterminate
D	Output value	Number of the soft element that stores the output value under coordinate control	-	INT/DINT

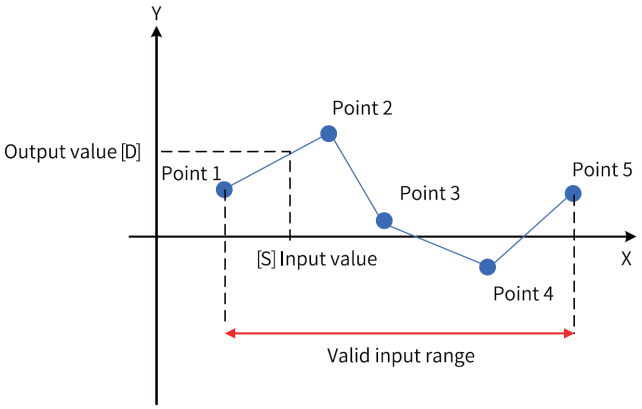
The list of soft elements is show as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function

- 16-bit Instruction
  The SCL2 instruction determines the output value ([D]) corresponding to the input value in [S1] based on the graph determined by the table data in [S2]. If the output value is not an integer, the digit in the first decimal place is rounded.

  The instruction is executed as follows:



The [S2] data organizational form in the 16-bit instruction is as follows:

Setting		Soft Component Assignment for the Data Table
Assume that the number of coordinate points is 5.		[S2]
X coordinate	Point 1	[S2+1]
	Point 2	[S2+2]
	Point 3	[S2+3]
	Point 4	[S2+4]
	Point 5	[S2+5]
Y coordinate	Point 1	[S2+6]
	Point 2	[S2+7]
	Point 3	[S2+8]
	Point 4	[S2+9]
	Point 5	[S2+10]

- 32-bit instruction
  The output value in [D+1, D] that corresponds to the input value in [S1+1, S1] is identified based on the graph determined by the table in [S2+1, S2]. If the output value is not an integer, the first digit of the fractional part is rounded.

  The [S2+1, S2] data organizational form in the 32-bit instruction is as follows:

Setting		Soft component assignment for the data table
Assume that the number of coordinate points is 5.		[S2+1, S2]
X coordinate	Point 1	[S2+3, S2+2]

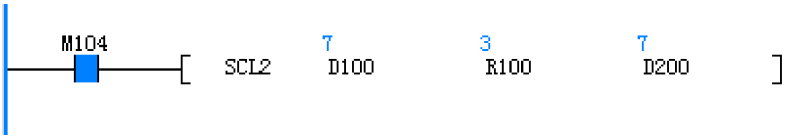
Setting		Soft component assignment for the data table
	Point 2	[S2+5, S2+4]
	Point 3	[S2+7, S2+6]
	Point 4	[S2+9, S2+8]
	Point 5	[S2+11, S2+10]
Y coordinate	Point 1	[S2+13, S2+12]
	Point 2	[S2+15, S2+14]
	Point 3	[S2+17, S2+16]
	Point 4	[S2+19, S2+18]
	Point 5	[S2+21, S2+20]

An error is returned in the following conditions:

The x coordinates of table data are not sorted in ascending order.

The value in [S1] is beyond the range of the table data.

Example



5.5.6 Exponent Operation

5.5.6.1 List of Instructions

The following table lists the exponent operation instructions.

Category	Instruction	Function	Language Support
Exponent operation instructions	EXP	Binary floating-point exponentiation operation	LD
	LOGE	Binary floating-point natural logarithm operation	LD
	LOG	Binary floating-point logarithm operation with a base of 10	LD and LiteST

Category	Instruction	Function	Language Support
	ESQR	Binary floating-point square root operation	LD
	SQR	Binary data square root operation	LD
	POW	Floating-point weight	LD
	LN	Base e (2.71828) logarithm	LiteST
	EXPT	Power operation	LiteST
	SQRT	Square root operation	LiteST

5.5.6.2 EXP

The EXP instruction performs exponentiation of a binary floating-point number with the base of mathematical constant e (2.71828).

EXP: Floating-point exponentiation operation

16-bit instruction	-			
32-bit instruction	DEXP: Continuous execution DEXPP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Binary floating-point variable used as the exponent	-	REAL
D	Operation result	Unit that stores the result of exponentiation	-	REAL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

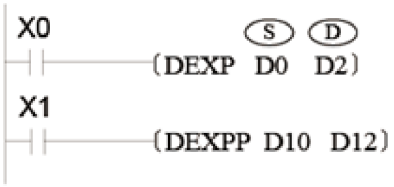
Function and Instruction Description

The EXP instruction performs exponentiation of a binary floating-point number with the base of mathematical constant e (2.71828). Specifically:

- S is the binary floating-point variable used as the exponent.
- D is the unit that stores the result of exponentiation.

> **NOTE**
>
> An operation error will occur when the operation result does not satisfy the following condition: $2^{-126} \leq$ Operation result $< 2^{128}$.

Example



When X0 is ON, exponentiation is performed for the binary floating-point value in (D1, D0) with the base of e, and the result is stored in (D3, D2). e(D1,D0)→（D3，D2）

5.5.6.3 LOG

The LD expression of the LOG instruction is different from its LiteST expression.

- **LD**

The LOG instruction calculates the common logarithm of a binary floating-point number with base 10.

LOG -- Floating-point common logarithm operation

16-bit Instruction	---			
32-bit Instruction	DLOG: Continuous execution/DLOGP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which common logarithm is to be calculated	-	REAL
D	Operation result	Unit that stores the operation result	-	REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

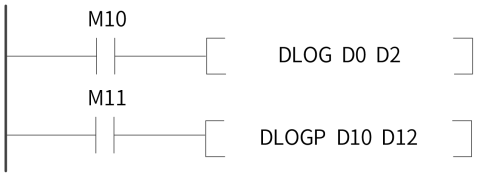
The LOG instruction calculates the common logarithm of a binary floating-point number with base 10. In this instruction:

- S is the binary floating-point variable for which common logarithm is to be calculated.
- D is the unit that stores the natural logarithm result.

NOTE

The value in S must be positive. If it is 0 or negative, an operation error will occur.

Instruction Example



When M10 is ON, the common logarithm for the binary floating-point value in (D1, D0) is calculated with base 10 and the operation result is stored in (D3, D2).

Log10 (D1, D0) → (D3, D2)

- **LiteST**

Return Value Type	Base-10 logarithm		
REAL			
Operand	Description	Range	Data Type
S	Variable to be logarithmic	-	BYTE, INT, DINT, and REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

The LOG instruction is used to perform the common logarithm operation on data with the base 10. In this instruction:

- S is the variable to be logarithmic.
- The return value is the result of the logarithm operation.

NOTE

The value in S must be positive. If it is 0 or negative, an operation error will occur.

Instruction Example

```
real0 := LOG(real1);
```

Perform the base-10 logarithm operation on real1 and store the result in real0.

5.5.6.4 LOGE

The LOGE instruction calculates the natural logarithm of a binary floating-point number with the base of mathematical constant e (2.71828).

LOGE: Floating-point natural logarithm operation

16-bit instruction	-			
32-bit instruction	DLOGE: Continuous execution DLOGEP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Binary floating-point variable whose natural logarithm is to be calculated	-	REAL
D	Operation result	Unit that stores the calculated natural logarithm	-	REAL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

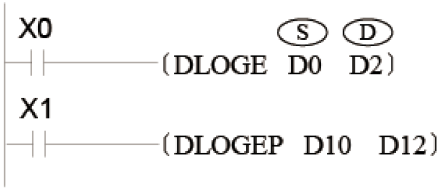
Function

The LOGE instruction calculates the natural logarithm of a binary floating-point number with the base of mathematical constant e (2.71828). Specifically:

- S is the binary floating-point variable for which natural logarithm is to be calculated.
- D is the unit that stores the natural logarithm result.

> **NOTE**
>
> The value in S must be positive. If it is 0 or negative, an operation error will occur.

Example



When X0 is ON, the natural logarithm for the binary floating-point value in (D1, D0) is calculated with the base of mathematical constant e and the operation result is stored in (D3, D2).

$$\log_e(D1、D0) \Longrightarrow (D3、D2)$$

The formula for converting the natural logarithm to common logarithm is as follows (0.4342945 used for common logarithm division):

$$10^{x} = e^{\frac{x}{0.4342945}}$$

5.5.6.5 ESQR

The ESQR instruction calculates the square root of a binary floating-point number.

ESQR: Floating-point square root operation

16-bit instruction	-			
32-bit instruction	DESQR: Continuous execution DESQRP: Pulse execution			
Operands	Name	Description	Value range	Data type

S	Data source	Binary floating-point variable whose square root is to be calculated	-	REAL
D	Operation result	Unit that stores the calculated square root	-	REAL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

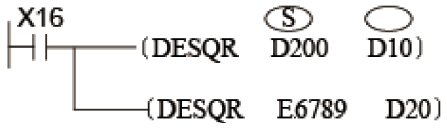
Function

The square root of a binary floating-point number is calculated.

The zero flag M8020 is set if the result of the calculation is 0.

The value in S must be positive. If it is negative, a calculation error occurs.

Example



The square root result of the binary floating-point number $\sqrt{(D201,D200)}$ is stored in (D11, D10).

The square root of the binary floating-point number E6789 is calculated, and the result is stored in (D21, D20).

5.5.6.6 SQR

The SQR instruction calculates the square root of an integer.

SQR: Square root operation

16-bit instruction	SQR: Continuous execution SQRP: Pulse execution			
32-bit instruction	DSQR: Continuous execution DSQRP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Data of which the square root is to be calculated, or	-	INT/DINT

		address of the word element that stores the data		
D	Operation result	Address of the word soft element that stores the calculated square root	-	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

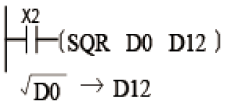
Function

The square root of the BIN value in S is calculated. The result is stored in D.

The operation result D must be an integer. The borrow flag M8021 is set to ON when the decimal places (if any) of the operation result are discarded.

The zero flag M8020 is set to ON when the operation result is 0.

Example



If D0 is K100, D12 is K10 when X2 is set to ON..

If D0 is K110, D12 is K10 (the decimal places are discarded) when X2 is set to ON.

5.5.6.7 POW

The POW instruction performs a mathematical operation in which the binary floating-point number in [S1+1, S1] is raised to the power in [S2+1, S2] and stores the result in [D+1, D].

POW: Floating-point weight instruction

16-bit instruction	-			
32-bit instruction	DPOW: Continuous execution DPOWP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Base		-	REAL

		Start address of elements that store the base, which cannot be 0		
S2	Power	Start address of soft elements that store the power	-	REAL
D	Result	Start address of soft elements that store the operation result	-	REAL

The list of soft elements is shown as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function

Since the POW instruction uses only floating-point numbers, the values in [S1] and [S2] must be converted to floating-point numbers.

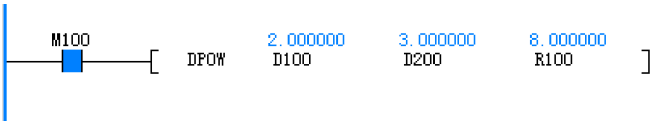
1. The carry flag M8022 is set to ON if the absolute value of the operation result is greater than the maximum floating-point value.

2. The borrow flag M8021 is set to ON if the absolute value of the operation result is less than the minimum floating-point value.

3. The zero flag M8020 is set to ON if the operation result is 0.

Example

If [S1] is 2 and [S2] is 3, then [D] = 23 = 8.



5.5.6.8 LN

Return value type	Base e (2.71828) logarithm		
REAL			
Operands	Description	Value range	Data type
S	Variable to be logarithmic	-	BYTE/INT/DINT/REAL

See the following table for soft elements.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function

This instruction is used to perform the natural logarithm operation on data with the base of e (2.71828), Specifically:

- S is the variable to be logarithmic.
- The return value is the result of the logarithm operation.

> **NOTE**
>
> The value in S must be positive. If it is 0 or negative, an operation error will occur.

Example

```
real0 := LN(real1);
```

Perform the base-e logarithm operation on real1 and store the result in real0.

5.5.6.9 EXPT

Return value type	Power operation instruction		
REAL			
Operands	**Description**	**Value range**	**Data type**
S1	Data to be powered	-	BYTE/INT/REAL
S2	Power	-	BYTE/INT/DINT/REAL

See the following table for soft elements.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S1	-	-	-	√	√	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	√	√	√	-	-

Function

This instruction is used to evaluate S1 to the power S2, Specifically:

- S1 is the data to be powered.
- S2 is the power.
- The return value is the calculation result of the power operation.

Example

```
real0 := EXPT(real1, real2);
```

Evaluate real1 to the power real2 and store the result in real0.

5.5.6.10 SQRT

Return value type	Square root operation instruction		
REAL			
Operands	Description	Value range	Data type
S	Data to be square rooted	-	BYTE/INT/DINT/REAL

See the following table for soft elements.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to take the square root of S, Specifically:

- S is the data to be square rooted.
- The return value is the calculation result of the square root operation.

> **NOTE**
>
> Note: The value in S must be positive. If it is negative, an operation error will occur.

Example

```
real0 := SQRT(real1);
```

Take the square root of real1 and store the result in real0.

5.6 Bit Operator Instructions

5.6.1 List of Instructions

The following table lists the bit operator instructions.

Category	Name	Description	Language Support
Bit operator instructions	AND	AND operation	LiteST
	OR	OR operation	LiteST

5.6.2 AND

Return Value Type	AND operation		
BYTE/INT/DINT/BOOL			
Operand	Description	Range	Data Type
S1	AND operation variable 1	-	BYTE/INT/DINT/BOOL
S2	AND operation variable 2	-	BYTE/INT/DINT/BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	√	√	√	√	√	-	√	√	√	-	√	-
S2	√	√	√	√	√	-	√	√	√	-	√	-

Function and Instruction Description

The AND instruction is used to perform an AND operation on two variables and return the result of the AND operation, In this instruction:

- S1 is operation variable 1 and expressed as an integer.
- S2 is operation variable 2 and expressed as an integer.
- The return value type is consistent with the input type.

NOTE

- The data types of S1 and S2 must be consistent.
- S1 and S2 can only be a single variable, the result of a bit operation, a binary constant, an octal constant, or a hexadecimal constant.
- AND is a bit operation instruction, and the result it returns can only be used with bit operation results, binary constants, octal constants, or hexadecimal constants in an operation.
- The data type of the return value of the AND operation is the same as that of the input value.
- When both S1 and S2 are either constant 0 or constant 1, the return value is of type INT.
- After the returned result is assigned to a variable, the variable can be used for the operations allowed by the variable.

Instruction Example

```
int0 := int1 AND int2;
```

Perform an AND operation on int1 and int2 and store the result in int0.

5.6.3 OR

Return Value Type	OR operation		
BYTE/INT/DINT/BOOL			
Operand	Description	Range	Data Type
S1	OR operation variable 1	-	BYTE/INT/DINT/BOOL

S2	OR operation variable 2	-	BYTE/INT/DINT/BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	√	√	√	√	√	-	√	√	√	-	√	-
S2	√	√	√	√	√	-	√	√	√	-	√	-

Function and Instruction Description

The OR instruction is used to perform an OR operation on two variables and return the result of the OR operation. In this instruction:

- S1 is operation variable 1 and expressed as an integer.
- S2 is operation variable 2 and expressed as an integer.
- The return value type is consistent with the input type.

NOTE

- The data types of S1 and S2 must be consistent.
- S1 and S2 can only be a single variable, the result of a bit operation, a binary constant, an octal constant, or a hexadecimal constant.
- OR is a bit operation instruction, and the result it returns can only be used with bit operation results, binary constants, octal constants, or hexadecimal constants in an operation.
- The data type of the return value of the OR operation is the same as that of the input value.
- When both S1 and S2 are either constant 0 or constant 1, the return value is of type INT.
- After the returned result is assigned to a variable, the variable can be used for the operations allowed by the variable.

Instruction Example

```
int0 := int1 OR int2;
```

Perform an OR operation on int1 and int2 and store the result in int0.

5.7 Data Processing Instructions

5.7.1 Data Conversion

5.7.1.1 List of Instructions

The following table lists the data conversion instructions.

Category	Instruction	Function	Language Support
Data conversion instructions	INT	Conversion from binary floating-point number into binary integer	LD
	BCD	Conversion from binary into BCD	LD
	BIN	Conversion from BCD into binary	LD
	FLT	Conversion from binary into binary floating-point	LD
	EBCD	Conversion from binary floating-point into decimal floating-point	LD
	EBIN	Conversion from decimal floating-point into binary floating-point	LD
	DABIN	Conversion from decimal ASCII into binary	LD
	BINDA	Conversion from binary into decimal ASCII	LD
	WTOB	Conversion from word to byte	LD
	BITW	Conversion from bit to word	LD and LiteST
	BTOW	Conversion from byte to word	LD
	WBIT	Conversion from word to bit	LD and LiteST
	WTODW	Conversion from word to dword	LD
	DWTOW	Conversion from dword to word	LD

Category	Instruction	Function	Language Support
	MCPY	Data copy (memory copy and type conversion)	LD and LiteST
	MSET	Data setting (memory setting and reset)	LD and LiteST
	UNI	4-bit combination of 16-bit data	LD
	DIS	4-bit separation of 16-bit data	LD
	ASCI	Conversion from hexadecimal into ASCII	LD
	HEX	Conversion from ASCII into hexadecimal	LD
	DECO	Data decoding	LD
	ENCO	Data encoding	LD
	SCALE_X	Scaling	LD
	NORM_X	Normalization	LD
	ROUND	Round up	LD
	TRUNC	Truncation	LD
	CELL	Floating-point number celling	LD
	FLOOR	Floating-point number flooring	LD
	AryByteTo	Combination of byte array members and saving to variables	LiteST
	AryToString	Conversion from byte array to string	LiteST
	BOOL_TO_<TYPE>	Conversion from BOOL to <TYPE>	LiteST
	DINT_TO_<TYPE>	Conversion from DINT to <TYPE>	LiteST

Category	Instruction	Function	Language Support
	INT_TO_<TYPE>	Conversion from INT to <TYPE>	LiteST
	REAL_TO_<TYPE>	Conversion from REAL to <TYPE>	LiteST
	BYTE_TO_<TYPE>	Conversion from byte to <TYPE>	LiteST
	STRING_TO_<TYPE>	Conversion from string to <TYPE>	LiteST
	StringToAry	Conversion from string to byte array	LiteST
	TO_<TYPE>	Conversion from variable to <TYPE>	LiteST
	ToAryByte	Variable separation by 1 byte and saving to byte array	LiteST

5.7.1.2 INT

The INT instruction rounds a binary floating-point number by discarding the decimal places and stores the result in D.

INT: Conversion from floating-point number to binary integer

16-bit instruction	INT: Continuous execution INTP: Pulse execution			
32-bit instruction	DINT: Continuous execution DINTP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Binary floating-point variable to be rounded	-	REAL, fixed to 32 bits
D	Operation result	Unit that stores the resulting BIN integer	-	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√[1]	-
D	-	-	-	√	√	√	-	-	-

NOTE

[1]: The 16-bit instruction does not support constant E.

Function

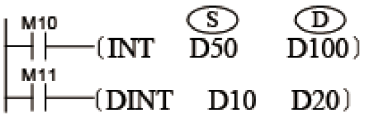
Binary floating-point numbers are rounded to remove the fractional part. The result is stored in D.

- M8020 is set when S = 0.
- The borrow flag M8021 is set when the absolute value of S is less than or equal to 1 (|S| ≤ 1).

The carry flag M8022 is set if the operation result falls beyond the following range (which results in an overflow):

- 16-bit instruction: -32,768 to +32,767
- 32-bit instruction: -2,147,483,648 to 2,147,483,647

Example



- The floating-point number in (D51, D50) is rounded and then stored in D100.
- The floating-point number in (D11, D10) is rounded and then stored in (D21, D20).

CAUTION

Note the difference in storing the operation result between the INT and DINT instructions.

5.7.1.3 BCD

The BCD instruction converts binary data into binary coded decimal (BCD) data.

BCD: Conversion from binary into BCD

16-bit instruction	BCD: Continuous execution BCDP: Pulse execution

32-bit instruction	DBCD: Continuous execution DBCDP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Binary data, or address of the word soft element that stores the binary data	-	INT/ DINT
D	Conversion result	Address of the word soft element that stores the BCD equivalent of a binary number	-	INT/ DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function

The BCD instruction requires contact driving and has two operands. It converts the binary number in S to a BCD number and stores the result in D. The BCD instruction is generally used for data format processing before data is displayed.

- For the 16-bit instruction, the conversion result ranges from 0 to 9999. An error occurs when the conversion result exceeds 9999.
- For the 32-bit instruction, the conversion result ranges from 0 to 99,999,999. An error occurs when the conversion result exceeds 99,999,999.

Example

The binary number in D200 is converted into a BCD value, which is then stored in D300.

If the value in D200 is H000E (hexadecimal) or K14 (decimal), the conversion result in D300 is 10100 (binary number)..

5.7.1.4 BIN

The BIN instruction converts BCD data into binary data.

BIN -- Conversion from BCD into binary

16-bit Instruction	BIN: Continuous execution/BINP: Pulse execution			
32-bit Instruction	DBIN: Continuous execution/DBINP: Pulse execution			
Operand	Name	Description	Range	Data Type

S	Data source	BCD data, or address of the word soft element that stores the data	-	INT and DINT
D	Conversion result	Address of the word element that stores the conversion result	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

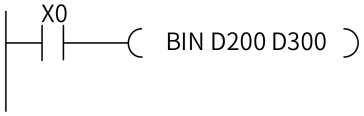
Function and Instruction Description

The BIN instruction requires contact driving and has two operands. It converts the BCD value in S to a binary number and stores the result in D. This instruction is generally used to convert the data (for example, encoder disk setting) read by external ports into binary data that can be directly used in operation.

The BCD value in S ranges from 0 to 9999 for the 16-bit instruction or from 0 to 99999999 for the 32-bit instruction.

If the data in S is not in BCD format (Hex indicates any digit beyond the range of 0 to 9), an operation error occurs.

Instruction Example



The BCD value in D200 is converted into a binary value, which is then stored in D300.

5.7.1.5 FLT

The FLT instruction converts a binary integer into a binary floating-point number.

FLT -- Conversion from binary integer to binary floating-point

16-bit Instruction	FLT: Continuous execution/FLTP: Pulse execution
32-bit Instruction	DFLT: Continuous execution/DFLTP: Pulse execution

Operand	Name	Description	Range	Data Type
S	Integer	Binary integer to be converted, or address of the word element that stores the binary integer	-	INT and DINT
D	Floating-point number	Address of the word element that stores the floating-point number after conversion	-	REAL, fixed to 32 bits

The following table lists soft elements in the instruction.

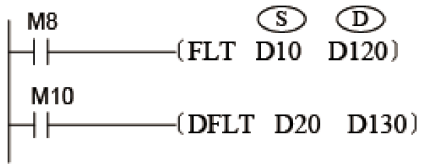
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The FLT instruction converts the integer in S into a floating-point number and stores the result in D and D+1.

This instruction implements the inverse function of the INT instruction (converting a binary floating-point number into a binary integer).

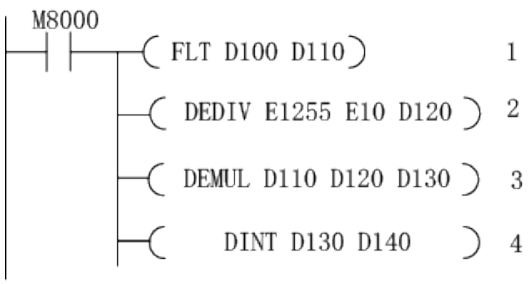
Instruction Example 1



- When M8 is ON, the 16-bit binary integer in D10 is converted into a binary floating-point number. The result is stored in (D121, D120).
- When M10 is ON, the 32-bit binary integer in (D21, D20) is converted into a binary floating-point number. The result is stored in (D131, D130).

Instruction Example 2

Instructions are executed to multiply D100 by 125.5 and convert the calculation result into an integer.



- 1. Convert the value in D100 into a floating-point number and store the result in D110.

- 2. Store the conversion result of 125.5 in D120.
- 3. Multiply the value in D110 by that in D120 and store the result in D130.
- 4. Convert the calculation result into an integer and store it in D140.

5.7.1.6 EBCD

EBCD: Conversion from binary floating-point to decimal floating-point

16-bit instruction	-			
32-bit instruction	DEBCD: Continuous execution DEBCDP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Binary floating-point variable	-	REAL
D	Operation result	Unit that stores the decimal equivalent of a binary floating-point number	-	DINT

The list of soft elements is shown below.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-

Function

Binary floating-point numbers are converted to decimal equivalents.

Example



The binary floating-point number in (D3, D2) is converted into a decimal floating-point number, which is then stored in (D11, D10).

For the binary floating-point number in (D3, D2), the real number occupies 23 bits, the exponent occupies eight bits, and the sign occupies one bit. For the decimal floating-point number in (D11, D10), the exponent (D3) and real number (D2) are expressed as D2 x 10D3 in scientific notation.

In floating-point operations of the PLC, all data is handled in binary floating-point format. Binary floating-point numbers are converted into decimal equivalents for easy monitoring.

5.7.1.7 EBIN

EBIN: Conversion from decimal floating-point to binary floating-point

16-bit instruction	-			
32-bit instruction	DEBIN: Continuous execution DEBINP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Decimal floating-point variable	-32768 to +32767	DINT
D	Result	Unit that stores the binary floating-point number after conversion	-	REAL

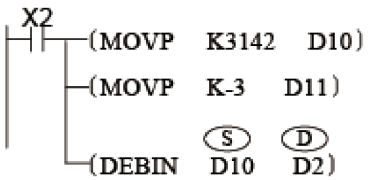
For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

Decimal floating-point numbers are converted to binary equivalents.

Example



The decimal floating-point number 3.142 in (D11, D10) is converted into a binary floating-point number, which is then stored in (D3, D2).

5.7.1.8 DABIN

The DABIN instruction converts numeric data expressed in ASCII codes (30H to 39H) into binary data.

Conversion from decimal ASCII format to BIN format

16-bit instruction	DABIN: Continuous execution DABINP: Pulse execution
	DDABIN: Continuous execution/DDABINP: Pulse execution

32-bit instruction				
Operands	Name	Description	Value range	Data type
S	Input value	Start number of soft elements that store the ASCII-encoded decimal number to be converted to a BIN number	-	INT/DINT, array*3
D	Output value	Number of the soft element that stores the conversion result	-	INT/DINT

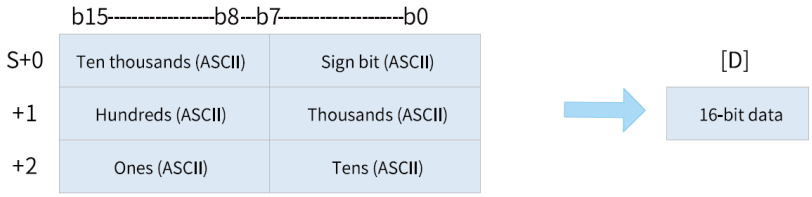
The list of soft elements is shown as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	√	-	-
D	-	-	-	√	√	-	-	-	-

Function

16-bit instruction

The ASCII-encoded decimal number (30H to 39H) stored in [S] to [S+2] is converted to a 16-bit BIN number. The result is stored in [D].



The value stored in [S] to [S+2] ranges from -32,768 to +32,767.

When the number to be converted is positive, the sign (lowest byte) is set to 20H (space); when it is negative, the sign is set to 2DH (-).

The ASCII code of every digit is in the range 30H to 39H.

When the ASCII code of each digit is 20H (space) or 00H (NULL), it is handled as 30H.

32-bit instruction

The ASCII-encoded decimal number (30H to 39H) stored in [S] to [S+5] is converted to a 32-bit BIN number. The result is stored in [D+1, D].

	b15----------b8--b7----------b0	
S+0	Billions (ASCII)	Sign bit (ASCII)
+1	Ten millions (ASCII)	Hundred millions (ASCII)
+2	Hundred thousands (ASCII)	Millions (ASCII)
+3	Thousands (ASCII)	Ten thousands (ASCII)
+4	Tens (ASCII)	Hundreds (ASCII)
+5	Ignored	Ones (ASCII)



[D+1]	[D]
High-order 16 bits	Low-order 16 bits

The value stored in [S] to [S+5] ranges from -2,147,483,648 to +2,147,483,647. The high-order byte in [S+5] is ignored.

When the number to be converted is positive, the sign (lowest byte) is set to 20H (space); when it is negative, the sign is set to 2DH (-).

The ASCII code of every digit is in the range 30H to 39H.

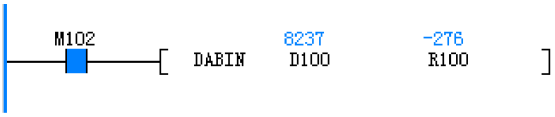
When the ASCII code of each digit is 20H (space) or 00H (NULL), it is handled as 30H.

Errors

An operation error occurs in the following conditions.

- The sign data is not 20H (space) or 2DH (-).
- The ASCII code of each digit is not 20H (space), 00H (NULL), or a value between 30H and 39H.
- The value to be converted is beyond the value range of the 16-bit or 32-bit signed number.
- The element [S+2] (16-bit operation) or [S+5] (32-bit operation) is out of range.

Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x202D
D101	16-bit INT	Hex	0x3220
D102	16-bit INT	Hex	0x3637
	16-bit INT	Hex	
R100	16-bit INT	Hex	-276

5.7.1.9 BINDA

The BINDA instruction converts BIN numbers to ASCII-encoded decimal numbers (30H to 39H).

BINDA: Conversion from BIN to decimal ASCII

16-bit instruction	BINDA: Continuous execution BINDAP: Pulse execution			
32-bit instruction	DBINDA: Continuous execution/DBINDAP: Pulse execution			
Operands	Name	Description	Value range	Data type

S	Input value	Number of the soft element that stores the BIN number to be converted to an ASCII-encoded decimal number	-	INT/DINT, array*4
D	Output value	Number of the soft element that stores the conversion result	-	INT/DINT

See the following table for the soft elements.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	√	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

16-bit Instruction

The BINDA instruction converts each digit of the 16-bit binary data in [S] into an ASCII code (30H to 39H) in decimal format and stores the result in elements starting from [D].



The 16-bit data in [S] ranges from -32768 to +32767.

The operation result is as follows:

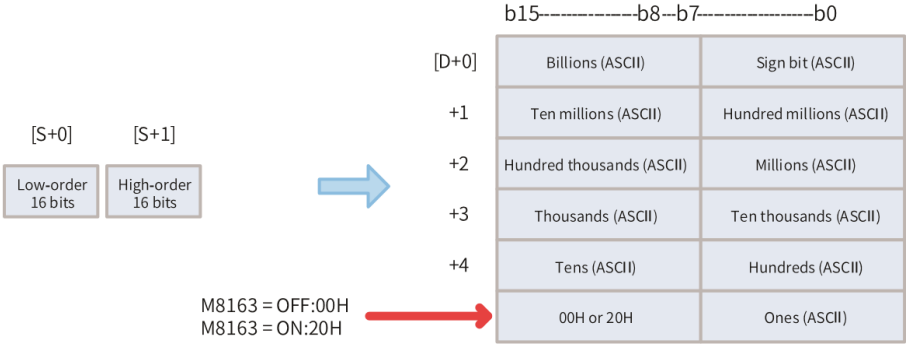
When the 16-bit number is positive, the sign bit is set to 20H (space). When it is negative, the sign bit is set to 2DH (minus sign).

When 0 exists on the left of valid digits, the sign bit is set to 20H (space).

The value in [D+3] is determined based on whether M8163 is set to ON or OFF.

32-bit instruction

The BINDA instruction converts each digit of the 32-bit binary data into an ASCII code (30H to 39H) in decimal format and stores the result in elements starting from [D].



The value in [S+1, S] ranges from -2,147,483,648 to +2,147,483,647.

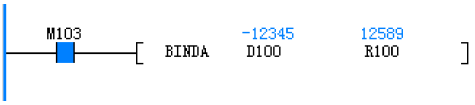
The operation result is as follows:

When the 32-bit number is positive, the sign bit is set to 20H (space). When it is negative, the sign bit is set to 2DH (minus sign).

When 0 exists on the left of valid digits, the sign bit is set to 20H (space).

The value in [D+5] is determined based on whether M8163 is set to ON or OFF.

Example



	Element Name	Data Type	Display Format	Current Value
1	D100	INT	Dec	-12345
2				
3	R100	INT	Hex	0x312D
4	R101	INT	Hex	0x3332
5	R102	INT	Hex	0x3534

5.7.1.10 WBIT

The WBIT instruction assigns the value of a word element to a combination of bit elements.

WBIT -- Conversion from word to bit

Instruction	Name	LD Expression	LiteST Expression
WBIT	Conversion from word to bit	WBIT ??? ??? ???	WBIT(???, ???, ???);

16-bit Instruction	WBIT: Continuous execution/WBITP: Pulse execution			
32-bit Instruction	DWBIT: Continuous execution/DWBITP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Value to be assigned to bit elements	-	INT and DINT
D	Bit element	Start number of bit elements	-	BOOL and array*n
n	Bit element quantity	Number of bit elements	1 to 16 or 1 to 32	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	√[1]	√	√	-	-	√	-	-	-
n	-	-	-	√	√	√	√	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

The WBIT instruction converts the binary number in S into bit states and assigns the conversion result to n bits starting from D.

Instruction Example



Program running flag

- Running: ON
- Stopped: OFF

	Element Name		Data Type	Display Format	Current Value
1	...	M0	BOOL	Binary	ON
2	...	M1	BOOL	Binary	OFF
3	...	M2	BOOL	Binary	OFF
4	...	M3	BOOL	Binary	OFF
5	...	M4	BOOL	Binary	OFF
6	...	M5	BOOL	Binary	OFF
7	...	M6	BOOL	Binary	OFF
8	...	M7	BOOL	Binary	ON
9	...	M8	BOOL	Binary	OFF
10	...	M9	BOOL	Binary	OFF
11	...	M10	BOOL	Binary	OFF
12	...	M11	BOOL	Binary	ON
13	...	M12	BOOL	Binary	OFF
14	...	M13	BOOL	Binary	OFF
15	...	M14	BOOL	Binary	OFF
16	...	M15	BOOL	Binary	ON
17	...	D100	INT	Binary	2#1000100010000001

5.7.1.11 WTOB

The WTOB instruction separates consecutive 16-bit/32-bit data by byte (eight bits).

WTOB: Conversion from word to byte

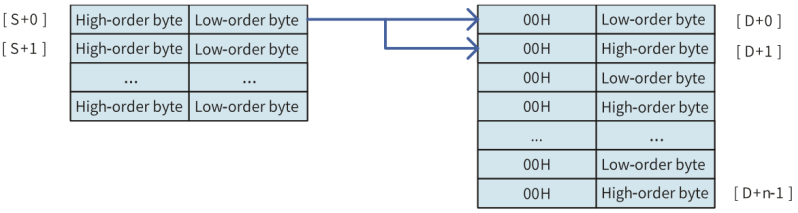
16-bit instruction	WTOB: Continuous execution WTOBP: Pulse execution			
32-bit instruction	DWTOB: Continuous execution DWTOBP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Source data	Start number of soft elements that store the data to be separated by byte	-	INT/DINT, array*n/2
D	Result	Start number of soft elements that store the data already separated byte by byte	-	INT, array*n
n	Separated data count	Number of bytes to be separated (n ≥ 0; no processing when n = 0)	0, 1 to 256	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

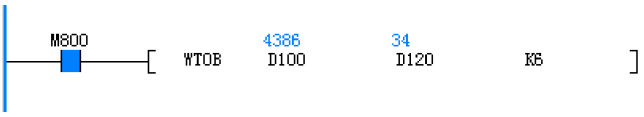
Function

The WTOB instruction separates the 16-bit/32-bit data in elements starting from [S] by byte and stores the bytes to the low-order eight bits of each of the n elements starting from [D]. 00H is stored in the high-order eight bits of each element.



An error occurs when the soft elements starting from [S] or [D] are out of the specified range.

Example

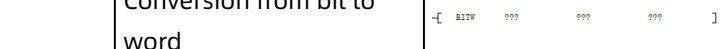


Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x5566
D120	16-bit INT	Hex	0x22
D121	16-bit INT	Hex	0x11
D122	16-bit INT	Hex	0x44
D123	16-bit INT	Hex	0x33
D124	16-bit INT	Hex	0x66
D125	16-bit INT	Hex	0x55
	16-bit INT	Hex	

5.7.1.12 BITW

The BITW instruction assigns the values of a combination of bit elements to a word element.

BITW: Conversion from bit to word

Instruction	Name	LD Expression	LiteST Expression
BITW	Conversion from bit to word		-

16-bit instruction	BITW: Continuous execution BITWP: Pulse execution			
32-bit instruction	BITDW: Continuous execution BITDWP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Bit element	Head element of bit elements	-	BOOL, array*n
D	Target data	Bit element combined value	-	INT/DINT
n	Bit element count	Number of bit elements	1-16/32	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The BITW instruction converts n bits starting from S into a binary number and transfers the conversion result to D.

Example



Program running flag

- Running: ON
- Stopped: OFF

	Element Name	Data Type	Display Format	Current Value
1	M0	BOOL	Binary	ON
2	M1	BOOL	Binary	OFF
3	M2	BOOL	Binary	OFF
4	M3	BOOL	Binary	OFF
5	M4	BOOL	Binary	OFF
6	M5	BOOL	Binary	OFF
7	M6	BOOL	Binary	OFF
8	M7	BOOL	Binary	ON
9	M8	BOOL	Binary	OFF
10	M9	BOOL	Binary	OFF
11	M10	BOOL	Binary	OFF
12	M11	BOOL	Binary	ON
13	M12	BOOL	Binary	OFF
14	M13	BOOL	Binary	OFF
15	M14	BOOL	Binary	OFF
16	M15	BOOL	Binary	ON
17	D100	INT	Binary	2#1000100010000001

5.7.1.13 BTOW

The BTOW instruction combines the low-order eight bits (low-order byte) of consecutive 16-bit/32-bit data.

BTOW: Conversion from byte to word

16-bit instruction	BTOW: Continuous execution BTOWP: Pulse execution			
32-bit instruction	BTODW: Continuous execution BTODWP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Source data	Start number of soft elements that store the data to be combined byte by byte	-	INT, array*n
D	Result	Start number of soft elements that store the data already combined byte by byte	-	INT/DINT, array*n/2
n	Combined data count	Number of bytes to be combined (n ≥ 0; no processing when n = 0)	0, 1 to 256	INT/DINT

The following table shows the soft elements.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The BTOW instruction combines the low-order eight bits of each of the n 16-bit data entries starting from [S] into 16-bit/32-bit data and stores the combination result in elements starting from [D]. The high-order eight bits of each source data entry ([S] and later) are ignored.

[S+0]	High-order byte	1st byte
[S+1]	High-order byte	2nd byte
	High-order byte	3rd byte
	High-order byte	4th byte
	...	...
	High-order byte	(n - 1)th byte
[S+n-1]	High-order byte	nth byte

2nd byte	1st byte
4th byte	3rd byte
...	...
nth byte	(n - 1)th byte

An error occurs when the soft elements starting from [S] or [D] are out of the specified range.

Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x5566
D103	16-bit INT	Hex	0x7788
D104	16-bit INT	Hex	0x99AA
D105	16-bit INT	Hex	0xBBCC
D120	16-bit INT	Hex	0x4422
D121	16-bit INT	Hex	0x8866
D122	16-bit INT	Hex	0xCCAA
D123	16-bit INT	Hex	0x0

5.7.1.14 UNI

The UNI instruction combines the low-order four bits of consecutive 16-bit data.

UNI: 4-bit combination of 16-bit data

16-bit instruction	UNI: Continuous execution UNIP: Pulse execution
32-bit instruction	-

Operands	Name	Description	Value range	Data type
S	Source data	Start number of soft elements that store the data to be combined	-	INT, array*n
D	Result	Number of the soft element that stores the data after combination	-	INT
n	Combined data count	Number of data entries to be combined (ranging from 0 to 4; no processing when n is 0)	0, 1-4	INT

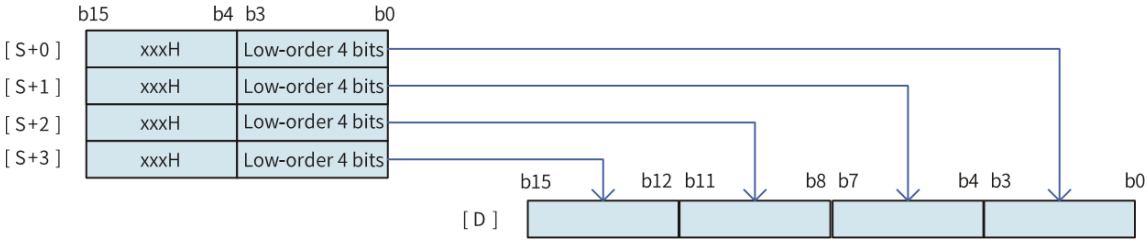
The list of soft elements is shown as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The UNI instruction combines the low-order four bits of each of the n 16-bit data entries starting from S into 16-bit data and stores the result in D.

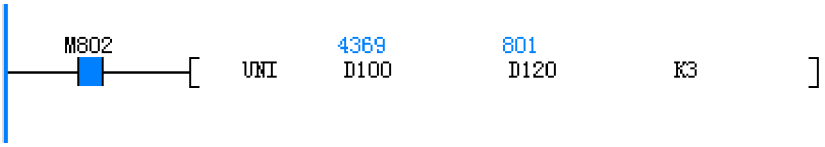
n ranges from 1 to 4. The instruction is not executed when n is 0. When n is in the range 1 to 3, the remaining upper bits are filled with 0s.



An operation error occurs in the following conditions.

- The element (S) is out of the specified range.
- n is out of the specified range.

Example

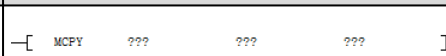


Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1111
D101	16-bit INT	Hex	0x2222
D102	16-bit INT	Hex	0x3333
D104	16-bit INT	Hex	0x0
	16-bit INT	Hex	
D120	16-bit INT	Hex	0x321
	16-bit INT	Hex	

5.7.1.15 MCPY

The MCPY instruction assigns data with specified length (in byte) to the target address without any change.

MCPY: Data copy (memory copy, type conversion) instruction

Instruction	Name	LD Expression	LiteST Expression
MCPY	Data copy (memory copy, type conversion)		MCPY (???, ???,???);

16-bit instruction	-			
32-bit instruction	MCPY: Continuous execution MCPYP: Pulse execution, 13 steps			
Operands	Name	Description	Value range	Data type
S	Source data address	Start number of elements that store the source data	-	Array*n, structure
D	Address of target data	Head component of target data	-	Array*n, structure
n	Length	Data length, in byte	-	-

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	√	√	√	√	√	-	-	-	-
D	√	√	√	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

> **CAUTION**
>
> This instruction is a higher-order application and must be used with caution.

This instruction implements the data copy operation with no change in the data. It can also implement memory copy, or even type conversion, if used skillfully.

n is the length of the data to be copied (in byte). For example, when two 16-bit data entries are assigned to a 32-bit data entry, n is 4; when two 32-bit integers are copies to structures or 16-bit integer arrays of the same size, n is 8.

When the operand S or D is a bit element, the addresses of the bit element must be aligned by byte. Otherwise, an addressing error will occur. For example, if the instruction is MCPY M1 M15 K1, the system will report "Invalid variable address: variable non-existent".

Example



Program running flag

- Running: ON
- Stopped: OFF

If "Size" is 5, then:

D0 Low-order byte =0x34 —> D100 Low-order byte = 0x34

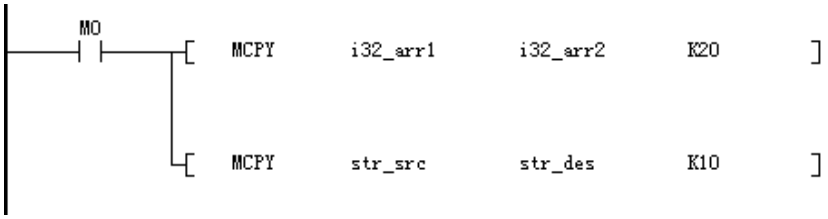
D0 High-order byte =0x12 —> D100 High-order byte= 0x12

D1 Low-order byte =0x78 —> D101 Low-order byte = 0x78

D1 High-order byte =0x56 —> D101 High-order byte=0x56

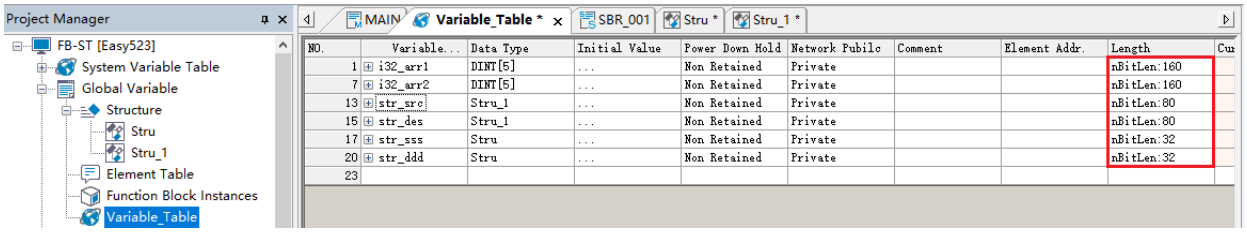
D2 Low-order byte =0x99 —> D102 Low-order byte =0x99

The MCPY instruction can be used to assign values between arrays and structures of the same type. As shown in the following figure, if the length of the array is DINT[5], data in i32_arr1 can be assigned to i32_arr2 (in byte). As each DINT data entry is 4 bytes, the data to be copied is 20 bytes (4 x 5 = 20). If the data to be assigned is of the structure type, its length and size must be consistent with the size of the structure, so that assignment can be performed properly.



After the program is completely compiled, the lengths of arrays and structures can be obtained in the variable table. The obtained length is in the unit of bit. One byte is equal to 8 bits, one word is equal to 16 bits, and one dword is equal to 32 bits. The formula for calculating the variable length is as follows:

Size = (Length + +7)/8



5.7.1.16 MSET

The MCPY instruction assigns data with specified length (in byte) to the target address without any change.

MSET: Data setting (memory setting and reset) instruction

Instruction	Name	LD Expression	LiteST Expression
MSET	Data setting (memory setting and reset)	-[MSET ??? ??? ???]	MSET(???,???,???);

16-bit instruction	-			
32-bit instruction	MSET: Continuous execution MSETP: Pulse execution, 13 steps			
Operands	Name	Description	Value range	Data type
S	Setting value	The value to be set, only single byte is valid	-	-
D	Address of target data	Head component of target data	-	Array*n, structure
n	Length	Data length, in byte	-	-

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	√	√	√	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

> **CAUTION**
>
> This instruction is a higher-order application and must be used with caution.

This instruction can set data in batches. It can be used to set fixed values in memory or reset the memory. It is typically used to clear structures or arrays.

n is the length of the data to be set (in byte). For example, when 0x12 is assigned to a 32-bit data entry, n is 4; when 0x1234 is set to two 32-bit elements, n is 8.

If you set 0x1234 to a 32-bit data entry in the unit of byte, the result is 0x34343434.

When the operand D is a bit element, the addresses of the bit element must be aligned by byte. Otherwise, an addressing error will occur. For example, if the instruction is MSET D0 M15 K1, the system will report "Invalid variable address: variable non-existent".

Example



Program running flag

- Running: ON
- Stopped: OFF

If "Size" is 5 and D0 is 0x1234, then:

D0 Low-order byte=0x34 — > D100 Low-order byte = 0x34

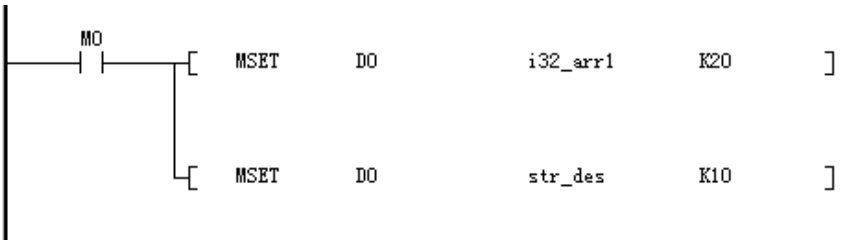
D0 Low-order byte=0x34 — > D100 High-order byte = 0x34

D0 Low-order byte=0x34 — > D101 Low-order byte = 0x34

D0 Low-order byte=0x34 — > D101 High-order byte = 0x34

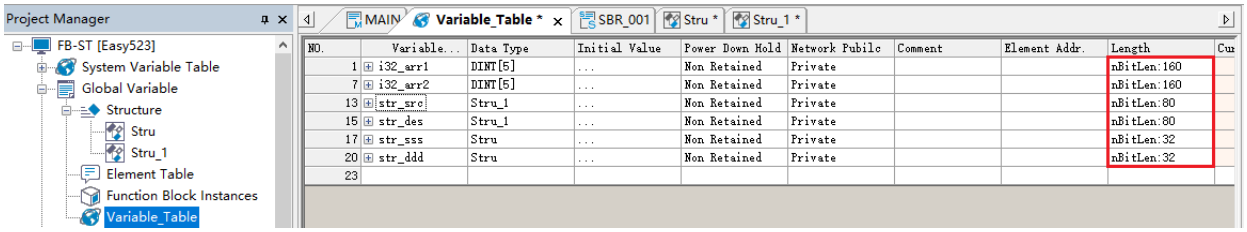
D0 Low-order byte=0x34 — > D102 Low-order byte = 0x34

The MSET instruction can be used to clear structures and arrays.



After the program is completely compiled, the lengths of arrays and structures can be obtained in the variable table. The obtained length is in the unit of bit. One byte is equal to 8 bits, one word is equal to 16 bits, and one dword is equal to 32 bits. The formula for calculating the variable length is as follows:

Size = (Length + 15)/16



5.7.1.17 DIS

The DIS instruction separates 16-bit data by every four bits.

DIS: 4-bit separation of 16-bit data

16-bit instruction	DIS: Continuous execution DISP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Source data	Start number of soft elements that store the data to be separated	-	INT
D	Result	Number of the soft element that stores the data after separation	-	INT, array*n
n	Separated data count	Number of separated data entries (value range: 0 to 4; no processing when n = 0)	0, 1 to 4	INT

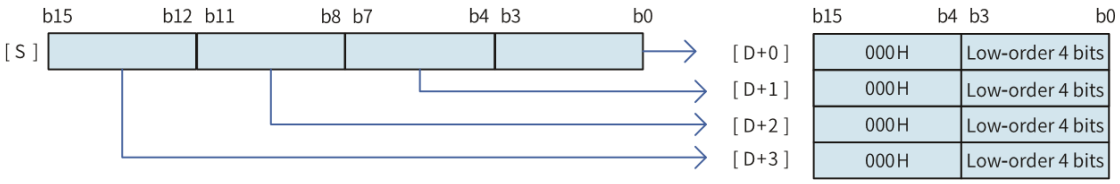
For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The DIS instruction separates the 16-bit data in S in 4-bit units and stores the separation results in the low-order four bits of each of the elements starting from D. The high-order 12 bits are filled with 0s.

n ranges from 1 to 4. The instruction is not executed when n is 0.

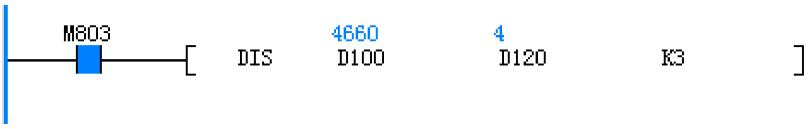


An operation error occurs in the following conditions.

- The element (D) is out of the specified range.
- n is out of the specified range.

Example

The 16-bit data in D100 is separated by every four bits. The result is stored in three consecutive D elements from start address D120.



	Element Name	Data Type	Display Format	Current Value
1	D100	INT	Hex	0x1234
2				
3	D120	INT	Dec	4
4	D121	INT	Dec	3
5	D122	INT	Dec	2
6	D123	INT	Dec	0

5.7.1.18 WTODW

The WTODW instruction assigns the values of 16-bit word elements to 32-bit word elements.

WTODW: Conversion from word to dword

16-bit instruction	-			
32-bit instruction	WTODW: Continuous execution WTODWP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Word element	Head element of word elements	-	INT, array*n
D	DWord element	Head element of doubleword elements	-	DINT, array*n
n	Element count	The number of elements	0, 1-256	DINT

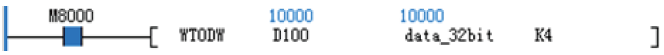
For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The WTODW instruction converts the 16-bit data in n elements starting from S into 32-bit data and stores the conversion result in n 32-bit registers starting from D. This instruction is used to convert the 16-bit data so that the data can be used in 32-bit instructions.

Example



Program running flag

- Running: ON
- Stopped: OFF

	Element Name	Data Type	Display Format	Current Value
1	D100	INT	Decimal	10000
2	D101	INT	Decimal	20000
3	D102	INT	Decimal	30000
4	D103	INT	Decimal	-20000
5	data_32bit[0]	DINT	Decimal	10000
6	data_32bit[1]	DINT	Decimal	20000
7	data_32bit[2]	DINT	Decimal	30000
8	data_32bit[3]	DINT	Decimal	-20000

5.7.1.19 DWTOW

The DWTOW instruction assigns the values of 32-bit word elements to 16-bit word elements.

DWTOW -- Conversion from dword to word

16-bit Instruction	-			
32-bit Instruction	DWTOW: Continuous execution/DWTOWP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Dword element	Start number of dword elements	-	DINT, array*n
D	Word element	Start number of word elements	-	INT, array*n
n	Element quantity	Number of elements	0, 1 to 256	DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The DWTOW instruction converts the 32-bit data in n elements starting from S into 16-bit data and stores the conversion result in n 16-bit registers starting from D. The data obtained by using this instruction can be used in 16-bit instructions.

> **NOTE**
>
> When the value to be converted in this instruction is greater than the upper limit of 16-bit data, only the low-order bits are retained after conversion.

Instruction Example



Program running flag

- Running: ON
- Stopped: OFF

	Element Name	Data Type	Display Format	Current Value
1	D100	DINT	Decimal	100000
2	D102	DINT	Decimal	-10000
3	D104	DINT	Decimal	20000
4	D106	DINT	Decimal	30000
5	data_16bit[0]	INT	Decimal	-31072
6	data_16bit[1]	INT	Decimal	-10000
7	data_16bit[2]	INT	Decimal	20000
8	data_16bit[3]	INT	Decimal	30000

5.7.1.20 ASCI

The ASCI instruction converts the value in S into ASCII codes and store the result in variables starting from D.

ASCI: Conversion from HEX into ASCII

16-bit instruction	ASCI: Continuous execution ASCIP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Data source	Address of the variable to be converted	-	INT, array*n
D	Conversion result	Start address of elements that store the ASCII codes after conversion	-	INT, array*n
n	Converted character count	n is the number of converted characters, which	1 to 256	INT

		ranges from 1 to 256.		

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

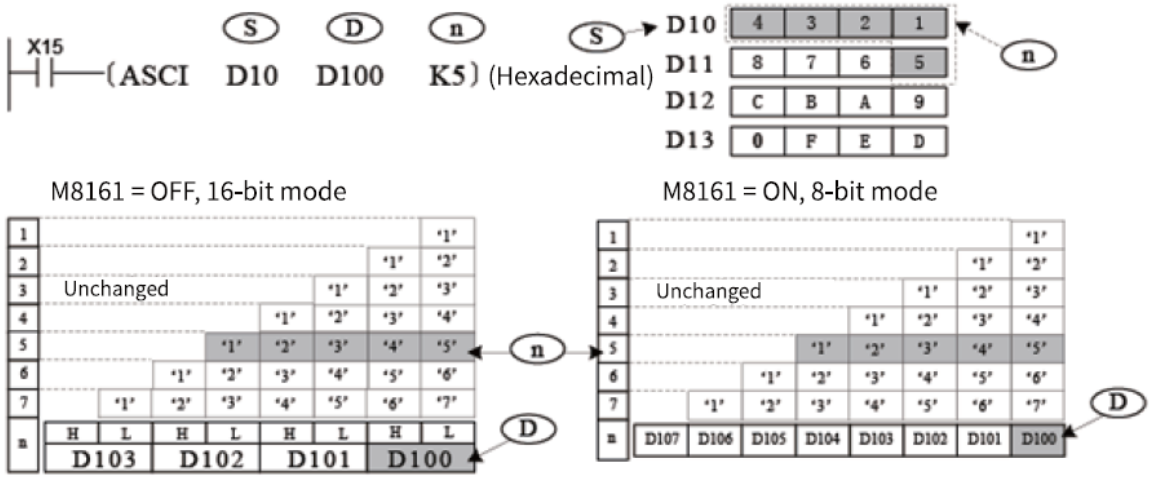
The value in S is encoded in ASCII format. The result is stored in variables from start address D. Specifically:

- S is the address of the variable to be converted.
- D is the start address for storing the ASCII characters after conversion
- n is the number of converted characters, in the range 1 to 256.

> **NOTE**
>
> Note during programming that instructions including HEX, ASCI, and CCD share M8161 mode flag.

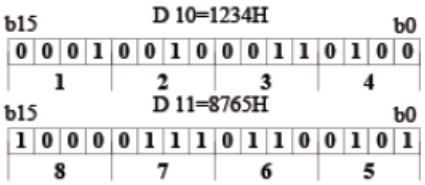
The ASCI instruction conforms to the ASCII-hexadecimal mapping table. For example, 0 in ASCII format corresponds to H30 in hexadecimal format, and F in ASCII format corresponds to H46 in hexadecimal format. For details about hexadecimal-ASCII mapping, see ASCII Conversion.

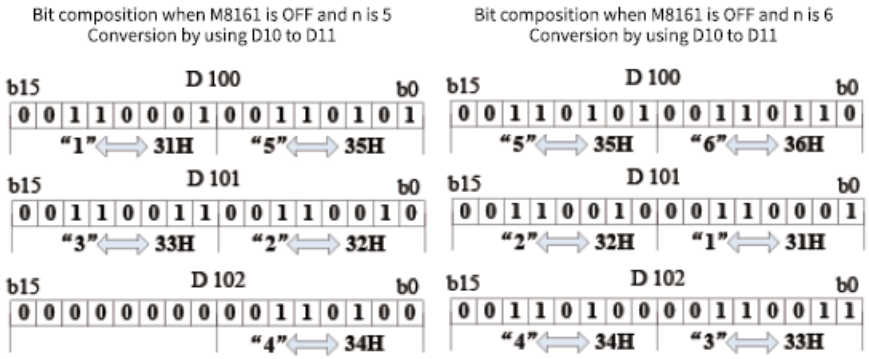
Example

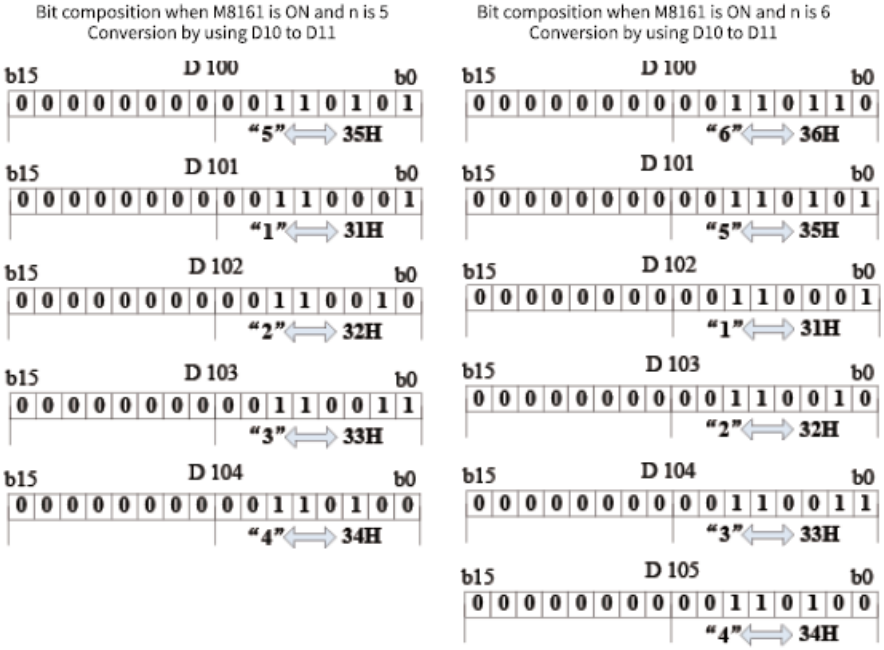


The M8161 flag determines the width mode of the target variable that stores the operation result. When M8161 is OFF, the 16-bit mode is used, whereby the operation result is stored in the high-order and low-order bytes of the variable separately. When M8161 is ON, the 8-bit mode is used, whereby the operation

result is stored only in the low-order byte of the variable. In this case, the length of the actually used variable area is increased.







5.7.1.21 HEX

The HEX instruction converts the values of variables starting from S to hexadecimal equivalents and stores the result in variables starting from D. You can specify the number of characters to convert and the storage mode.

HEX -- Conversion from ASCII to HEX

16-bit Instruction	HEX: Continuous execution/HEXP: Pulse execution
32-bit Instruction	-

Operand	Name	Description	Range	Data Type
S	Data source	Start address of variables to be converted. Register variables are converted and separated in units of 32 bits (four ASCII codes).	-	INT, array*n
D	Conversion result	Start address of variables that store the hexadecimal characters after conversion. The occupied variable space is related to S2.	-	INT, array*n
n	Converted character count	Number of converted characters	1 to 256	INT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

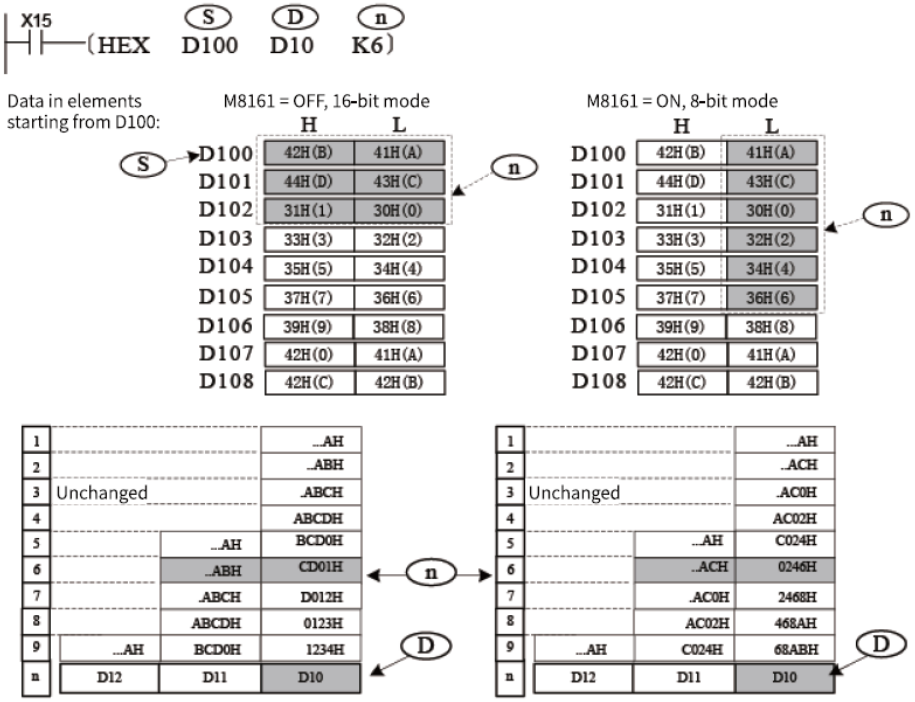
The HEX instruction converts the values of variables starting from S to hexadecimal equivalents and stores the result in variables starting from D. You can specify the number of characters to convert and the storage mode. In this instruction:

- S is the start address of variables to be converted. Register variables are converted and separated in units of 32 bits (four ASCII codes).
- D is the start address of variables for storing the hexadecimal characters after conversion. The occupied variable space is related to n.
- n is the number of converted characters, which ranges from 1 to 256.

NOTE

- The HEX, ASCI, and CCD instructions share the M8161 mode flag during programming.
- The source data in S must be ASCII codes. Otherwise, a conversion error occurs.
- If the format of the output data is BCD, BCD-to-BIN conversion is required on the hexadecimal characters after conversion to get the correct value.

Instruction Example



The M8161 flag determines the variable width mode. When M8161 is OFF, the 16-bit mode is used, whereby both the high-order and low-order bytes of variables are taken for the operation. When M8161 is ON, the 8-bit mode is used, whereby only the low-order bytes of variables are taken for the operation and the high-order bytes are discarded. In this case, the length of the actually used variables in the S area is increased.

Bit composition when M8161 is OFF and n is 5
Conversion by using D100 to D102 (high- and low-order bytes)

D 10

1	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0
B				C				D				0			

D 11

0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
															A

Bit composition when M8161 is OFF and n is 6
Conversion by using D100 to D102 (high- and low-order bytes)

D 10

1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	1
C				D				0				1			

D 11

0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	1
												A			B

Bit composition when M8161 is ON and n is 5
Conversion by using D100 to D104 (low-order bytes)

D 10

1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0
C				0				2				4			

D 11

0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
															A

Bit composition when M8161 is ON and n is 6
Conversion of D100 to D105 (low-order bytes)

D 10

0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0
0				2				4				6			

D 11

0	0	0	0	0	0	0	0	1	0	1	0	1	1	0	0
												A			C

5.7.1.22 DECO

The DECO instruction decodes data and stores the result.

DECO: Data decoding

16-bit instruction	DECO: Continuous execution DECOP: Pulse execution
32-bit instruction	DDECO: Continuous execution DDECOP: Pulse execution

Operands	Name	Description	Value range	Data type
S	Decoding source	Source data to be decoded	-	INT, DINT
D	Decoding result	Address of the element that stores the decoding result	-	BOOL
n	Bit length of the decoding source	Bit length of source data to be decoded	0 to 8	INT, DINT

The list of soft elements is shown as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	√[1]	√	√	-	-	-	-	-	-
n	-	-	-	√	√	√	√	-	-

NOTE

[1] The X element is not supported.

Function

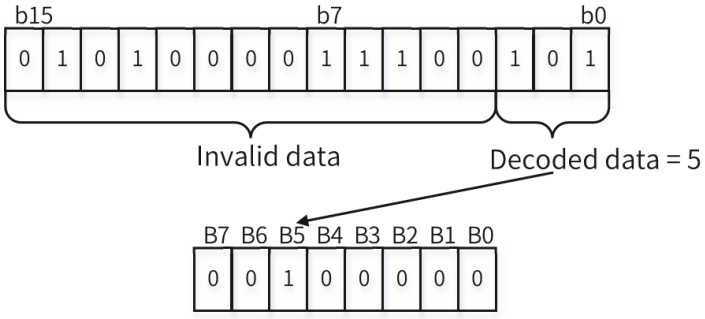
The DECO instruction calculates the value of the low-order n bits of S, sets the bit variable or element pointed to by this value among the 2n elements starting from D to ON, and resets other bits.

- Words are decoded into bits.
- When n is 0, the instruction is not executed. When n is beyond the range of 0 to 8, an operation error occurs. When n is 8, D includes a total of 256 bit variables or elements.
- If the instruction flow is OFF, the instruction is not executed.
- The instruction of the pulse execution type is generally used.

Example

When the M345 flow is ON, the DECO instruction is executed. The low-order 3 bits of D400 is calculated to obtain 5. Then B5 among the 23 (8) bit elements starting from B0 is set to ON.





5.7.1.23 ENCO

The ENCO instruction encodes data and stores the result.

ENCO: Data encoding

16-bit instruction	ENCO: Continuous execution/ENCOP: Pulse execution			
32-bit instruction	DENCO: Continuous execution DENCOP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Encoding source	Source data to be encoded	-	BOOL
D	Encoding result	Address of the element that stores the encoding result	-	INT, DINT
n	Bit length of the encoding result	Data bit length of the encoding result	0 to 8	INT, DINT

See the following table for the soft elements.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	√	√	√	-	-	-	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

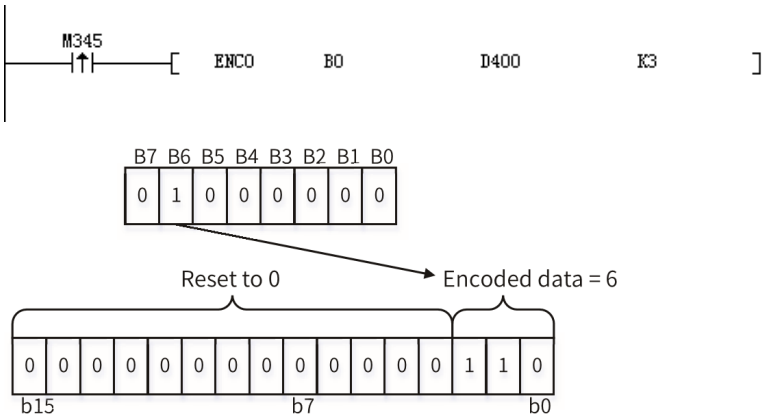
The ENCO instruction obtains the position of the bit variable or element that is set to ON among the 2n S bit variables or elements, converts it into a value and assigns the value to the low-order n bits of D, and resets other bits of D.

- Bits are encoded into words.

- This instruction can be used to determine whether there is a bit element set to ON among multiple consecutive bit elements.
- When n is 0, the instruction is not executed. When n is beyond the range of 0 to 8, an operation error occurs. When n is 8, S includes a total of 256 bit variables or elements.
- If multiple bits in the source address are 1, only the first ON bit on the high-order side is calculated. If all bits in the source address S are 0, an operation error occurs.
- If the instruction flow is OFF, the instruction is not executed.
- The instruction of the pulse execution type is generally used.

Example

When the M345 flow is ON, the ENCO instruction is executed. The ON bit B6 is obtained among the 23 (8) bit elements starting from B0. Then the value 6 is assigned to the low-order 3 bits of D400, and other bits are cleared.



5.7.1.24 SCALE_X

The SCALE_X instruction calibrates the standard parameter VALUE of the REAL type based on the data type and range specified by MIN and MAX by using the following formula: OUT = VALUE * (MAX - MIN) + MIN. In this formula, VALUE is in the range of 0.0 to 1.0.

SCALE_X -- Scaling

32-bit Instruction	SCALE_X: Continuous execution/SCALE_XP: Pulse execution			
32-bit Instruction	DSCALE_X: Continuous execution/DSCALE_XP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Lower limit	Minimum value of the range	-	INT and DINT
S2	Upper limit	Maximum value of the range	-	INT and DINT
S3	Input value	Value to be scaled	-	REAL
D	Output value	Scaling result	-	INT and DINT

The following table lists soft elements in the instruction.

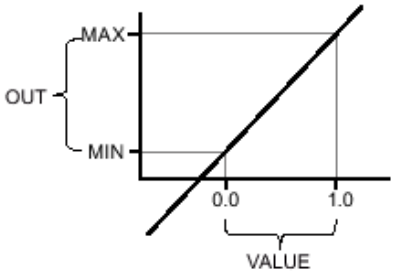
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The VALUE parameter in the SCALE_X instruction must be in the range of 0.0 to 1.0.

The SCALE_X instruction can be used to map the value of VALUE to the specified range for scaling. When the SCALE_X instruction is executed, the floating-point number of VALUE is scaled to the range specified by MIN (lower limit) and MAX (upper limit). The scaling result is an integer and stored in OUT.

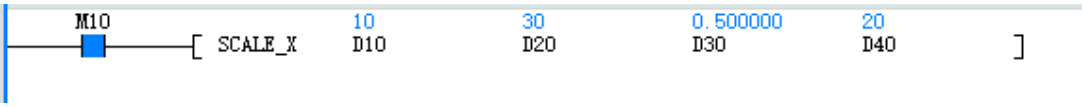
The following figure shows how to scale a value.



An error is reported when one of the following conditions is met:

- The value of MIN is greater than or equal to the value of MAX.
- The value of VALUE is less than 0.0 or greater than 1.0.
- The value of MIN is less than the negative limit or the value of MAX is greater than the positive limit of the specified data type.
- The value of VALUE is NaN, which indicates an invalid arithmetic operation result.

Instruction Example



When M10 is ON, the lower limit of the range D10 is K10, the upper limit of the range D20 is K30, the input value D30 is E0.5, and output result is K20.

5.7.1.25 NORM_X

The NORM_X instruction normalizes VALUE in the range specified by MIN and MAX according to the following formula: VALUE:OUT = (VALUE – MIN)/(MAX – MIN). In this formula, OUT is in the range of 0.0 to 1.0.

NORM_X -- Normalization

32-bit Instruction	NORM_X: Continuous execution/NORM_XP: Pulse execution			
32-bit Instruction	DNORM_X: Continuous execution/DNORM_XP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Lower limit	Minimum value of the range	-	INT and DINT
S2	Upper limit	Maximum value of the range	-	INT and DINT
S3	Input value	Value to be normalized	-	INT and DINT
D	Output value	Normalized result	-	REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The VALUE parameter in the NORM_X instruction must be in the range of MIN to MAX.

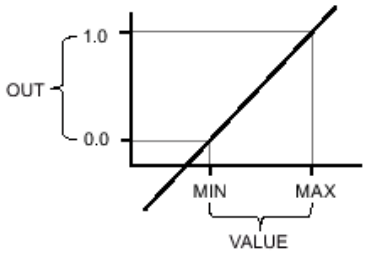
The NORM_X instruction normalizes the variable value in VALUE based on a linear scale.

The range of the linear scale is defined by MIN and MAX.

The result in OUT is calculated and stored as a floating-point number according to the position of the value to be normalized in the range.

- If the value to be normalized is equal to the value of MIN, the return value of OUT is 0.0.
- If the value to be normalized is equal to the value of MAX, the return value of OUT is 1.0.

The following figure shows how to normalize a value.



An error is reported when one of the following conditions is met:

- The value of MIN is greater than or equal to the value of MAX.
- The value of VALUE is less than that of MIN or greater than that of MAX.
- The value of MIN is less than the negative limit or the value of MAX is greater than the positive limit of the specified data type.
- The value of VALUE is NaN, which indicates an invalid arithmetic operation result.

Instruction Example



When M11 is ON, the lower limit of the range D10 is K10, the upper limit of the range D20 is K30, the input value D30 is K20, and output result is E0.5.

5.7.1.26 ROUND

The ROUND instruction rounds the part after the decimal point up to convert a floating-point number into an integer.

ROUND -- Round up

32-bit Instruction	ROUND: Continuous execution/ROUNDP: Pulse execution			
32-bit Instruction	DROUND: Continuous execution/DROUNDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Data source	Input value to be rounded up	-	REAL
D	Operation result	Value rounded up to	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

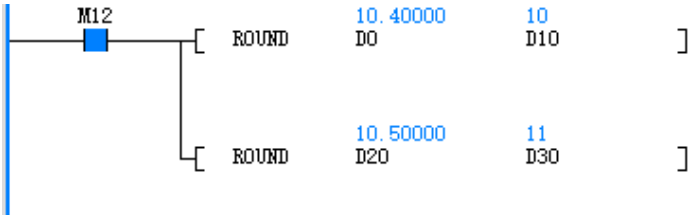
NOTE

If the operation result is beyond the range specified by the upper and lower limits of the data type, the upper or lower limit is used as the output.

Function and Instruction Description

The ROUND instruction rounds up the value specified by S1 to the nearest integer and stores the result in D.

Instruction Example



- When M12 is ON, the value of D0, that is, E10.4 is rounded up to an integer and output value of D10 is K10.
- The value of D20, that is, E10.5 is rounded up to an integer and output value of D30 is K11.

5.7.1.27 TRUNC

The TRUNC instruction converts a real number into an integer. The fractional part of the real number is truncated to 0. IEEE indicates that the number is rounded to the nearest integer towards zero.

TRUNC -- Truncation

32-bit Instruction	TRUNC: Continuous execution/TRUNCP: Pulse execution			
32-bit Instruction	DTRUNC: Continuous execution/DTRUNCP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Data source	Input value to be truncated	-	REAL
D	Operation result	Value after truncation	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

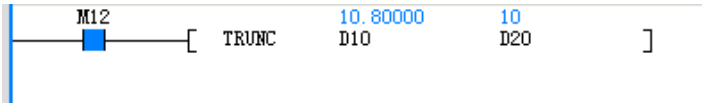
NOTE

If the operation result is beyond the range specified by the upper and lower limits of the data type, the upper or lower limit is used as the output.

Function and Instruction Description

The TRUNC instruction takes only the integral part from the value specified by S1 and stores the result in D.

Instruction Example



When M12 is ON, the input value of D30 is E10.8, D10 is truncated into an integer, and the output is K10.

5.7.1.28 CELL

The CELL instruction converts a floating-point number into an integer that is greater than or equal to the selected value of the REAL type. IEEE indicates celling up to an infinite positive number.

CELL -- Floating-point number celling

32-bit Instruction	CELL: Continuous execution/CELLP: Pulse execution			
32-bit Instruction	DCELL: Continuous execution/DCELLP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Data source	Input value to be celled up	-	REAL
D	Operation result	Adjacent greater integer	-	INT and DINT

The following table lists soft elements in the instruction.

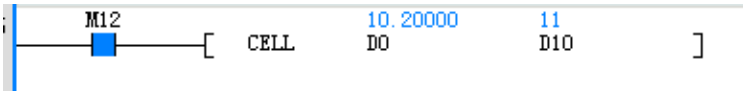
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

NOTE

If the operation result is beyond the range specified by the upper and lower limits of the data type, the upper or lower limit is used as the output.

Function and Instruction Description

The CELL instruction cells up the value specified by S1 to the adjacent integer and stores the result in D.



When M12 is ON, the value of D0, that is, E10.2 is celled up to an integer and output value of D10 is 11.

5.7.1.29 FLOOR

The FLOOR instruction converts a floating-point number into an integer that is less than or equal to the selected value of the REAL type. IEEE indicates flooring down to an infinite negative number.

FLOOR -- Floating-point floor down

32-bit Instruction	FLOOR: Continuous execution/FLOORP: Pulse execution			
32-bit Instruction	DFLOOR: Continuous execution/DFLOORP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Data source	Input value to be floored down	-	REAL
D	Operation result	Adjacent smaller integer	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

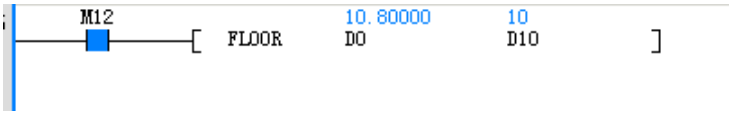
NOTE

If the operation result is beyond the range specified by the upper and lower limits of the data type, the upper or lower limit is used as the output.

Function and Instruction Description

The FLOOR instruction floors down the value specified by S1 to the adjacent integer and stores the result in D.

Instruction Example



When M12 is ON, the value of D0, that is, E10.8 is floored down to an integer and output value of D10 is K10.

5.7.1.30 AryByteTo

Return Value Type	Conversion from byte array elements to specific variable type		
BOOL			
Operand	Description	Range	Data Type
S1	Data to be converted	-	Byte array
S2	Number of elements to convert	-	BYTE, INT, and DINT
S3	Conversion order	0 or 1	BYTE, INT, and DINT
S4	Conversion result	-	BOOL, BYTE, INT, DINT, REAL, STRING, STRUCT, and array of these types

The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S4	√	√	√	√	√	√	-	-	-	-	-	-	-	-

Function and Instruction Description

- Function description
  - The AryByteTo instruction converts the elements of the number specified by S2 in the data to be converted starting from S1[] into the conversion result S4 of the specified data type and size and stores the result in S4.
  - The following table describes the meanings of S3 values that indicate the conversion order of elements in S1[].

S3 Value	Meaning
0	_LOW_HIGH: Lower byte first, higher byte last
1	_HIGH_LOW: Higher byte first, lower byte last

- Conversion rules

Byte Size by S4 Type	S4 Type	S3 Conversion Order	Array	Conversion Rule
2 bytes and more	◦ INT, DINT, and REAL ◦ String of more than 2 bytes ◦ Array with total size of all elements over 2 bytes, one element in the array of over 2 bytes, structure with the total size of all structure elements over 2 bytes, or one structure element in the structure of over 2 bytes	_LOW_HIGH or _HIGH_LOW	Yes or No	See rule 1.
1 byte	◦ Byte ◦ String of 1 byte ◦ Array with total size of all elements as 1 byte, one element in the array of 1 byte,	_LOW_HIGH	No	Store S1[0] in S4.

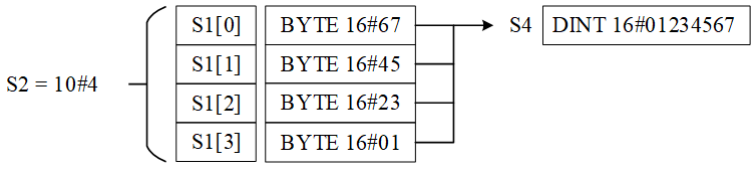
Byte Size by S4 Type	S4 Type	S3 Conversion Order	Array	Conversion Rule
	structure with the total size of all structure elements as 1 byte, or one member in the structure of 1 byte		Yes	Store S1[i] in S4[i].
		_HIGH_LOW	No	Store S1[1] in S4.
			Yes	See rule 2.
1 bit	BOOL	_LOW_HIGH	No	Store the 0th bit of S1[0] in S4.
			Yes	See rule 3.
		_HIGH_LOW	No	Store the 0th bit of S1[1] in S4.
			Yes	See rule 4.

◦ Rule 1

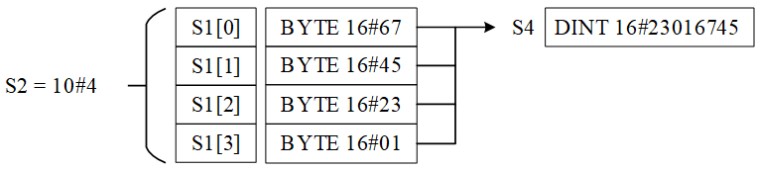
1. Combine S1[0] and S1[1] to create a word (two bytes) based on the conversion order specified by S3. If S3 is _LOW_HIGH, store higher bytes in S1[1] and lower bytes in S1[0]. If S3 is _HIGH_LOW, store higher bytes in S1[0] and lower bytes in S1[1].
2. Combine the elements next to In by two bytes in the same way to obtain multiple words.
3. Combine these words obtained in the previous step based on the data type and size specified by S4 and store the data in S4. For example, if the data type of S4 is DINT, combine the two words.
4. Store the obtained data in OutVal.

Example

- The data type of S4 is DINT, the S2 value is 4, and the conversion order is _LOW_HIGH.



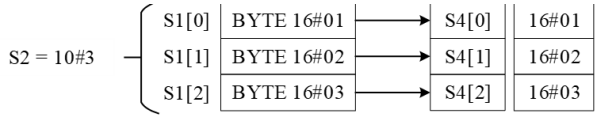
- The data type of S4 is DINT, the S2 value is 4, and the conversion order is _HIGH_LOW.



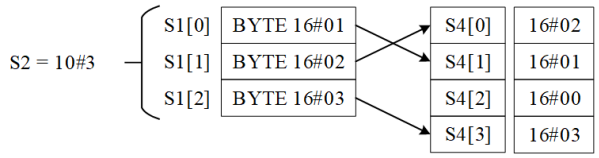
◦ Rule 2: Store S1[i] (i is an even number) in S4[i+1] and S1[i] (i is an odd number) in S4[i-1]. If the size of the S2 value is an odd number, store the data in S4[Size] and 0x00 in S4[Size-1].

Example

- The data type of S4 is byte array, the S2 value is 3, and the conversion order is _LOW_HIGH.



- The data type of S4 is byte array, the S2 value is 3, and the conversion order is _HIGH_LOW.



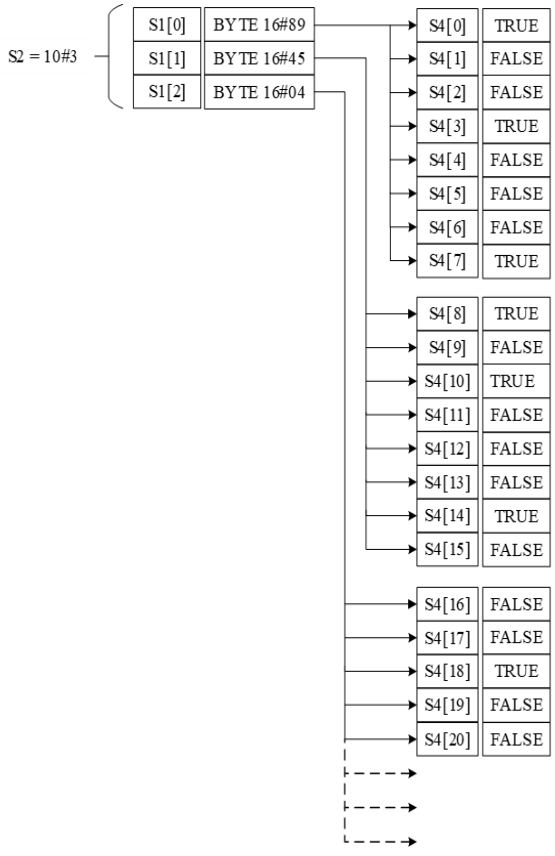
◦ Rule 3

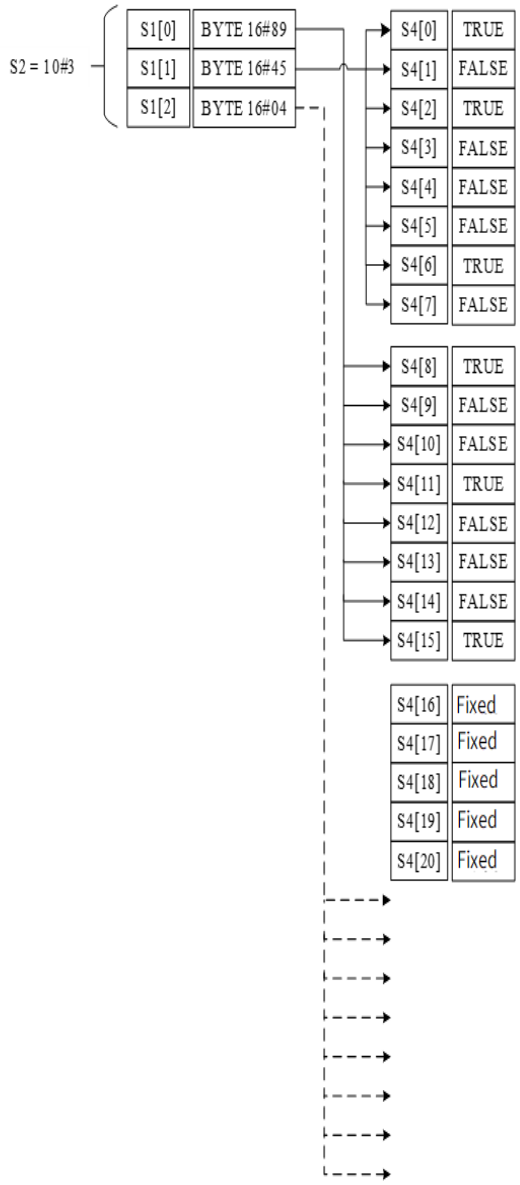
1. Split the S1[0] value into 8 bits and store them to S4[0] to S4[7] separately.
2. Split the S1[1] value into 8 bits and store them to S4[8] to S4[15] separately.
3. Repeat the preceding steps to process other data and discard excess bits.

◦ Rule 4

1. Split the S1[0] value into 8 bits and store them to S4[8] to S4[15] separately.
2. Split the S1[1] value into 8 bits and store them to S4[0] to S4[7] separately.
3. Repeat the preceding steps to process other data and discard excess bits.

Example: The data type of S2 is BOOL array and the number of elements is 21. The following table describes the conversion rules when the conversion order is _LOW_HIGH or _HIGH_LOW.

_LOW_HIGH	_HIGH_LOW
	

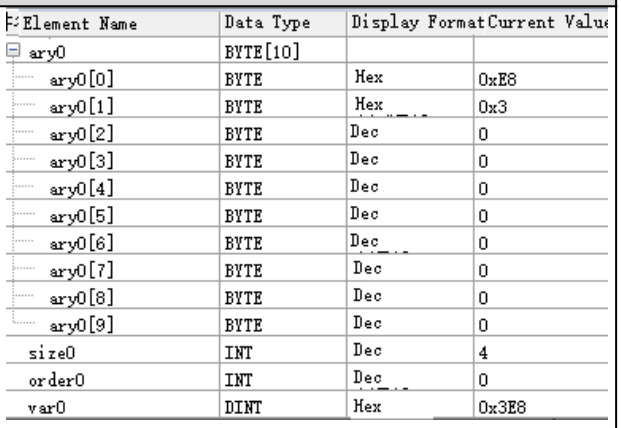
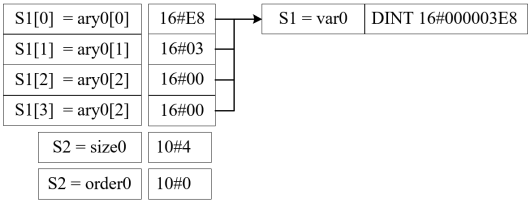
_LOW_HIGH	_HIGH_LOW
	

Instruction Example

Item	ST
Defined variable	NO. | Variable Name | Data Type | Initial Value 1 | Return_Bool | BOOL | OFF 2 | ary0 | BYTE[10] | ... 13 | size0 | INT | 0 14 | order0 | INT | 0 15 | var0 | DINT | 0 16

Program	`Return_Bool := AryByteTo(ary0[0],`
	`size0,`
	`order0,`
	`var0);`

Operation result	

Item	ST
	
Principle	

NOTE

- S4 specifies the conversion result, which can be a single variable or array variables. If S4 specifies array variables, you only need to enter the array name without square bracket.
- If the number of elements specified by S2 is less than the size specified by S4, conversion is performed based on the S2 value. Otherwise, conversion is performed based on the S4 value. If S1[] does not have sufficient bytes, S4 remains unchanged.
- When the data type of S4 is Struct, part of the S1[] value can be inserted into adjustment areas between members depending on the composition.
- When the S2 value is 0, the return value is TRUE and S4 remains unchanged.
- When the S2 value exceeds the array area of S1[], the return value is FALSE and S4 remains unchanged.

5.7.1.31 AryToString

Return Value Type	Conversion from byte array into string		
BOOL			
Operand	Description	Range	Data Type
S1	Byte array to convert	-	Byte array
S2	Number of elements to convert	0 to 255	BYTE, INT, and DINT
S3	String after conversion	-	STRING

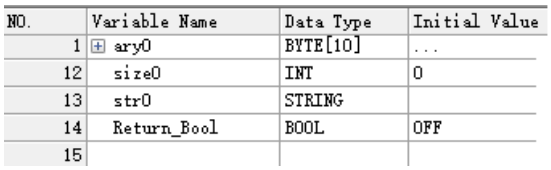
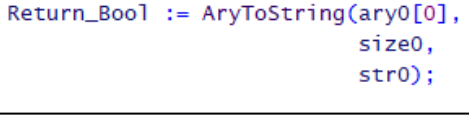
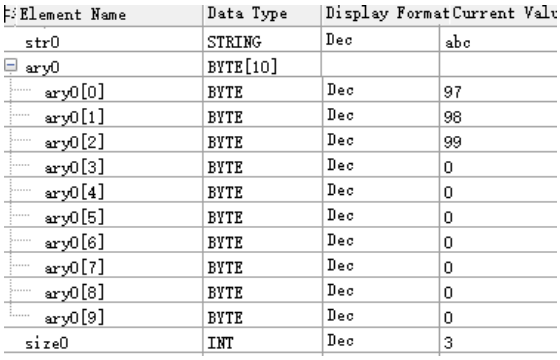
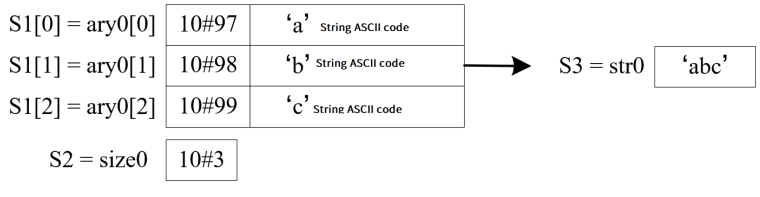
The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S3	-	-	-	-	-	√	-	-	-	-	-	-	-	-

Function and Instruction Description

- The AryToString instruction processes elements in the byte array starting from S1[0] as ASCII codes, converts them into corresponding characters, and store the result in the string specified by S3. S2 specifies the number of elements to convert in S1[].
- S1 is a member of the byte array, such as a[2].

Instruction Example

Item	ST
Defined variable	
Program	
Operation result	
Principle	


```
Return_Bool := AryToString(ary0[0],
                           size0,
                           str0);
```

NOTE

- The data type of S1 must be BYTE. Otherwise, an error occurs during compilation.
- When the S2 value is 0, S3 is a null string and the function returns TRUE.
- If the S2 value is beyond the array area of S1[] or greater than the maximum length of characters specified by S3, the function returns FALSE and S3 remains unchanged.
- When the S2 value is negative, the function returns FALSE and S3 remains unchanged.

5.7.1.32 BOOL_TO_(TYPE)

Return value type	Convert the BOOL type into the type specified by < TYPE>.		
<TYPE>			
Operands	Description	Value range	Data type
S	Variable to be converted	-	BOOL

The following < TYPE> options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	√	√	√	-	√	-	-

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S	√	√	√	-	-	-	-	-	-	-	√	-

Function and Instruction Description

This instruction is used to explicitly convert the variable of the BOOL type into the variable of the type specified by < TYPE>. Specifically:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST Code	Result
byte0 := BOOL_TO_BYTE(TRUE)	1
int0 := BOOL_TO_INT(TRUE)	1
dint0 := BOOL_TO_DINT(TRUE)	1
real0 := BOOL_TO_REAL(TRUE)	1.0
string0 := BOOL_TO_STRING(TRUE)	'TRUE'

NOTE

- For the numeric type, if the operand is TRUE, the conversion result is 1. If the operand is FALSE, the result is 0.
- When S is an expression, its result will be implicitly converted into the BOOL type and then converted again by using BYTE_TO_<TYPE>.

5.7.1.33 DINT_TO_(TYPE)

Return value type	This instruction converts DINT type into the type specified by (TYPE).		
(TYPE)			
Operands	Description	Value range	Data type
S	Variable to be converted	-	BYTE/INT/DINT

The following (TYPE) options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	√	-	√	√	√	-	-

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	-	-	-

Function and Instruction Description

This instruction is used to explicitly convert the variable of the DINT type into the variable of the type specified by (TYPE). Specifically:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST Code	Result
bool0 := DINT_TO_BOOL(10)	TRUE
byte0 := DINT_TO_BYTE(10)	10
int0 := DINT_TO_INT(10)	10
real0 := DINT_TO_REAL(10)	10.0
string0 := DINT_TO_STRING(10)	'10'

NOTE

- For the boolean type, if the operand is 0, the conversion result is FALSE. If the operand is any other value, the result is TRUE.
- When S is an expression, its result will be implicitly converted into the DINT and then converted again by using DINT_TO_(TYPE).

5.7.1.34 INT_TO_(TYPE)

Return value type	Convert the INT type into the type specified by (TYPE).		
TYPE			
Operands	Description	Value range	Data type
S	Variable to be converted	-	BYTE/INT

The following (TYPE) options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	-	√	√	√	√	-	-

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	-	-	-	-

Function

This instruction is used to explicitly convert the variable of the INT type into the variable of the type specified by (TYPE). Specifically:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST Code	Result
bool0 := INT_TO_BOOL(10)	TRUE
byte0 := INT_TO_BYTE(10)	10
dint0 := INT_TO_DINT(10)	10
real0 := INT_TO_REAL(10)	10.0
string0 := INT_TO_STRING(10)	'10'

NOTE

- For the boolean type, if the operand is 0, the conversion result is FALSE. If the operand is any other value, the result is TRUE.
- When S is an expression, its result will be implicitly converted into the INT and then converted again by using INT_TO_(TYPE).

5.7.1.35 REAL_TO_(TYPE)

Return value type	This instruction converts REAL type into the type specified by <TYPE>.		
<TYPE>			
Operands	Description	Value range	Data type

S	Variable to be converted	-	BYTE/INT/REAL

The following <TYPE> options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	√	√	-	√	√	-	-

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to explicitly convert the variable of the REAL type into the variable of the type specified by <TYPE>. Specifically:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST Code	Result
bool0 := REAL_TO_BOOL(11.2)	TRUE
byte0 := REAL_TO_BYTE(11.2)	11
int0 := REAL_TO_INT(11.2)	11
dint0 := REAL_TO_DINT(11.2)	11
string0 := REAL_TO_STRING(11.2)	'11.2'

NOTE

- For the boolean type, if the operand is 0, the conversion result is FALSE. If the operand is any other value, the result is TRUE.
- For the numeric type, the general rule for rounding applies during conversion.
- When S is an expression, its result will be implicitly converted into the REAL type and then converted again by using REAL_TO_ <TYPE>.

5.7.1.36 BYTE_TO_(TYPE)

Return value type	This instruction converts the BYTE type into the type specified by <TYPE>.		
<TYPE>			
Operands	Description	Value range	Data type
S	Variable to be converted	-	BYTE

The following <TYPE> options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	-	√	√	√	√	√	-	-

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	-	√	-	√	-	-	-	-	-

Function

This instruction is used to explicitly convert the variable of the BYTE type into the variable of the type specified by < TYPE>. Specifically:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST code	Result
bool0 := BYTE_TO_BOOL(10)	TRUE
int0 := BYTE_TO_INT(10)	10
dint0 := BYTE_TO_DINT(10)	10
real0 := BYTE_TO_REAL(10)	10.0
string0 := BYTE_TO_STRING(10)	'10'

NOTE

- For the boolean type, if the operand is 0, the conversion result is FALSE. If the operand is any other value, the result is TRUE.
- When S is an expression, its result will be implicitly converted into the BYTE type and then converted again by using BYTE_TO_<TYPE>.

5.7.1.37 STRING_TO_(TYPE)

Return Value Type	Conversion from string to <TYPE>		
<TYPE>			
Operand	Description	Range	Data Type
S	Variable to be converted	-	STRING

The following table lists the supported <TYPE> options.

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	√	√	√	√	-	-	-

The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S	-	-	-	-	-	√	-	-	-	-	-	-	√	-

Function and Instruction Description

The STRING_TO_(TYPE) instruction explicitly converts the variable of the STRING type to the variable of the type specified by <TYPE>. In this instruction:

- S is the variable to be converted.
- The return value is the conversion result.

Instruction Example

LiteST Code	Result
bool0 := STRING_TO_BOOL('true')	TRUE
byte0 := STRING_TO_BYTE('11')	11
int0 := STRING_TO_INT('11')	11
dint0 := STRING_TO_DINT('11')	11
real0 := STRING_TO_REAL('1.23')	1.23

NOTE

- When the <TYPE> option is not BOOL, for example, STRING_TO_BYTE, STRING_TO_INT, STRING_TO_DINT, or STRING_TO_REAL, the source string In can contain only digits 0 to 9. Pay attention to the following special processing:
  - If the variable to be converted is signed, the first character (positive sign or negative sign) in the string In must be processed as a sign.
  - Spaces before and after In are ignored.
  - Underscores (_) before and after In are ignored. An underscore (_) can be inserted in the middle of a string, but characters in the string must be continuous.
  - If In is invalid as a whole but the characters at the beginning are valid, the output is these valid characters. For example, if In is 5abc, the output is 5.
  - If In contains one or more spaces in the middle, the output is the characters before the first space.
  - Other non-compliant character strings are not converted.
- When the <TYPE> option is BOOL, for example, STRING_TO_BOOL, the variable of the BOOL type can be set to TRUE if the source string In is TRUE or true. Otherwise, the variable is set to FALSE. Pay attention to the following special processing:
  - In must be a string of continuous characters TRUE or true. If any other character is contained in the middle, FALSE is returned.
  - Spaces before and after In are ignored.
  - Underscores (_) before In are not ignored, and FALSE is returned.
  - If In is invalid as a whole but the characters at the beginning are valid, the output is these valid characters. For example, if In is TRUEabc, the output is TRUE.
  - If In contains one or more spaces in the middle, the output is the characters before the first space.
  - Other non-compliant character strings are not converted.

5.7.1.38 StringToAry

Return value type	This instruction takes the character codes (ASCII codes) in the text string as numbers and stores them individually in a Byte array.		
BOOL			
Operands	Description	Value range	Data type
S1	Source string	-	STRING
S2	Byte array	-	Byte array
S3	Number of bytes after conversion	0 to 255	BYTE/INT/DINT

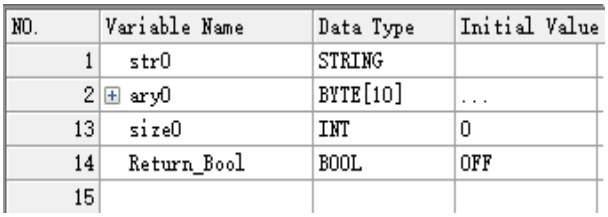
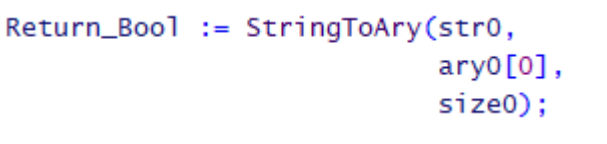
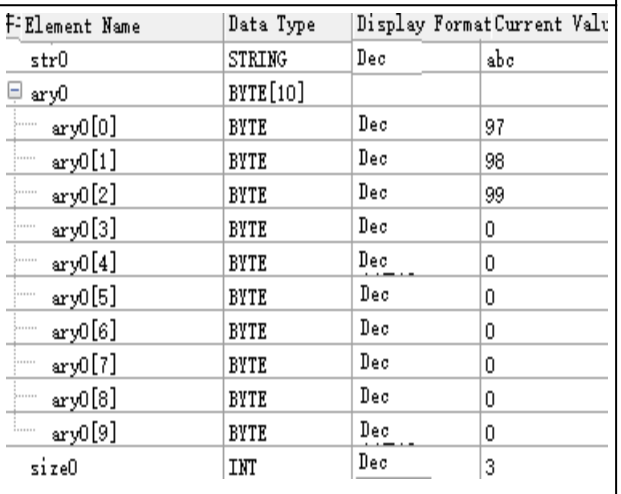
For soft elements, see the following table.

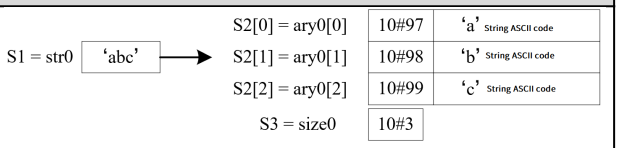
Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom string variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	-	√	-	-	-	-	-	-	√	-
S2	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S3	-	-	-	-	√	-	-	-	-	-	-	-	-	-

Function

- This instruction takes the character codes in text string S1 as numbers and stores them individually in a Byte array starting from *S2[0]*. The number of bytes after conversion is stored in S3.
- S1 supports constant strings.
- S2 include Byte array members, such as a[2].

Example

Items	ST
Define variables	
Program	
Running result	
Work principle	

Items	ST
	

NOTE

- '\0' at the end of S1 is not stored in *S2[]*.
- When the S1 text string contains only the '\0' character, the value of S3 is 0, and *S2[]* does not change.
- When the number of bytes of S1 is greater than the number of elements in *S2[]*, the returned value is FALSE, and *S2[]* doees not change.

5.7.1.39 TO_(TYPE)

Return value type	Convert the BYTE, INT, DINT, REAL, or BOOL type into the type specified by <TYPE>.		
<TYPE>			
Operands	Description	Value range	Data type
S	Variable to be converted	-	BYTE/INT/DINT/REAL/ BOOL

The following <TYPE> options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	√	√	√	√	-	-	-

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S	√	√	√	√	√	-	√	√	√	√	√	-

Function

This instruction is used to explicitly convert the variable of the BYTE, INT, DINT, REAL, or BOOL type into the variable of the type specified by < TYPE>. Specifically:

- S is the variable to be converted.

- The return value is the conversion result.

Example

LiteST Code	Result
bool0 := TO_BOOL(11.2)	TRUE
byte0 := TO_BYTE(15)	15
int0 := TO_INT(11.2)	11
dint0 := TO_DINT(TRUE)	1
real0 := TO_REAL(11)	11.0

5.7.1.40 ToAryByte

Return Value Type	Conversion from variable to byte array		
BOOL			
Operand	Description	Range	Data Type
S1	Data to be converted	-	BOOL, BYTE, INT, DINT, REAL, STRING, STRUCT, and array of these types
S2	Conversion order	0 or 1	BYTE, INT, and DINT
S3	Conversion result array	-	BYTE array
S4	Number of elements in the conversion result	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	Byte	INT	DINT	REAL	BOOL	STRING	
S1	√	√	√	√	√	√	-	-	-	-	-	-	-	-

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	Byte	INT	DINT	REAL	BOOL	STRING	
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S3	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S4				√	√									

Function and Instruction Description

- Function description
  - The ToAryByte instruction splits the data to be converted S1 by 1 byte, stores the split data in the conversion result array starting from S3[0] in turn, and stores the number of elements after conversion in S4.
  - The conversion order is specified by S2. The following table describes the meanings of values of S2.

S2 Value	Meaning
0	_LOW_HIGH: Lower byte first, higher byte last
1	_HIGH_LOW: Higher byte first, lower byte last

- Conversion rules

Byte Size by S1 Type	S1 Type	S2 Conversion Order	Array	Conversion Rule
2 bytes and more	◦ INT, DINT, and REAL ◦ String of more than 2 bytes ◦ Array with total size of all elements over 2 bytes, one element in the array of over 2 bytes, structure with the total size of all structure elements over 2 bytes, or one structure element in the structure of over 2 bytes	_LOW_HIGH or _HIGH_LOW	Yes or No	See rule 1.
1 byte	◦ Byte ◦ String of 1 byte ◦ Array with total size of all elements as 1 byte, one element in the array of 1 byte, structure with the total size of all structure elements as 1 byte, or one member in the structure of 1 byte	_LOW_HIGH	No	Store S1 in S3[0].
			Yes	Store S1[i] in S3[i].
		_HIGH_LOW	No	Store the S1 value in S3[1] and 0x00 in S3[0].

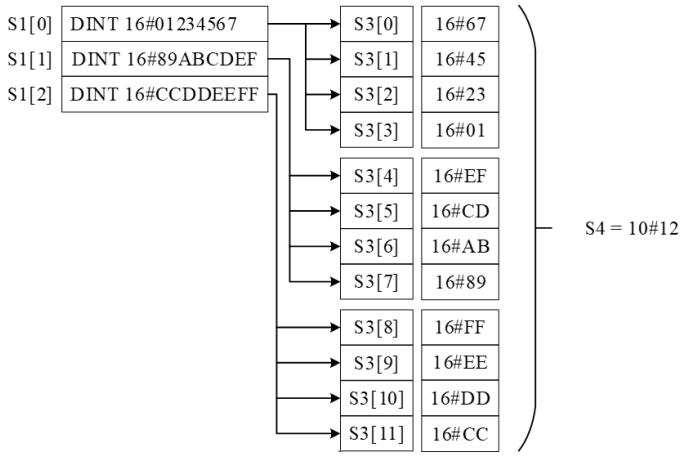
Byte Size by S1 Type	S1 Type	S2 Conversion Order	Array	Conversion Rule
			Yes	See rule 2.
1 bit	BOOL	_LOW_HIGH	No	Store the logic OR of the S1 value and 0x00 in S3[0].
			Yes	See rule 3.
		_HIGH_LOW	No	Store the logic OR of the S1 value and 0x00 in S3[1] and 0x00 in S3[0].
			Yes	See rule 4.

◦ Rule 1

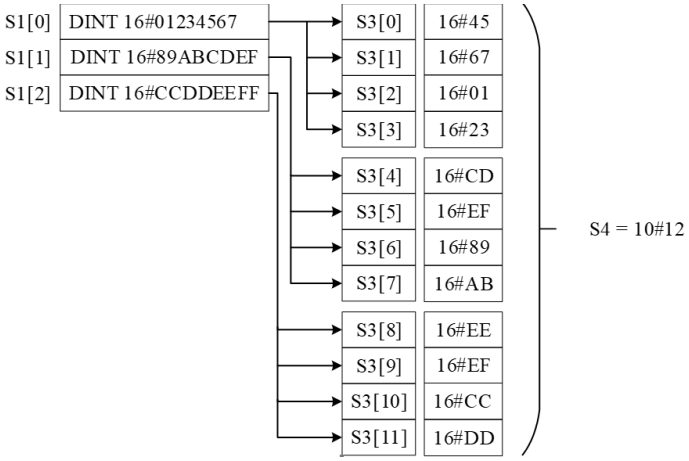
1. Split the S1 value into multiple words by word (two bytes).
2. Split the word with least significant that is obtained from the previous step by byte. If the S2 value is _LOW_HIGH, store the lower bytes in S3[0] and higher bytes in S3[1]. If the S2 value is _HIGH_LOW, store the higher bytes in S3[0] and lower bytes in S3[1].
3. Split the next word by byte in the same way and store the obtained data in S3[2] and S3[3].
4. Repeat the previous steps until the S1 value is processed. If S1 is array, the process ends when the last element in the S1 value is processed.

Example

- The data type of S1 is DINT, the number of elements is 3, and the conversion order is _LOW_HIGH.



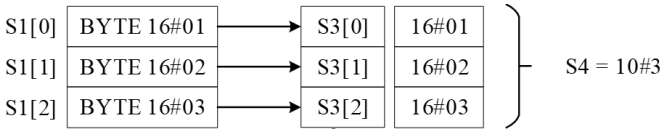
- The data type of S1 is DINT, the number of elements is 3, and the conversion order is _HIGH_LOW.



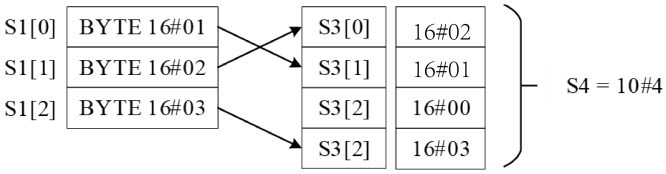
◦ Rule 2: Store S1[i] (i is an odd number) in S3[i+1] and S1[i] (i is an even number) in S3[i-1]. If the number of elements (n) in S1[ ] is an odd number, store 0x00 in S3[n-1].

Example

- The data type of S1 is SINT, the number of elements is 3, and the conversion order is _LOW_HIGH.



- The data type of S1 is SINT, the number of elements is 3, and the conversion order is _HIGH_LOW.



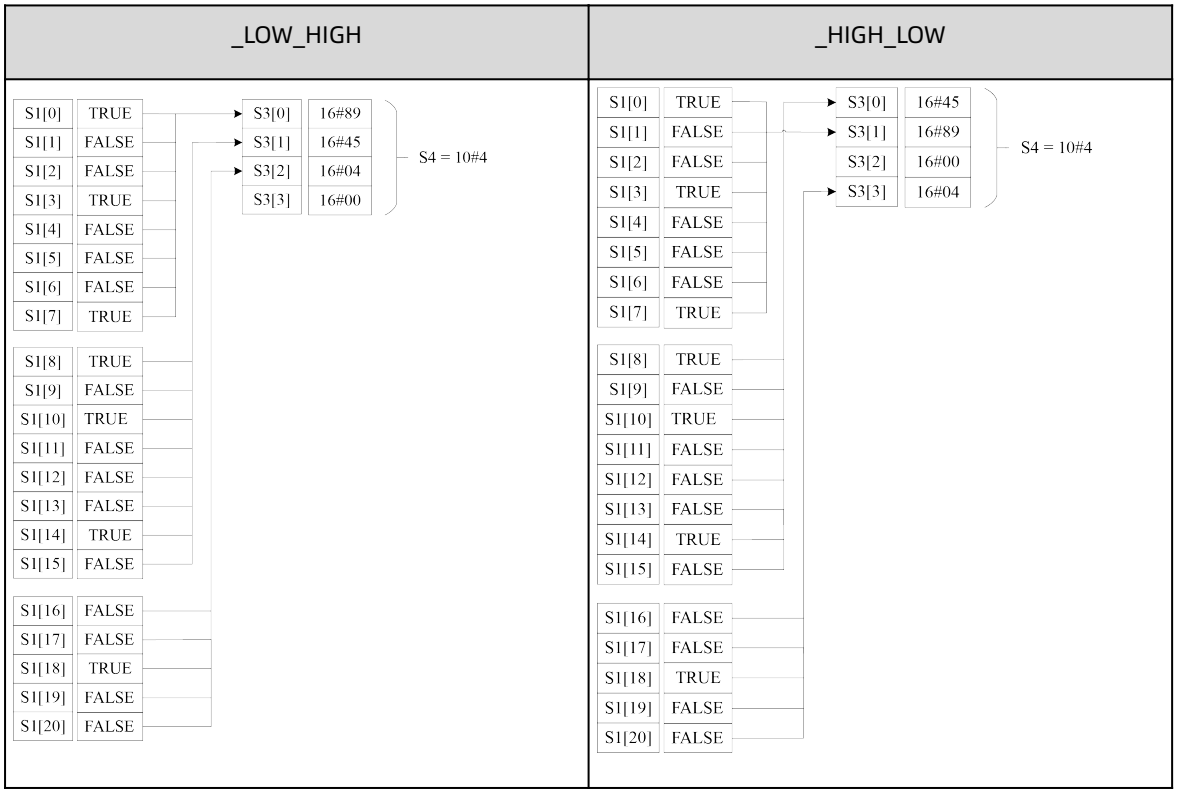
◦ Rule 3

1. Sort the values in S1[0] to S1[7] into a byte and store it in S3[0].
2. Sort the values in S1[8] to S1[15] into a byte and store it in S3[1].
3. Repeat the previous steps to process other data. If S1[] contains less than 8 bits, supplement the insufficient bits by FALSE.
4. If the S4 value is always an even number, supplement the insufficient bits by FALSE.

◦ Rule 4

1. Sort the values in S1[0] to S1[7] into a byte and store it in S3[1].
2. Sort the values in S1[8] to S1[15] into a byte and store it in S3[0].
3. Repeat the previous steps to process other data. If S1[] contains less than 8 bits, supplement the insufficient bits by FALSE.
4. If the S4 value is always an even number, supplement the insufficient bits by FALSE.

Example: The data type of S1 is BOOL and the number of elements is 21. The following table describes the conversion rules when the conversion order is _LOW_HIGH or _HIGH_LOW.



Instruction Example

Item	ST

Defined variable

NO.	Variable Name	Data Type	Initial Value
1	var0	DINT	0
2	order0	INT	0
3	ary0	BYTE[10]	...
14	size0	INT	0
15	Return_Bool	BOOL	OFF
16			

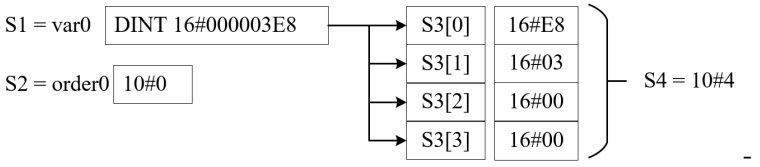
Program

```
Return_Bool := ToAryByte(var0,
                         order0,
                         ary0[0],
                         size0);
```

Operation result

Element Name	Data Type	Display Format	Current Value
var0	DINT	Hex	0x3E8
order0	INT	Dec	0
ary0	BYTE[10]		
ary0[0]	BYTE	Hex	0xE8
ary0[1]	BYTE	Hex	0x3
ary0[2]	BYTE	Dec	0
ary0[3]	BYTE	Dec	0
ary0[4]	BYTE	Dec	0
ary0[5]	BYTE	Dec	0
ary0[6]	BYTE	Dec	0
ary0[7]	BYTE	Dec	0
ary0[8]	BYTE	Dec	0
ary0[9]	BYTE	Dec	0
size0	INT	Dec	4

Principle



NOTE

- S1 must be a variable.
- When S1 is String, it cannot be converted to a numerical value. The content of the variable is taken as a bit string and converted to a byte array.
- When the S2 value is 0 and S1 contains an odd number of bytes, 0x00 must be added to the end of S1 to make an even number of bytes before the conversion is started.
- When the conversion result exceeds the array area of S3[ ], the return value is FALSE, S4 is 0, and S3[ ] remains unchanged.

5.7.2 Data Transfer and Comparison

5.7.2.1 List of Instructions

The following table lists the data transfer and comparison instructions.

Category	Instruction	Function	Language Support
Data transfer and comparison instructions	MOV	Move	LD
	EMOV	Binary floating-point move	LD
	SMOV	Shift move	LD
	BMOV	Batch move	LD
	FMOV	Multi-point move	LD
	CML	Complement	LD
	CMP	Comparison	LD
	ECMP	Floating-point comparison	LD
	ZCP	Zone comparison	LD
	EZCP	Floating-point zone comparison	LD

5.7.2.2 MOV

The MOV instruction copies the data at the source address S to the destination address.

MOV: Move

16-bit instruction	MOV: Continuous execution MOVP: Pulse execution			
32-bit instruction	DMOV: Continuous execution DMOVP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Data to be moved, or address of the word element that stores the data	-	INT/ DINT
D	Destination to which data is copied	Address of the word element to which data is copied	-	INT/ DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	

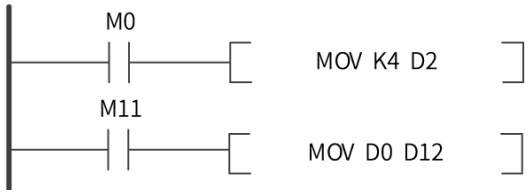
Function and Instruction Description

The MOV instruction requires contact driving and has two operands. It copies the value in S to D.

When the 32-bit instruction (DMOV) is executed, S and D use adjacent variable units with high-numbered addresses.

For example, when the statement [DMOV D1 D5] is executed, data in D1 is moved to D5, and data in D2 is moved to D6.

Example



When X0 is ON, K4 is assigned to D2. When X0 switches from ON to OFF, K4 in D2 remains unchanged unless the value of D2 is changed again by the user program.

When the PLC is powered on again, the value of D2 becomes 0. The value of a register that is retentive upon power failure remains unchanged when the PLC is powered on or starts running after stop.

5.7.2.3 EMOV

The EMOV instruction transfers binary floating-point numbers. This instruction requires contact driving. When it is executed, the binary floating-point data in S is copied to D.

EMOV: Floating-point move

16-bit instruction	-			
32-bit instruction	DEMOV: Continuous execution DEMOVP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Source from which a binary floating-point number is transferred	-	REAL
D	Transfer destination	Unit that stores the binary floating-point number transferred to a destination	-	REAL

For soft elements, see the following table.

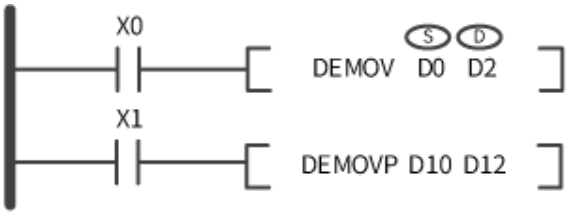
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function

The EMOV instruction transfers binary floating-point numbers. Contact driving is required. After the instruction is executed, the value of the binary floating-point number in S is copied to D. Specifically:

- S is the source from which a binary floating-point number is transferred.
- D is the unit that stores the binary floating-point number transferred to a destination.

Example



Assume that the binary floating-point number in (D1, D0) is 12.3456. When X0 is ON, the binary floating-point number in (D3, D2) becomes 12.3456. When X0 switches from ON to OFF, 12.3456 in (D3, D2) remains unchanged, unless the value of (D3, D2) is changed again by the user program, or the PLC is powered on again or starts running after stop. The value of a register that is retentive upon power failure remains unchanged when the PLC is powered on or starts running after stop.

5.7.2.4 BMOV

When the driving conditions are met, the BMOV instruction transfers the data of n registers in addresses starting from S to the n registers in addresses starting from D.

BMOV: Batch data transmission

16-bit instruction	BMOV: Continuous execution BMOVP: Pulse execution			
32-bit instruction	DBMOV: Continuous execution DBMOVP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source start address	Start address of word soft elements that store the data to be transferred in batches	-	INT/DINT, array*n
D	Transfer destination start address	Start address of word soft elements that store the data arriving at a destination	-	INT/DINT, array*n
n	Data length	Number of word soft elements whose data will be transferred in batches	1 to 512	INT/DINT

The list of soft elements is shown as follows.

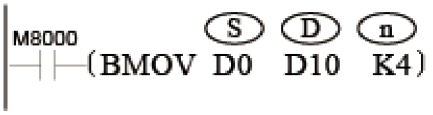
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The BMOV instruction requires contact driving and has three operands. It copies the values of n variables in addresses starting from S to n units in addresses starting from D.

n ranges from 1 to 512.

Example



The operations are as follows:

D0→D10

D1→D11

D2→D12

D3→D13

5.7.2.5 SMOV

The SMOV instruction transfers m2 bits starting from the m1th bit in S to m2 bits starting from the nth bit in D.

SMOV: Shift move

16-bit instruction	SMOV: Continuous execution SMOVP: Pulse execution			
32-bit instruction	DSMOV: Continuous execution DSMOVP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Data source	Address of the word soft element that stores the bits to be transferred	-	INT/ DINT
m1	Initial bit to be transferred	Position of the initial bit in S to be transferred	1-4/1-8	INT/ DINT
m2	Transferred bit count	Number of bits in S to be transferred	1-m1	INT/ DINT
D	Destination device	Address of the word soft element that stores the bits transferred to a destination	-	INT/ DINT
n	Initial bit at the destination	Position of the initial bit transferred to D	m2-4/ m2-8	INT/ DINT

See the following table for the soft elements.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
m1	-	-	-	√	√	√	√	-	-
m2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

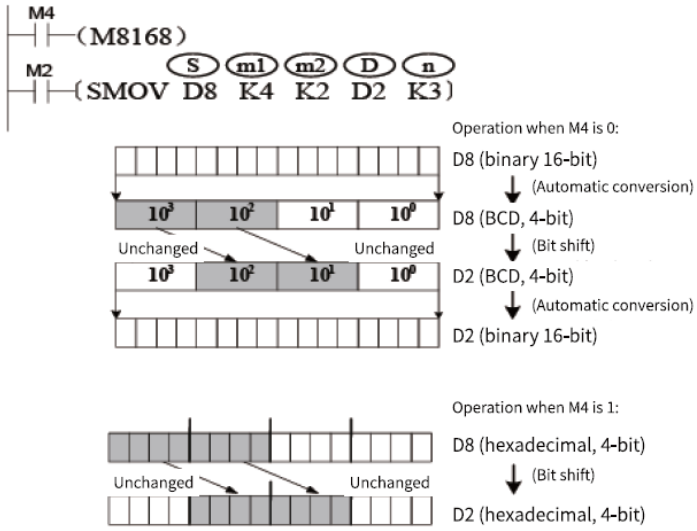
Function

The SMOV instruction requires contact driving and has a maximum of five operands, which are described as follows:

- S is the data source variable. When M8168 is OFF, the BCD mode (decimal bits) is used. The S operand ranges from 0000 to 9999/00000000 to 99999999 and cannot be negative. When M8168 = ON, the BIN mode is enabled. The S operand can be a negative number.
- m1 is the start position of bits to be transferred. It ranges from 1 to 4/1 to 8.
- m2 is the number of bits to be transferred. The value range is 1 to m1.
- D is the destination variable to which bits are transferred.
- n is the start bit in the destination variable that stores transferred data. It ranges from m2 to 4/m2 to 8.

The data bit transfer process is related to the status of the special flag M8168. When M8168 = OFF, the BCD mode (decimal bits) is enabled When M8168 = ON, the BIN mode is enabled, whereby every four bits (hexadecimal) are transferred at a time as a whole unit.

Example



Assume that D8 = K1234 and D2 = K5678. When M8168 = OFF (BCD mode enabled), the value in D2 changes to K5128 if M2 is set to ON.

When M8168 = ON (BIN mode enabled) with D8 = H04D2 = K1234 and D2 = H162E = K5678, then D2 = H104E = K4174 if M2 is set to ON.

5.7.2.6 FMOV

When the driving conditions are met, the FMOV instruction transfers the data in S to n registers starting from the address specified in D.

FMOV: Multi-point move

16-bit instruction	FMOV: Continuous execution FMOVP: Pulse execution			
32-bit instruction	DFMOV: Continuous execution DFMOVP: Pulse execution			
Operands	Name	Description	Value range	Data type

S	Data source	Data to be transferred, or address of the word element that stores the data	-	INT/DINT
D	Transfer destination start address	Start address of word soft elements that store the data arriving at a destination	-	INT/DINT, array*n
n	Target number	Number of word soft elements that store the data arriving at a destination	1 to 512	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

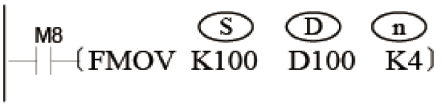
Function

The FMOV instruction requires contact driving and has three operands. It copies the data in S to n units starting from the address specified in D.

The value of n ranges from 1 to 512.

FMOV is a 16-bit multi-point transfer instruction, whereas DFMOV is 32-bit.

Example



The operations are as follows when M8 is set to ON:

K100→D100

K100→D101

K100→D102

K100→D103

5.7.2.7 CML

The CML instruction inverts the data in S by bit and transfers the inverted data to D.

CML: Complement

16-bit instruction	CML: Continuous execution CMLP: Pulse execution

32-bit instruction	DCML: Continuous execution DCMLP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Inverted data source	Data to be inverted, or address of the word soft element that stores the data	-	INT/DINT
D	Transfer destination	Address of the word soft element that stores the transferred inverted data	-	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The CML instruction requires contact driving and has two operands. It inverts the bit pattern of the BIN value in S and copies the resulting data to D.

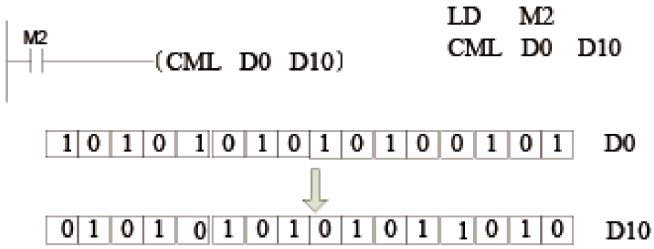
When the number of bits in D is less than 16, the inverted data is aligned by low-order bits and then transferred to D.

When the 32-bit instruction (DCML) is executed, S and D use adjacent variable units with high-numbered addresses.

For example, when the statement [DCML D1 D5] is executed, data in /D1 is moved to D5, and data in /D2 is moved to D6.

Example

Ladder chart and instruction list



5.7.2.8 CMP

CMP -- Comparison

16-bit Instruction	CMP: Continuous execution/CMPP: Pulse execution			
32-bit Instruction	DCMP: Continuous execution/DCMPP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Comparand 1	Data of comparand 1, or address of the word soft element that stores the data	-	INT and DINT
S2	Comparand 2	Data of comparand 2, or address of the word soft element that stores the data	-	INT and DINT
D	Comparison result	Start address of three consecutive bits that store the comparison result (ON or OFF)	-	BOOL, array*3

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	√[1]	√	-	-	-	√	-	-	-

NOTE

[1]: The X element is not supported.

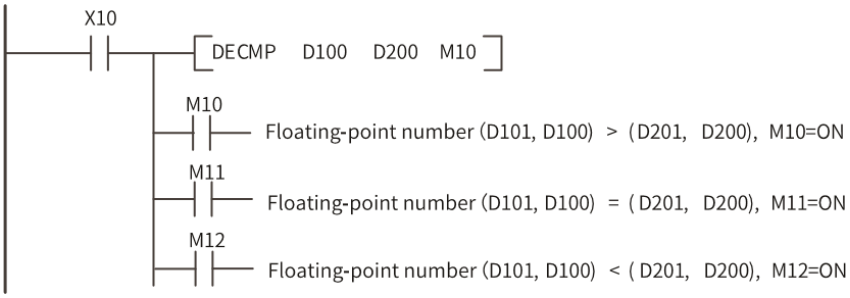
Function and Instruction Description

The CMP instruction compares the size of two operands and outputs the comparison result to the specified bit variable. The operands are processed as signed numbers in algebraic comparison.

When the driving conditions are met, the CMP instruction compares the size of S1 and S2 and sets a bit element among D, D+1, and D+2 to ON based on the comparison result (S1 > S2, S1 = S2, or S1 < S2).

D is a bit variable that occupies three consecutive addresses.

Instruction Example



When X10 is ON, one among M10 to M12 is set to ON.

When X10 switches from ON to OFF and the CMP instruction is not executed, M10 to M12 remain in the state just before X10 switches to OFF.

To clear the comparison result of M10 to M12, use the RST or ZRST instruction.

To obtain the results of ≥, ≤, or ≠, connect M10 to M12 in series or in parallel.

5.7.2.9 ECMP

ECMP -- Floating-point comparison

16-bit Instruction	-			
32-bit Instruction	DECMP: Continuous execution/DECMPP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Comparand 1	Binary floating-point number 1 to be compared	-	REAL
S2	Comparand 2	Binary floating-point number 2 to be compared	-	REAL
D	Comparison result	Comparison result storage unit, which occupies three (bit) variables	-	BOOL, array*3

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	√[1]	√	-	-	-	√	-	-	-

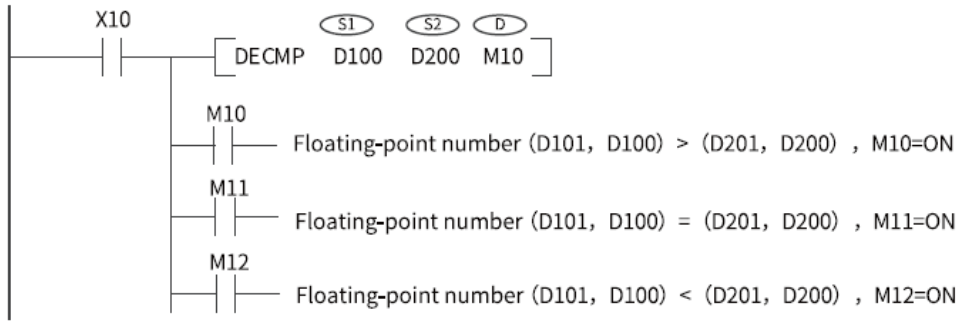
NOTE

[1]: The X element is not supported.

Function and Instruction Description

The ECMP instruction compares two floating-point variables and outputs the comparison result to first three variables.

Instruction Example



When X10 is ON, one among M10 to M12 is set to ON.

When X10 switches from ON to OFF and the DECMP instruction is not executed, M10 to M12 remain in the state just before X10 switches from ON to OFF. To clear the comparison result indicated by M10 to M12, use the RST or ZRST instruction.

To obtain the results of ≥, ≤, or ≠, connect M10 to M12 in series or in parallel.

5.7.2.10 ZCP

When the driving conditions are met, the ZCP instruction compares S with S1 and S2 and sets a bit element among D, D+1, and D+2 to ON based on the comparison result (S < S1, S1≤S≤S2, S > S2).

ZCP: Zone comparison

16-bit instruction	ZCP: Continuous execution ZCPP: Pulse execution			
32-bit instruction	DZCP: Continuous execution DZCPP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Lower limit for range comparison	Data, or address of the word soft element that stores the data	-	INT/DINT
S2	Upper limit for range comparison	Data, or address of the word soft element that stores the data	-	INT/DINT
S	Comparative variable	Data, or address of the word soft element that stores the data	-	INT/DINT
D	Comparison result	Start address of three consecutive components that store the comparison result (ON or OFF)	-	BOOL, array*3

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S	-	-	-	√	√	√	√	-	-
D	√[1]	√	-	-	-	√	-	-	

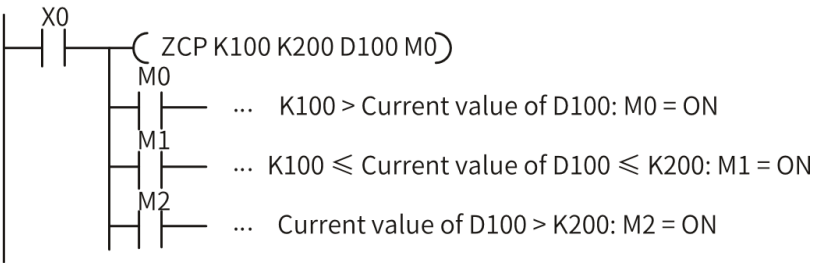
NOTE

[1] The X element is not supported.

Function

The ZCP instruction requires contact driving and has four operands. When the control flow is active, use this instruction to algebraically compare operands as signed numbers. The comparison result is indicated by the position of S within the range defined by S1 and S2 and stored in three consecutive bit variables from start address D.

Example



When X0 is set to ON, one of M0 to M2 switches to ON.

When X0 switches from ON to OFF and the ZCP instruction is not executed, M0 to M2 remain in the state just before X0 switches from ON to OFF.

To clear the comparison result indicated by M0 to M2, use the RST or ZRST instruction.

5.7.2.11 EZCP

The EZCP instruction compares a binary floating-point variable with the upper and lower limits of a floating-point variable zone and outputs the comparison result to three variables starting from D.

EZCP: Floating-point zone comparison

16-bit instruction	-
	DEZCP: Continuous execution DEZCPP: Pulse execution

32-bit instruction				
Operands	Name	Description	Value range	Data type
S1	Lower limit for range comparison	Lower limit of a binary floating-point variable range	-	REAL
S2	Upper limit for range comparison	Upper limit of a binary floating-point variable range	-	REAL
S	Comparand	Binary floating-point variable to be compared	-	REAL
D	Comparison result	Comparison result storage unit, which occupies three (bit) variables	-	BOOL, array*3

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
S	-	-	-	√	√	√	-	√	-
D	√[1]	√	√	-	-	√	-	-	

NOTE

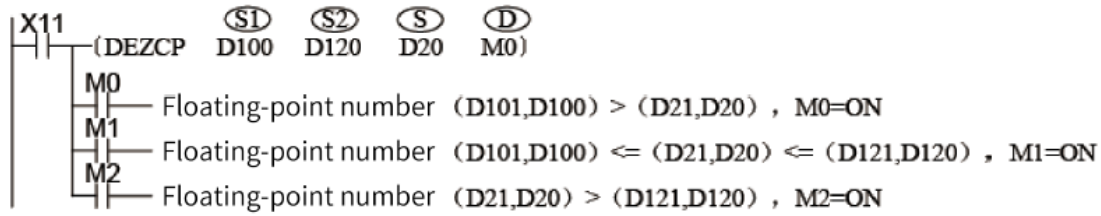
[1] The X element is not supported.

Function

The EZCP instruction compares a binary floating-point variable with the upper and lower limits of a binary floating-point variable zone and outputs the comparison result to three variables starting from D. Specifically:

- S1 is the lower limit of the binary floating-point variable zone.
- S2 is the upper limit of the binary floating-point variable zone.
- S is the binary floating-point variable to be compared.
- D is the comparison result storage unit, which occupies three (bit) variables.

Example



When X11 is set to ON, one of M0 to M2 is ON.

When X11 switches from ON to OFF and the DEZCP instruction is not executed, M0 to M2 remain in the state just before X11 switches from ON to OFF.

To clear the comparison result indicated by M0 to M2, use the RST or ZRST instruction.

5.7.3 Table Operation

5.7.3.1 List of Instructions

The following table lists the table management instructions.

Category	Name	Description	Language Support
Table management instructions	SORTR	Data sorting by row	LD
	SORTC	Data sorting by column	LD
	SER	Data search	LD
	FDEL	Deletion of data from data table	LD
	FINS	Insertion of data to data table	LD
	POP	Last-in data read	LD
	RAMP	Ramp	LD

5.7.3.2 SORTR

When the driving conditions are met, the SORTR instruction sorts a data table (starting from address S) with m1 rows and m2 columns in ascending order according to the data of the nth row, and stores the sorting result in a data table starting from address D1.

SORTR: Data sorting by row

16-bit instruction	SORTR: Continuous execution
32-bit instruction	DSORTR: Continuous execution

Operands	Name	Description	Value range	Data type
S	Data to be sorted	Start address of the array to be sorted	-	INT/DINT, array*m1*m2
m1	Number of lines	Total number of rows	1 to 32	INT/DINT
m2	Column count	Total number of columns	1 to 32	INT/DINT
D1	Sorting result	Data sorting result	-	INT/DINT, array*m1*m2
n	Reference row for sorting	Number of the row which serves as the reference for sorting	1 to m1	INT/DINT
D2	Sorting process data	Process data during sorting	-	INT/DINT

The list of soft elements is shown as follows.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
m1	-	-	-	√	√	√	√	-	-
m2	-	-	-	√	√	√	√	-	-
D1	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-
D2	-	-	-	√	√	√	-	-	-

Function

This instruction sorts an array of m1 rows and m2 columns (starting from address S) based on data of the nth row. The sorting result will then be stored in a variable area starting from D1. Specifically:

- S is the start unit for the first variable in the first row (or record).
- m1 is the number of array rows (or records).
- m2 is the number of array columns, or the number of columns in every record.
- D1 is the start unit for storing the sorted data. The number of subsequent variable units occupied is the same as the number of array variables before the sorting.
- n is the number of the array row that serves as the reference for sorting. n ranges from 1 to m1.
- D2 stores the sorting process data, ranging from 0 to (m2 – 1).

The following shows the sorting process of a data table which occupies 3 x 3 points.

Before sorting:

Row number	Column number		
	1	2	3
1	S	S+3	S+6
	1	2	8
2	S+1	S+4	S+7
	6	7	2
3	S+2	S+5	S+8
	3	4	3

After sorting (sorted in ascending order based on the second row)

Row number	Column number		
	1	2	3
1	D	D+3	D+6
	8	1	2
2	D+1	D+4	D+7
	2	6	7
3	D+2	D+5	D+8
	3	3	4

The ON/OFF state of M8165 determines the sorting order. When M8165 is ON, sorting is in descending order; when M8165 is OFF, sorting is in ascending order.

Data sorting starts when the instruction flow becomes active. After m1 scan cycles, the sorting is completed, and the instruction completion flag, M8029, is set to ON.

Note

- Do not modify the operands during the execution of the SORTR instruction.
- Switch the flow from OFF to ON before you execute the instruction for the second time.
- Keep the operands and data unchanged during execution.
- Data in S and D1 can overlap completely or be staggered, but they cannot overlap partially. Otherwise, error 6705 will occur.
- The 32-bit instruction is used in the same way as the 16-bit instruction. The operands occupy two 16-bit soft elements.

Example



When M520 is set to ON, the flow of the SORTR instruction becomes active. The instruction sorts a 4x4 table starting from D500 in ascending order based on the values in the second row. The sorting result is stored in a table starting from D600, and the process data is stored in D720.

Once the sorting is completed, the flag M8029 will be set to ON. However, if there are multiple sorting instructions, the value of M8029 will be overwritten by the subsequent sorting instructions.

- Before sorting

	Element Name	Data Type	Display Format	Current Value
1	D500	INT	Dec	1
2	D501	INT	Dec	2
3	D502	INT	Dec	3
4	D503	INT	Dec	2
5	D504	INT	Dec	6
6	D505	INT	Dec	4
7	D506	INT	Dec	8
8	D507	INT	Dec	7
9	D508	INT	Dec	3
10	D509	INT	Dec	1
11	D510	INT	Dec	2
12	D511	INT	Dec	3
13	D512	INT	Dec	2
14	D513	INT	Dec	6
15	D514	INT	Dec	4
16	D515	INT	Dec	8

- After sorting:

18	D600	INT	Dec	2
19	D601	INT	Dec	1
20	D602	INT	Dec	2
21	D603	INT	Dec	3
22	D604	INT	Dec	4
23	D605	INT	Dec	6
24	D606	INT	Dec	7
25	D607	INT	Dec	8
26	D608	INT	Dec	1
27	D609	INT	Dec	3
28	D610	INT	Dec	3
29	D611	INT	Dec	2
30	D612	INT	Dec	6
31	D613	INT	Dec	2
32	D614	INT	Dec	8
33	D615	INT	Dec	4

5.7.3.3 SORTC

When the driving conditions are met, the SORTC instruction sorts a data table (starting from address S) with m1 rows and m2 columns in ascending order according to the data of the nth column, and stores the sorting result in a data table starting from address D.

SORTC: Data sorting by column

16-bit instruction	SORTC: Continuous execution
32-bit instruction	DSORTC: Continuous execution

Operands	Name	Description	Value range	Data type
S	Data to be sorted	Start address of the array to be sorted	-	INT/DINT
m1	Number of lines	Total number of rows	1 to 32	INT/DINT
m2	Column count	Total number of columns	1 to 32	INT/DINT
D1	Sorting result	Data sorting result	-	INT/DINT
n	Reference column for sorting	Number of the column which serves as the reference for sorting	1 to m2	INT/DINT
D2	Sorting process data	Process data during sorting	-	INT/DINT

The list of soft elements is shown below.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
m1	-	-	-	√	√	√	√	-	-
m2	-	-	-	√	√	√	√	-	-
D1	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-
D2	-	-	-	√	√	√	-	-	-

Function

This instruction sorts an array of m1 rows and m2 columns (starting from address S) based on data of the nth column. The sorting result will then be stored in a variable area starting from D1. Specifically:

- S is the start unit for the first variable in the first row (or record).
- m1 is the number of array rows (or records).
- m2 is the number of array columns, or the number of columns in every record.
- D1 is the start unit for storing the sorted data. The number of subsequent variable units occupied is the same as the number of array variables before the sorting.
- n is the number of the array column whose data will be sorted. n ranges from 1 to m2.
- D2 stores the sorting process data, ranging from 0 to (m1 – 1).

The ON/OFF state of M8165 determines the sorting order. When M8165 = ON, sorting is in descending order; when M8165 = OFF, sorting is in ascending order.

Data sorting starts when the instruction flow becomes active. After m1 scan cycles, the sorting is completed, and the instruction completion flag, M8029, is set to ON.

The following shows the sorting process of a data table which occupies 3 x 3 point.

Before sorting:

Row number	Column number		
	1	2	3
1	S	S+3	S+6
	1	2	8
2	S+1	S+4	S+7
	2	6	7
3	S+2	S+5	S+8
	3	4	3

After sorting (sorted in ascending order based on the second column)

Row number	Column number		
	1	2	3
1	D	D+3	D+6
	1	2	8
2	D+1	D+4	D+7
	3	4	3
3	D+2	D+5	D+8
	2	6	7

Note

- Do not modify the operands during the execution of the SORTR instruction.
- Switch the flow from OFF to ON before you execute the instruction for the second time.
- Keep the operands and data unchanged during execution.
- Data in S and D1 can overlap completely or be staggered, but they cannot overlap partially. Otherwise, error 6705 will occur.
- The 32-bit instruction is used in the same way as the 16-bit instruction. The operands occupy two 16-bit soft elements.

Example



When M500 is set to ON, the flow of the SORTC instruction becomes active. The instruction sorts a 4x4 table starting from D500 in ascending order based on the values in the second column. The sorting result is stored in a table starting from D600, and the process data is stored in D700.

Once the sorting is completed, the flag M8029 will be set to ON. However, if there are multiple sorting instructions, the value of M8029 will be overwritten by the subsequent sorting instructions.

Before sorting:

	Element Name	Data Type	Display Format	Current Value
1	D500	INT	Dec	1
2	D501	INT	Dec	2
3	D502	INT	Dec	3
4	D503	INT	Dec	2
5	D504	INT	Dec	6
6	D505	INT	Dec	4
7	D506	INT	Dec	8
8	D507	INT	Dec	7
9	D508	INT	Dec	3
10	D509	INT	Dec	1
11	D510	INT	Dec	2
12	D511	INT	Dec	3
13	D512	INT	Dec	2
14	D513	INT	Dec	6
15	D514	INT	Dec	4
16	D515	INT	Dec	8

After sorting:

18	D600	INT	Dec	3
19	D601	INT	Dec	1
20	D602	INT	Dec	2
21	D603	INT	Dec	3
22	D604	INT	Dec	1
23	D605	INT	Dec	2
24	D606	INT	Dec	3
25	D607	INT	Dec	2
26	D608	INT	Dec	6
27	D609	INT	Dec	4
28	D610	INT	Dec	8
29	D611	INT	Dec	7
30	D612	INT	Dec	2
31	D613	INT	Dec	6
32	D614	INT	Dec	4
33	D615	INT	Dec	8

5.7.3.4 SER

When the driving conditions are met, the SER instruction searches n data entries starting from source address S1 to find the address of the data compliant with the condition specified by S2 and stores the result in five consecutive registers starting from D.

SER: Data search

16-bit instruction	SER: Continuous execution SERP: Pulse execution			
32-bit instruction	DSER: Continuous execution DSERP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Search start address		-	

		Start address of the data (n consecutive registers) to be searched		INT/DINT, array*n
S2	Compared data	Compared data, or address of the word soft element that stores the data	-	INT/DINT
D	Search result storage start address	Start address of word soft elements that store the search result	-	INT/DINT, array*5
n	Searched data count	Number of searched data entries	1 to 256	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	√		
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

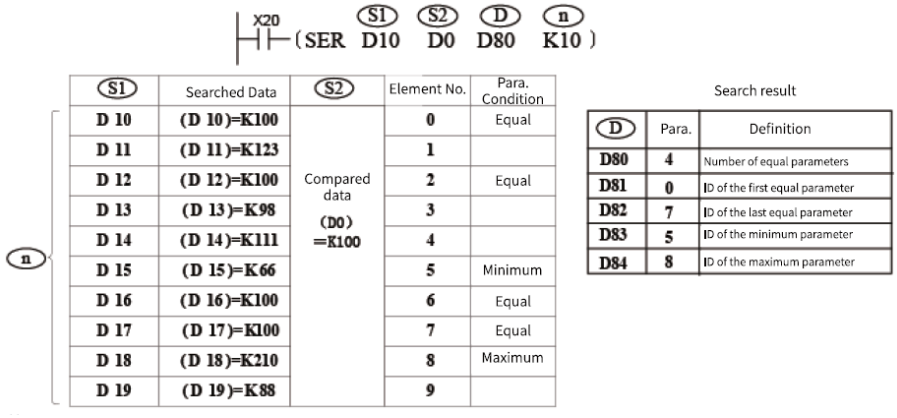
The SER instruction searches a defined data stack for the units with the same data as the data for comparison as well as the maximum and minimum values. Specifically:

- S1 is the start address of the searched data stack.
- S2 is the data to be searched for.
- D is the start address of search result storage.
- n is the length of the searched data area. It ranges from 1 to 256.

In 32-bit operation, S1, S2, and D point to 32-bit variables, and n is calculated based on 32-bit variable width.

When the driving conditions are met, the SER instruction searches k data entries starting from source address S1 to find the address of the data compliant with the condition specified by S2 and stores the result in five consecutive registers starting from D.

Example



Remarks

Comparison is performed only when X20 is ON. Signed numbers are compared algebraically, for example, -8 < +2.

When there are multiple minimum or maximum values, the element with the largest serial number is displayed.

The search result is stored in five consecutive units starting from D. If no equal data exists, the values in D80 to D82 in the preceding example are all 0s.

5.7.3.5 FDEL

The FDEL instruction deletes any data from a table.

FDEL: Deletion of data from a table

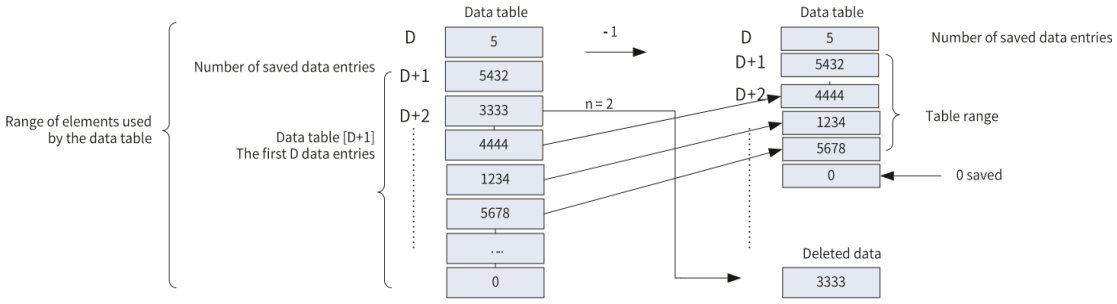
16-bit instruction	FDEL: Continuous execution FDELP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Deleted data	Number of the soft element that stores the data to be deleted	-	INT
D	Data table information	Start number of soft elements that store a data table D: Number of stored data entries D+1: Start position of a data table	-	INT, array*(D+1)
n	Position of deletion	Position in a table at which data is deleted	1 to 256	INT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The FDEL instruction deletes the nth data entry from a table (starting from [D+1]) and stores the deleted data in [S]. The (n+1)th data entry and subsequent ones in the data table are shifted forward one by one, and the number of stored data entries (D) decreases by 1.



An error is returned in the following conditions:

An error is returned in the following conditions:

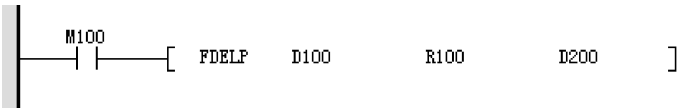
1. The number of stored data entries is out of range.

2. The table position n of the data to be deleted exceeds the number of stored data entries (D).

3. n is less than or equal to 0.

4. The number of stored data entries is less than or equal to 0.

Example



Before execution of the instruction

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	0
R100	16-bit INT	Decimal	5
R101	16-bit INT	Decimal	1111
R102	16-bit INT	Decimal	2222
R103	16-bit INT	Decimal	3333
R104	16-bit INT	Decimal	4444
R105	16-bit INT	Decimal	5555
R106	16-bit INT	Decimal	0
D200	16-bit INT	Decimal	3

After execution of the instruction

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	3333
R100	16-bit INT	Decimal	4
R101	16-bit INT	Decimal	1111
R102	16-bit INT	Decimal	2222
R103	16-bit INT	Decimal	4444
R104	16-bit INT	Decimal	5555
R105	16-bit INT	Decimal	0
R106	16-bit INT	Decimal	0
D200	16-bit INT	Decimal	3

5.7.3.6 FINS

The FINS instruction inserts data at any position in a table.

FINS: Insertion of data to a table

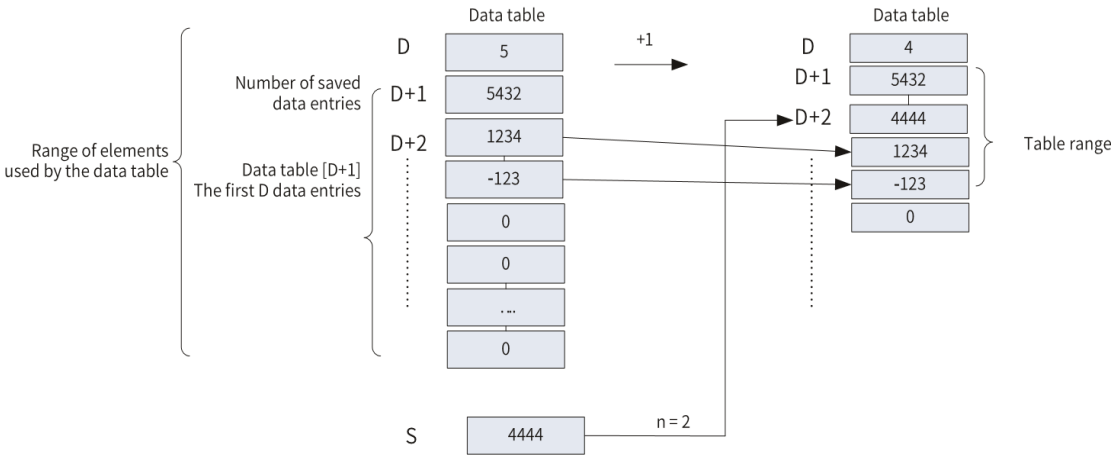
16-bit instruction	FINS: Continuous execution FINSP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Inserted data	Number of the soft element that stores the data to be inserted	-	INT
D	Data table information	Start number of soft elements that store a data table D: Number of stored data entries D+1: Start position of a data table	-	INT, array*(D+2)
n	Position of insertion	Position in a table at which data is inserted	1 to 256	INT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The data stored in [S] is inserted at the nth data entry position in a table from start address [D+1]. The original nth data entry and subsequent ones move backward, and [D] (which indicates the number of stored data entries) is incremented by 1.



An error is returned in the following conditions:

An error is returned in the following conditions:

1. The number of stored data entries is out of range.

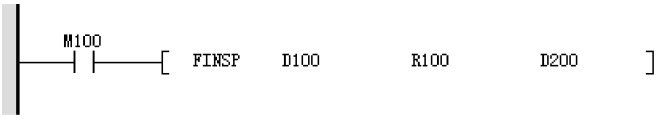
2. The data table range after insertion exceeds the corresponding element range.

3. The position of insertion (n) exceeds the number of stored data entries (D).

4. n is less than or equal to 0.

5. The number of stored data entries is less than or equal to 0.

Example



Before execution of the instruction

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	3333
R100	16-bit INT	Decimal	4
R101	16-bit INT	Decimal	1111
R102	16-bit INT	Decimal	2222
R103	16-bit INT	Decimal	4444
R104	16-bit INT	Decimal	5555
R105	16-bit INT	Decimal	0
R106	16-bit INT	Decimal	0
D200	16-bit INT	Decimal	3

After execution of the instruction

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	3333
R100	16-bit INT	Decimal	5
R101	16-bit INT	Decimal	1111
R102	16-bit INT	Decimal	2222
R103	16-bit INT	Decimal	3333
R104	16-bit INT	Decimal	4444
R105	16-bit INT	Decimal	5555
R106	16-bit INT	Decimal	0
D200	16-bit INT	Decimal	3

5.7.3.7 POP

The POP instruction reads the last data written by a shift write instruction (SFWR) for first in last out (FILO) control.

POP: Last-in data read

16-bit instruction	POP: Continuous execution POPP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Data to be read	Start number of soft elements that store first-in data (including pointer data) S: Pointer data (number of stored data entries) S+1: Data area	-	INT, array*n
D	Stored result	Number of the soft element that stores the last-out data	-	INT
n	Data count	Number of stored data points (Because pointer data is also included, set n to a value plus 1. Value range: 2 ≤ n ≤ 512.)	2 to 512	INT

See the following figure for soft elements.

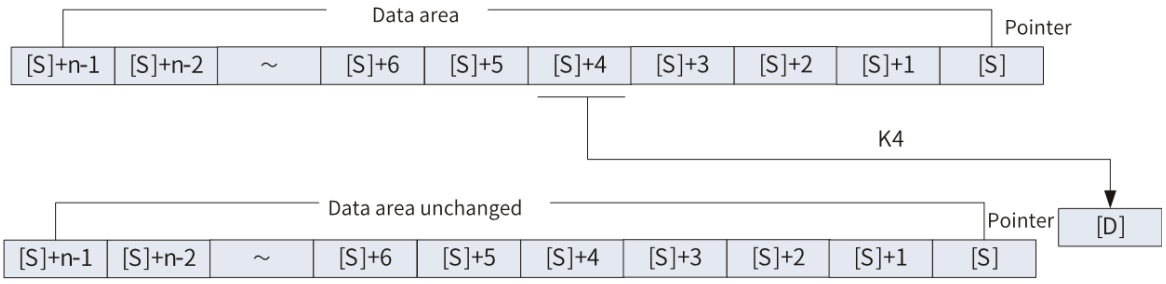
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

For the word soft elements in the range [S-S+n-1], the POP instruction reads the values in components from start address S as well as the offset pointer (pointer data) in [S]. The result is stored in [D]. The pointer in [S] is decremented by 1. n is a number in the range of 2 to 512.

Address	Description
S	Pointer data (number of stored data entries)

Address	Description
[S]+1	Data area (First-in data written by the SFWR instruction)
[S]+2	
[S]+3	
to	
[S]+n-3	
[S]+n-2	
[S]+n-1	



When the pointer in [S] is 0, the zero flag M8020 is set to ON and the POP instruction is not executed.

Check in advance using a comparison instruction whether the current value of [S] satisfies the following condition before executing the POP instruction: 1 ≤ [S] ≤ (n – 1).

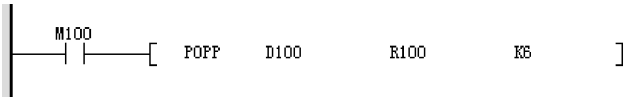
When the current value of the pointer in [S] is 1, 0 is written to [S] and the zero flag M8020 turns ON.

An error is returned in the following conditions:

[S] > n – 1

[S] < 0

Example



Before execution of the instruction

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	4
D101	16-bit INT	Decimal	1111
D102	16-bit INT	Decimal	2222
D103	16-bit INT	Decimal	3333
D104	16-bit INT	Decimal	4444
D105	16-bit INT	Decimal	5555
D106	16-bit INT	Decimal	6666
D107	16-bit INT	Decimal	0
	16-bit INT	Decimal	
R100	16-bit INT	Decimal	0

After execution of the instruction

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	3
D101	16-bit INT	Decimal	1111
D102	16-bit INT	Decimal	2222
D103	16-bit INT	Decimal	3333
D104	16-bit INT	Decimal	4444
D105	16-bit INT	Decimal	5555
D106	16-bit INT	Decimal	6666
D107	16-bit INT	Decimal	0
	16-bit INT	Decimal	
R100	16-bit INT	Decimal	4444

5.7.3.8 RAMP

When the driving conditions are met, the RAMP instruction changes the value in D linearly from S1 to S2 after a number (indicated by n) of scan cycles are completed.

RAMP: Ramp instruction

16-bit instruction	RAMP: Continuous execution			
32-bit instruction	DRAMP: Continuous execution			
Operands	Name	Description	Value range	Data type
S1	Start value	Address of the word soft element that stores the ramp start value	-	INT/DINT
S2	End value	Address of the word soft element that stores the ramp end value	-	INT/DINT
D	Current value	Address of the word soft element that stores the current ramp value	-	INT/DINT, array*2
n	Cycle count	Number of scan cycles required to complete a ramp change	≥1	INT/DINT

For soft elements, see the following table.

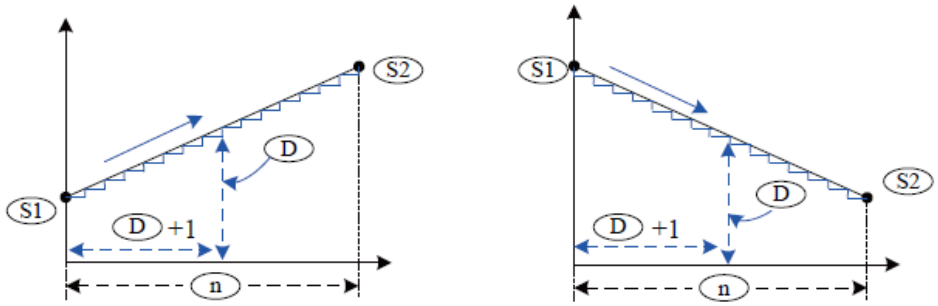
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	√	-	-
n	-	-	-	√	√	√	√	-	-

Function

This instruction is used to perform linear interpolation between two given data entries at a specified time interval. It outputs intermediate values in a sequential manner based on the scanning execution time, until the end value is reached. After each interpolation operation is completed, the M8029 flag is set to ON. Specifically:

- S1 is the start value unit of a ramp signal.
- S2 is the end value unit of a ramp signal.
- D stores the process values of a linear interpolation signal. The interpolation timer is stored in D+1.
- n is the number of program scans required to complete the interpolation. (For a 16-bit instruction, n ranges from 1 to 32767.) Since interpolation output is performed within the normal main loop, set program execution to fixed scan mode to ensure linear output.

Interpolation adopts integer calculation, and the fractional part is discarded. The function of the instruction is shown as follows:



Example



When M40 is ON, interpolation is performed 100 times directly from 1 to 10000. The output results are stored in D30, and the number of interpolations is displayed in D31.

Remarks

Different from that in H3U, the RAMP instruction in H5U only supports single interpolation mode but not continuous interpolation.

5.7.4 Data Shift

5.7.4.1 List of Instructions

The following table lists the data shift instructions.

Category	Instruction	Function	Language Support
Data shift instructions	ROR	Rotation right	LD
	ROL	Rotation left	LD
	RCR	Rotation right with carry	LD

Category	Instruction	Function	Language Support
	RCL	Rotation left with carry	LD
	SFTR	Bit shift right	LD
	SFTL	Bit shift left	LD
	WSFR	Word shift right	LD
	WSFL	Word shift left	LD
	SFWR	Data write (FIFO)	LD
	SFRD	Data read (FIFO)	LD
	SFR	16-bit data shift right with carry by n bits	LD
	SFL	16-bit data shift left with carry by n bits	LD
	SHL	Shift left operation	LiteST
	SHR	Shift right operation	LiteST

5.7.4.2 ROR

When the driving conditions are met, the ROR instruction shifts and rotates the data in D rightwards by n bits. The low-order bits that are rotated out of D fill the high-order bits of D.

ROR: Rotation right

16-bit instruction	ROR: Continuous execution RORP: Pulse execution			
32-bit instruction	DROR: Continuous execution DRORP: Pulse execution			
Operands	Name	Description	Value range	Data type
D	Source data/Target data	Address of the word soft element that stores the data	-	INT/ DINT
n	Number of bit places rotated on every execution	Value range: 1 ≤ n ≤ 16 (16-bit operation); 1 ≤ n ≤ 32 (32-bit operation)	1-16/32	INT/ DINT

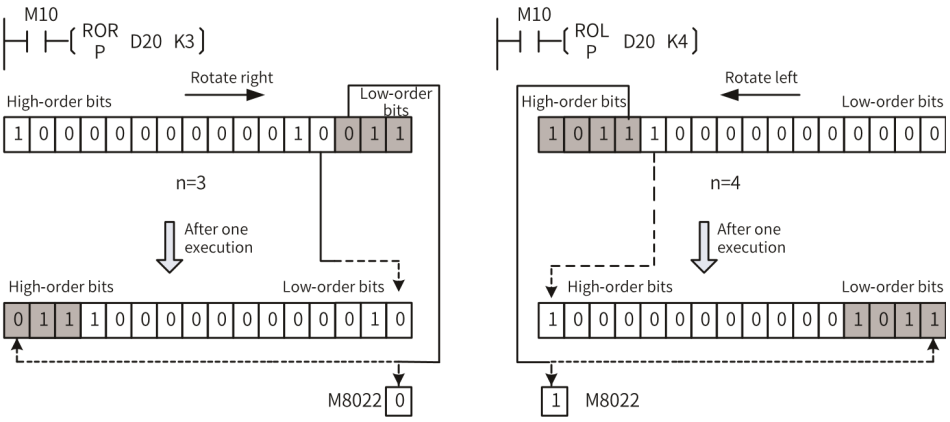
For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The RCR instruction shifts and rotates the data in D with the carry flag M8022 rightwards by n bits. The instruction of the pulse execution type is generally used. When the 32-bit instruction is executed, the register variable occupies two consecutive units.

Example



5.7.4.3 ROL

When the driving conditions are met, the ROL instruction shifts and rotates the data in D leftwards by n bits. The high-order bits that are rotated out of D fill the low-order bits of D.

ROL: Rotation left

16-bit instruction	ROL: Continuous execution ROLP: Pulse execution			
32-bit instruction	DROL: Continuous execution DROLP: Pulse execution			
Operands	Name	Description	Value range	Data type
D	Source data/Target data	Address of the word soft element that stores the data	-	INT/ DINT
n	Number of bit places rotated on every execution	Value range: 1 ≤ n ≤ 16 (16-bit operation); 1 ≤ n ≤ 32 (32-bit operation)	1-16/32	INT/ DINT

For soft elements, see the following table.

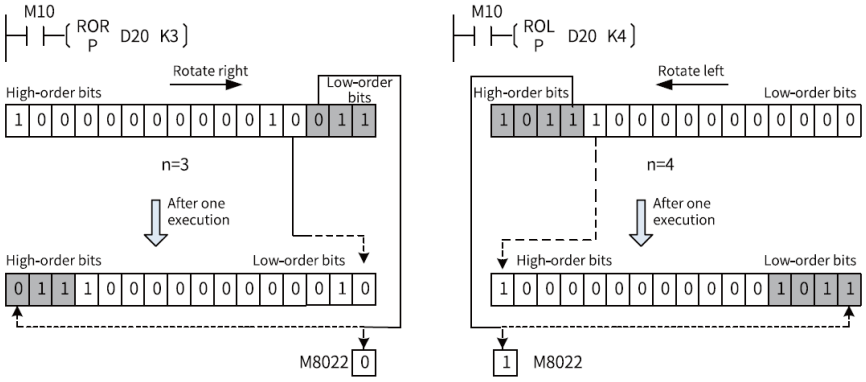
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The ROL instruction shifts and rotates the data in D leftwards by n bits. The instruction of the pulse execution type is generally used. When the 32-bit instruction is executed, the register variable occupies two consecutive units.

The final bit is stored in the carry flag.

Example



5.7.4.4 RCR

When the driving conditions are met, the RCR instruction shifts and rotates the data in D with the carry flag (M8022) rightwards by n bits. The low-order bits with the carry flag (M8022) that are rotated out of D fill the high-order bits of D.

RCR: Rotation right with carry

16-bit instruction	RCR: Continuous execution RCRP: Pulse execution			
32-bit instruction	DRCR: Continuous execution DRCRP: Pulse execution			
Operands	Name	Description	Value range	Data type
D	Source data/Target data	Address of the word soft element that stores the data	-	INT/ DINT
n				

	Number of bit places rotated on every execution	Value range: 1 ≤ n ≤ 16 (16-bit operation);1 ≤ n ≤ 32 (32-bit operation)	1 to 16/32	INT/ DINT

For soft elements, see the following table.

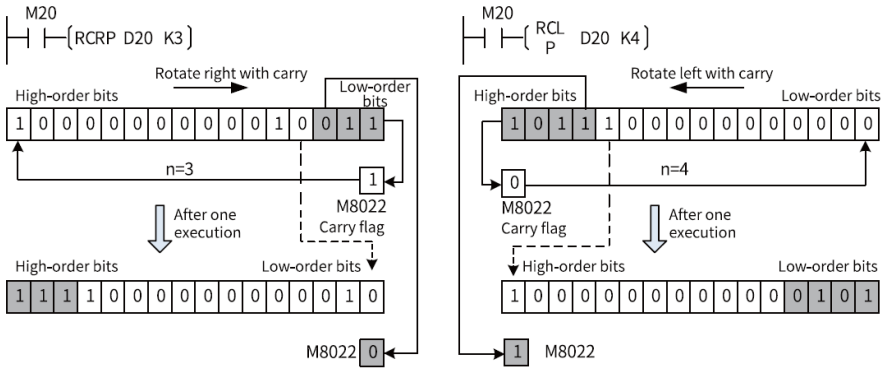
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The RCR instruction shifts and rotates the data in D with the carry flag M8022 rightwards by n bits.

The instruction of the pulse execution type is generally used. When the 32-bit instruction is executed, the register variable occupies two consecutive units.

Example



5.7.4.5 RCL

When the driving conditions are met, the RCL instruction shifts and rotates the data in D with the carry flag (M8022) leftwards by n bits. The high-order bits with the carry flag (M8022) that are rotated out of D fill the low-order bits of D.

RCL: Rotation left with carry

16-bit instruction	RCL: Continuous execution RCLP: Pulse execution			
32-bit instruction	DRCL: Continuous execution DRCLP: Pulse execution			
Operands	Name	Description	Value range	Data type
D	Source data/Target data	Address of the word soft element that stores the data	-	INT/ DINT

n	Number of bit places rotated on every execution	Value range: 1 ≤ n ≤ 16 (16-bit operation); 1 ≤ n ≤ 32 (32-bit operation)	1-16/32	INT/ DINT

For soft elements, see the following table.

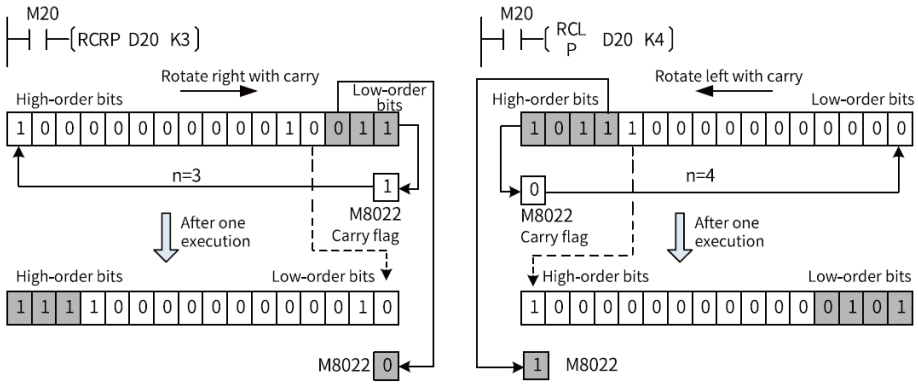
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The RCL instruction shifts and rotates the data in D with the carry flag (M8022) leftwards by n bits. The instruction of the pulse execution type is generally used.

When the 32-bit instruction is executed, the register variable occupies two consecutive units.

Example



5.7.4.6 SFTR

When the driving conditions are met, the SFTR instruction shifts n1 bit elements starting from D rightwards by n2 bits and transfers n2 bit elements starting from S to fill the high-order bits.

The n2 low-order bits that are shifted out are discarded. The values in the bit elements starting from S remain unchanged.

SFTR: Bit shift right

16-bit instruction	SFTR: Continuous execution SFTRP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Bit element start address	Start address of shifted bit elements	-	BOOL, array*n2

D	Incoming bit start address	Start address of incoming bit elements	-	BOOL, array*n1
n1	Incoming bit count	Number of incoming bit elements	1 to 256	INT
n2	Bit element count	Number of shifted bit elements	1 to n1	INT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
D	√[1]	√	√	-	-	√	-	-	-
n1	-	-	-	√	√	√	√	-	-
n2	-	-	-	√	√	√	√	-	-

NOTE

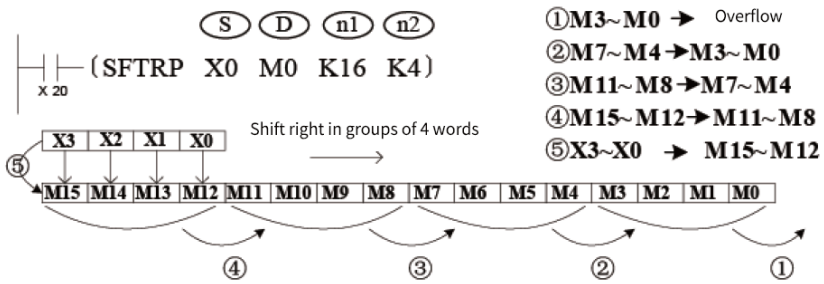
[1] The X element is not supported.

Function and Instruction Description

The SFTL instruction transfers n2 bit variables starting from S to D and shifts the bit variables rightwards by n2 bits.

The instruction of the pulse execution type is generally used.

Example



5.7.4.7 SFTL

When the driving conditions are met, the SFTL instruction shifts n1 bit elements starting from D leftwards by n2 bits and transfers n2 bit elements starting from S to fill the low-order bits.

The n2 high-order bits that are shifted out are discarded. The values in the bit elements starting from S remain unchanged.

SFTL: Bit shift left

16-bit instruction	SFTL: Continuous execution SFTLP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Bit element start address	Start address of shifted bit elements	-	BOOL, array*n2
D	Incoming bit start address	Start address of incoming bit elements	-	BOOL, array*n1
n1	Incoming bit count	Number of incoming bit elements	1 to 256	INT
n2	Bit element count	Number of shifted bit elements	1 to n1	INT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
D	√[1]	√	√	-	-	√	-	-	-
n1	-	-	-	√	√	√	√	-	-
n2	-	-	-	√	√	√	√	-	-

NOTE

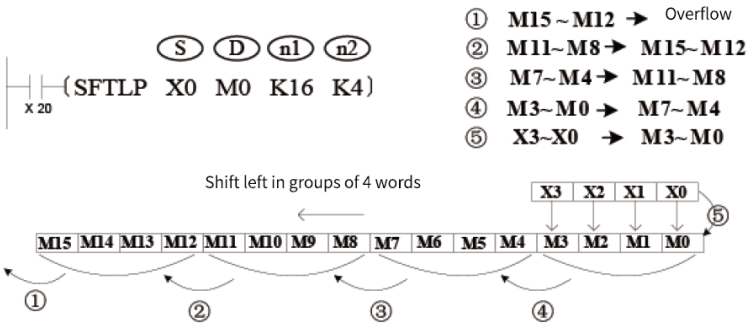
[1] The X element is not supported.

Function and Instruction Description

The instruction transfers n2 bit variables starting from S to D and shifts the bit variables rightwards by n2 bits.

The instruction of the pulse execution type is generally used.

Example



5.7.4.8 WSFR

When the driving conditions are met, the WSFR instruction shifts n1 word elements starting from D rightwards by n2 bits and transfers n2 word elements starting from S to fill the high-order bits.

The n2 low-order words that are shifted out are discarded. The values in the word elements starting from S remain unchanged.

WSFR: Word shift right

16-bit instruction	WSFR: Continuous execution WSFRP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Word element start address	Start address of shifted word elements	-	INT, array*n2
D	Incoming word start address	Start address of incoming word elements	-	INT, array*n1
n1	Incoming word count	Number of incoming word elements	1 to 256	INT
n2	Word element count	Number of shifted word elements	1 to n1	INT

For soft elements, see the following table.

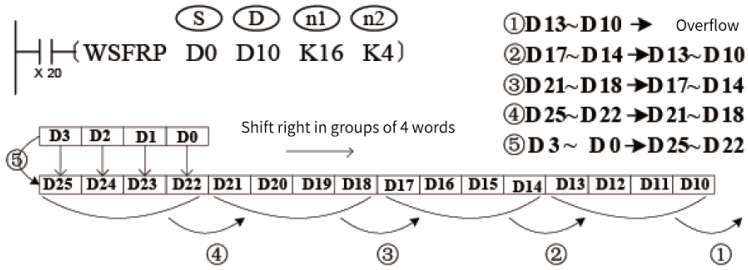
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n1	-	-	-	√	√	√	√	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
n2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The WSFR instruction shifts n1 word variables starting from D rightwards by n2 words and then transfers the n2 word variables starting from S to D to fill the high-order words. The instruction of the pulse execution type is generally used.

Example



5.7.4.9 WSFL

When the driving conditions are met, the WSFL instruction shifts n1 word elements starting from D leftwards by n2 words and transfers n2 word elements starting from S to fill the low-order words.

The n2 high-order words that are shifted out are discarded. The values in the word elements starting from S remain unchanged.

WSFL: Word shift left

16-bit instruction	WSFL: Continuous execution WSFLP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Word element start address	Start address of shifted word elements	-	INT, array*n2
D	Incoming word start address	Start address of incoming word elements	-	INT, array*n1
n1	Incoming word count	Number of incoming word elements	1 to 256	INT

n2	Word element count	Number of shifted word elements	1 to n1	INT

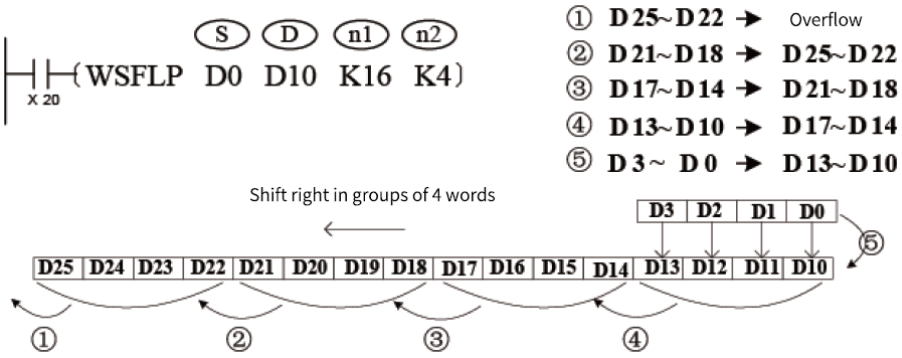
For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n1	-	-	-	√	√	√	√	-	-
n2	-	-	-	√	√	√	√	-	-

Function

The WSFL instruction shifts n1 word variables starting from D leftwards by n2 words and then transfers the n2 word variables starting from S to D to fill the low-order words. The instruction of the pulse execution type is generally used.

Example



5.7.4.10 SFWR

When driving conditions are met, the SFWR instruction writes the current content of S to a data register with a length of n from start address D+1. The value of the pointer D is incremented by 1 each time a data entry is written to the database.

SFWR: Shift write (FIFO)

16-bit instruction	SFWR: Continuous execution SFWRP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description		

			Value range	Data type
S	Data source	Data to be written, or address of the word soft element that stores the data	-	INT
D	Data area start address	Start address of word soft elements that store the data in a data area	-	INT, array*n
n	Data area length	Length of a data area, including pointer data	2 to 512	INT

For soft elements, see the following table.

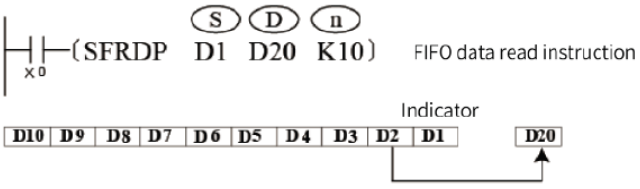
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	√	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The content of S is written to a first in first out (FIFO) queue with a length of n from start address D. The device with the first number stores a pointer. When the instruction is executed, the pointer is incremented by 1 and then the content of the source device (S) is written to the FIFO queue (D). The position of insertion into the queue is specified by the pointer.

The instruction of the pulse execution type is generally used.

Example



When X0 is 1, the value in D0 is written to D2 and the value in D1 changes to 1. When X0 switches from OFF to ON, the value in D0 is written to D3 and the value in D1 changes to 2, and so on. If the value in D1 exceeds the value of n minus 1, the instruction is not executed, and the carry flag M8022 is set to 1.

5.7.4.11 SFRD

When the driving conditions are met, the SFRD instruction reads the data from start address S+1 in a data register with a length of n. The read data is written to the destination register D.

SFRD: Shift read (FIFO)

	SFRD: Continuous execution SFRDP: Pulse execution

16-bit instruction				
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Data area start address	Start address of word soft elements that store the data in a data area	-	INT, array*n
D	Read data	Address for storing read data	-	INT
n	Data area length	Length of a data area	2 to 512	INT

See the following table for the soft elements.

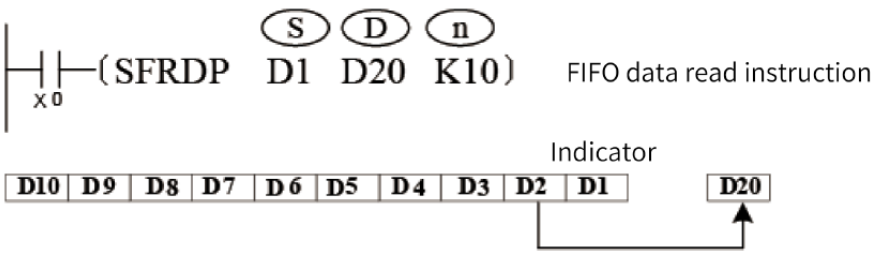
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The first piece of data in the FIFO queue (S) is read. The data within the queue is subsequently moved one word to the right to fill the read area, and the queue pointer is decremented by 1. The read data is written to D. The device with the first number stores a pointer. When the instruction is executed, the pointer is decremented by 1 and then the content of the source device specified by S is written to the FIFO queue specified by D. The position of insertion into the queue is specified by the pointer. If the pointer is 0, the instruction is not executed and the zero flag M8020 is set to 1.

The instruction of the pulse execution type is generally used.

Example



When X0 switches from OFF to ON, the following operations are performed (the content in D10 remains unchanged):

1. The content of D2 is read and transferred to D20.

2. D10 to D3 are shifted rightwards by one register.

3. The value of pointer D1 decreases by 1.

5.7.4.12 SFR

The SFR instruction shifts the data in a word element rightwards by n bits.

SFR: Bit shift right with carry

16-bit instruction	SFR: Continuous execution SFRP: Pulse execution			
32-bit instruction	DSFR: Continuous execution DSFRP: Pulse execution			
Operands	Name	Description	Value range	Data type
D	Word to be shifted	Number of the soft element that stores the data to be shifted	-	INT/DINT
n	Shift times	Number of shift times 0 ≤ n ≤ 15 (16-bit operation); 0 ≤ n ≤ 31 (32-bit operation)	0-15/31	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

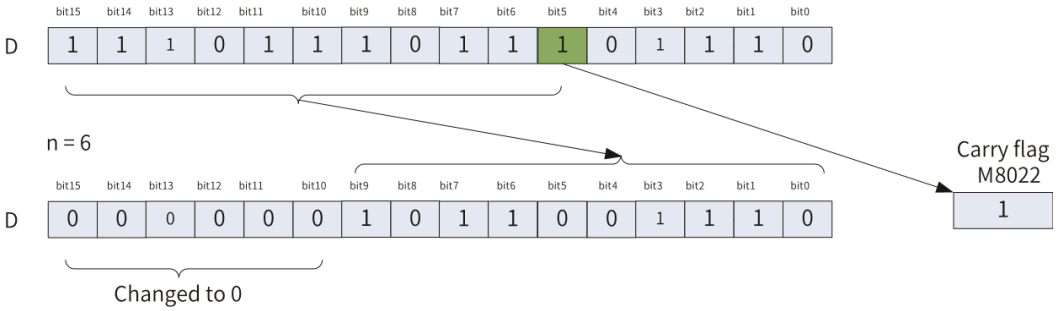
16-bit Instruction

n is a number in the range 0 to 15. When n is greater than or equal to 16, bits are shifted by the remainder of n%16. For example, when n is 20, bits are shifted four bit places (20%16 = 4) to the right.

32-bit instruction

n is a number in the range 0 to 31. When n is greater than or equal to 32, bits are shifted by the remainder of n%32. For example, when n is 40, bits are shifted rightwards by eight bits (40%32 = 8).

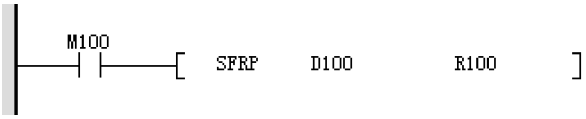
The 1/0 state of the (bn – 1)th bit in [D] is written to the carry flag M8022. The n bits starting from the most significant bit in [D] are filled with 0s.



Errors

An error is reported when n is less than 0.

Example



Before execution of the instruction

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0xAAAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x8
	16-bit INT	Dec	
M8022	BOOL	Bin	OFF
	16-bit INT	Dec	

After execution of the instruction

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x5500
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x8
	16-bit INT	Dec	
M8022	BOOL	Bin	ON

5.7.4.13 SFL

The SFL instruction shifts the data in a word element leftwards by n bits.

SFL: Bit shift left with carry

16-bit instruction	SFL: Continuous execution SFLP: Pulse execution			
32-bit instruction	DSFL: Continuous execution DSFLP: Pulse execution			
Operands	Name	Description	Value range	Data type
D	Word to be shifted	Number of the soft element that stores the data to be shifted	-	INT/DINT
n	Shift times	Number of shift times 0 ≤ n ≤ 15 (16-bit operation); 0 ≤ n ≤ 31 (32-bit operation)	0-15/31	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

16-bit instruction

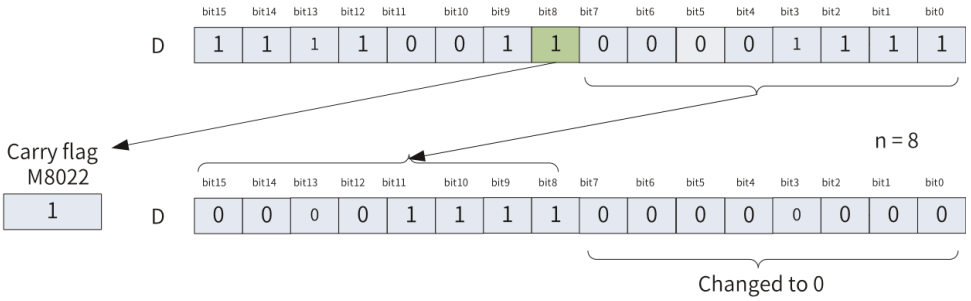
n is a number in the range 0 to 15. When n is greater than or equal to 16, bits are shifted by the remainder of n%16. For example, when n is 20, bits are shifted leftwards by four bits (20%16 = 4).

32-bit instruction

n is a number in the range 0 to 31. When n is greater than or equal to 32, bits are shifted by the remainder of n%32. For example, when n is 40, bits are shifted leftwards by eight bits (40%32 = 8).

The 1/0 state of the bn bit in [D] is written to the carry flag M8022.

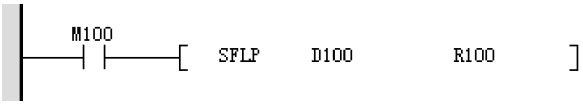
The n bits starting from the least significant bit in [D] are filled with 0s.



Errors

An error is reported when n is less than 0.

Example



Before execution of the instruction

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x5555
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x8
	16-bit INT	Dec	
M8022	BOOL	Bin	OFF
	16-bit INT	Dec	

After execution of the instruction

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0xAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x8
	16-bit INT	Dec	
M8022	BOOL	Bin	ON
	16-bit INT	Dec	

5.7.4.14 SHL

Return Value Type	Shift left operation. Shift an input value by one or more bits to the left, with zeros automatically inserted into the right side.		
BYTE, INT, and DINT			
Operand	Description	Range	Data Type
S1	Shift operation variable	-	BYTE, INT, and DINT
S2	Number of shifted bits	-	BYTE, INT, and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	-	-	-	√	√	-	√	-	-	-	-	-
S2	-	-	-	√	√	-	√	√	√	-	-	-

Function and Instruction Description

The SHL instruction shifts the variable S1 in binary form to the left by the number of bits in S2 and returns the shift result. In this instruction:

- S1 is the variable to be shifted.
- S2 is the number of shifted bits.
- The return value is the shift result.
- The type and length of the return value are consistent with those of S1.

NOTE

1. SHL is a bit operation instruction, and the result it returns can only be used with bit operation results, binary constants, octal constants, or hexadecimal constants in an operation. If the result is used with other types of data in an operation, a compilation error is reported.
2. The result returned by SHL can be used in an explicit conversion operation TO_XXX, such as TO_REAL.
3. When the value of S2 is greater than the number of bits in the data type of S1, no exception occurs, and the return value is 16#0.
4. When S1 is an expression, it is implicitly converted into the type with the maximum length in the expression before the shift operation is performed.
5. After the returned result is assigned to a variable, the variable can be used for the operations allowed by the variable.

Instruction Example

```
dint0 := SHL(dint1, 2);
```

Shift dint1 to the left by 2 bits and store the result in dint0.

```
dint0 := 16#0F AND SHL(dint1, 2);
```

Perform bit AND operation on the left shift result and hexadecimal constant and store the result in dint0.

NOTE

Operation with binary constants and octal constants is similar to that of hexadecimal constants. You only need to change 16 to 2 or 8 as required.

5.7.4.15 SHR

Return value type	Shift right operation, with zeros and ones automatically inserted into the positive number left side and negative number left side respectively.		
BYTE/INT/DINT			
Operands	**Description**	**Value range**	**Data type**
S1	Shift operation variable	-	BYTE/INT/DINT
S2	Number of shifted bits	-	BYTE/INT/DINT

See the following table for soft elements.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S1	-	-	-	√	√	-	√	-	-	-	-	-
S2	-	-	-	√	√	-	√	√	√	-	-	-

Function

This instruction is used to shift the variable S1 in binary form to the right by S2 bits and return the shift result, Specifically:

- S1 is the variable to be shifted.
- S2 is the number of shifted bits.
- The return value is the shift result.
- The type and length of the return value are consistent with those of S1.

NOTE

1. SHR is a bit operation instruction, and the result it returns can only be used with bit operation results, binary constants, octal constants, or hexadecimal constants in an operation.
2. The result returned by SHR can be used in an explicit conversion operation TO_XXX, such as TO_REAL.
3. When the value of S2 is greater than the number of bits in the data type of S1, no exception will occur, and the return value is 16#0.
4. When S1 is an expression, it will be implicitly converted into the type with the maximum length in the expression before the shift operation is performed.
5. After the returned result is assigned to a variable, the variable can be used for the operations allowed by the variable.

Example

```
dint0 := SHR(dint1, 2);
```

Shift dint1 to the right by 2 bits and store the result in dint0.

5.7.5 Other Data Processing

5.7.5.1 List of Instructions

The following table lists other data processing instructions.

Category	Instruction	Function
Other data processing instructions	SWAP	Byte swap
	BON	Bit state check
	SUM	Sum of ON bits
	RAND	Random number generation within limits
	XCH	Data exchange
	ABS	Absolute value
	EABS	Absolute value of floating-point number
	EFMOV	Multi-point floating-point move
	CCD	Check code
	CRC	CRC code calculation
	LRC	LRC code calculation

5.7.5.2 SWAP

The SWAP instruction exchanges the upper and lower bytes of the variable in S.

SWAP: Byte swap

16-bit instruction	SWAP: Continuous execution SWAPP: Pulse execution			
32-bit instruction	DSWAP: Continuous execution DSWAPP: Pulse execution			
Operands	Name	Description	Value range	Data type
D	Operands	Unit that stores the data whose upper and lower bytes will be exchanged	-	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-

Function

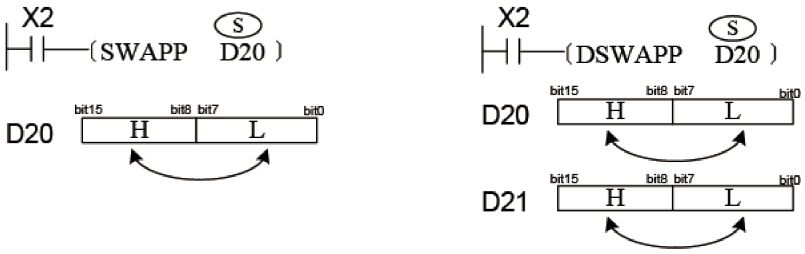
The upper and lower bytes of the variable in S are exchanged.

In 16-bit operation, the upper eight bits and the lower eight bits are exchanged.

In 32-bit operation, the upper eight bits and the lower eight bits of two registers are exchanged.

> **NOTE**
>
> This instruction is generally programmed as the pulse execution type. If it is programmed as the continuous execution type, swap is performed on every program scan.

Example



In the figure on the left, the values of the high-order 8 bits and low-order 8 bits in D20 are swapped.

In the figure on the right, the values of the high-order 8 bits and low-order 8 bits in D20 are swapped.

The values of the high-order 8 bits and low-order 8 bits in D21 are swapped.

5.7.5.3 BON

When the driving conditions are met, the BON instruction checks the state of the nth bit of the binary data in S and outputs the result to D.

BON: Bit state check

16-bit instruction	BON: Continuous execution BONP: Pulse execution			
32-bit instruction	DBON: Continuous execution DBONP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Source data	Data, or address of the word soft element that stores the data	-	INT/ DINT
D	Controlled bit	Controlled bit element	-	BOOL
n	Designated bit	Designated bit in S; value range: 0 ≤ n ≤ 15 (16-bit operation); 0 ≤ n ≤ 31 (32-bit operation)	0-15/31	INT/ DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	√[1]	√	√	-	-	√	-	-	-
n	-	-	-	√	√	√	√	-	-

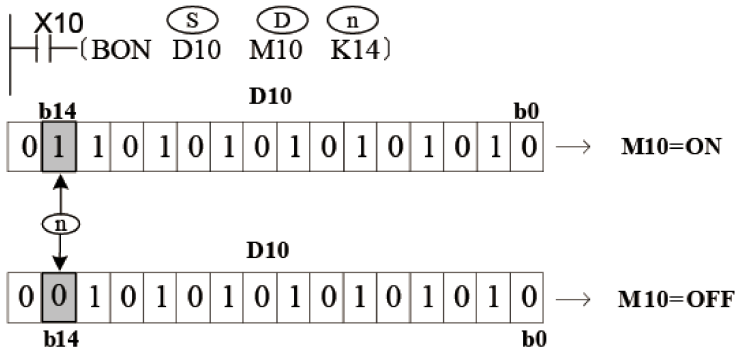
NOTE

[1] The X element is not supported.

Function

The BON instruction checks the state of the nth bit in S and stores the result in D.

Example



When the 14th bit in D10 is 1, M10 is set to ON.

When the 14th bit in D10 is 0, M10 is reset.

When X10 switches from ON to OFF, the state of M10 remains unchanged.

5.7.5.4 SUM

When driving conditions are met, the SUM instruction counts the ON bits (with a value of 1) of the binary data in S. The result is stored in D.

SUM: Sum of ON bits

16-bit instruction	SUM: Continuous execution SUMP: Pulse execution			
32-bit instruction	DSUM: Continuous execution DSUMP: Pulse execution			
Operands	Name	Description	Value range	Data type
S			-	INT/DINT

	Data to be counted	Data to be counted, or address of the soft element that stores the data		
D	Counting result	Address of the soft element that stores the data	-	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

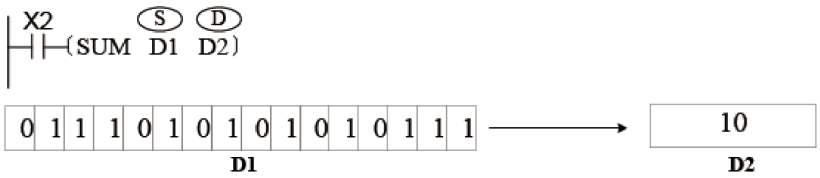
Function

ON bits (with a value of 1) of the BIN data in S are counted. The result is stored in D.

When DSUM or DSUMP is executed, the number of ON bits (the value is 1) among the 32 bits in (S+1, S) is written to D, and all bits in D+1 are set to 0.

If all bits in S are 0, the zero flag M8020 is set to ON.

Example



The number of ON bits (the value is 1) in D1 is counted and the result is stored in D2.

5.7.5.5 RAND

The RAND instruction generates a random number within a specified range.

RAND: Random number generation within limits

16-bit instruction	-			
32-bit instruction	RAND: Continuous execution RANDP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	RAND lower limit	The lower limit of random number	-	DINT

S2	RAND upper limit	The upper limit of random number	-	DINT
S3	Random seed	The seed for random number calculation. The generated random number changes with the seed	-	DINT
D	Generated RAND	The generated random number	-	DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The RAND instruction generates a random number within a specified range. The random number is generated by the random seed. When the range is determined, there is a one-to-one correspondence between the random seed and the generated random number. That is, when the random seed changes, the generated random number changes accordingly.

Command parameter description

- Lower limit: Lower limit of the random number
- Upper limit: Upper limit of the random number
- Random seed: Input for generating the random number, which is not restricted by the upper and lower limits
- Random number: Generated random number, which is between the upper and lower limits An error is returned when S1 is greater than S2.

Example



Program running flag

- Running: ON
- Stopped: OFF

1	...	D100	DINT	DEC	0
2	...	D200	DINT	DEC	10000
3	...	D300	DINT	DEC	1000
4	...	D400	DINT	DEC	9247
5	...				

5.7.5.6 XCH

When the driving conditions are met, the XCH instruction exchanges the data in S and D.

XCH -- Data exchange

16-bit Instruction	XCH: Continuous execution/XCHP: Pulse execution			
32-bit Instruction	DXCH: Continuous execution/DXCHP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data 1	Word element 1 that stores the data to be exchanged	-	INT and DINT
D	Data 2	Word element 2 that stores the data to be exchanged	-	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The XCH instruction requires contact driving and has two operands. It exchanges the values in S and D.

Instruction Example

Example 1

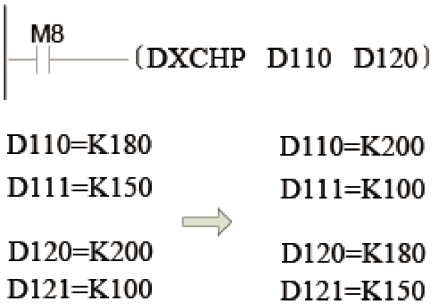
Before execution and after execution





Example 2

Before execution and after execution



5.7.5.7 EFMOV

When the driving conditions are met, the EFMOV instruction transfers the floating-point number data in S to the n registers starting from the address specified in D.

EFMOV: Multi-point floating-point move

16-bit instruction	-			
32-bit instruction	DEFMOV: Continuous execution/DEFMOVP: Pulse execution, 13 steps			
Operands	Name	Description	Value range	Data type
S	Source data	Data to be transferred, or address of the floating-point element that stores the data	-	-
D	Transfer destination start address	Start address of word soft elements that store the data arriving at a destination	-	Array *n
n	Target number	Number of word soft elements that store the data arriving at a destination	1-512	-

For soft elements, see the following table.

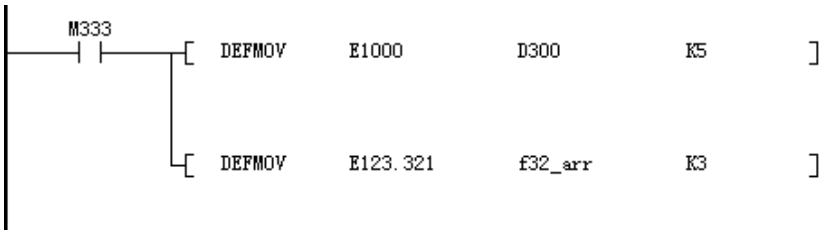
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The EFMOV instruction requires contact driving and has three operands. It copies the floating-point number data in S to the n units starting from the address specified in D.

The value of n ranges from 1 to 512.

Example



When M333 is ON, the calculation result is as follows:

1000 Floating point — > D300 ,D301

1000 Floating point — > D3 02,D303

1000 Floating point — > D304 ,D305

1000 Floating point — > D306 ,D307

1000 Floating point — > D308 ,D309

123.321 Floating point — > f32_arr[0]

123.321 Floating point — > f32_arr[1]

123.321 Floating point — > f32_arr[2]

D300	REAL	Dec	1000.000
D302	REAL	Dec	1000.000
D304	REAL	Dec	1000.000
D306	REAL	Dec	1000.000
D308	REAL	Dec	1000.000
D310	REAL	Dec	0.000000
f32_arr	REAL[5]	Dec	
f32_arr[0]	REAL	Dec	123.3210
f32_arr[1]	REAL	Dec	123.3210
f32_arr[2]	REAL	Dec	123.3210
f32_arr[3]	REAL	Dec	0.000000
f32_arr[4]	REAL	Dec	0.000000

5.7.5.8 CCD

When the driving conditions are met, the CCD instruction calculates the checksum of the n data entries starting from S and stores the result in D. The XOR operation result is stored in D+1.

CCD: Check code

16-bit instruction	CCD: Continuous execution CCDP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Data source	Start address of consecutive units that store the variables whose checksum will be calculated	-	INT, array*n
D	Operation result	Summation result stored in D; XOR logical operation result stored in D+1	-	INT, array*2

n	Checked byte count	Number of bytes contained in checked variables	1 to 256	INT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function

The CCD instruction performs two types of checksum operations on n variables starting from S and stores the summation result in D and the XOR logical operation result in D+1. The string checksum operation ensures correct data transfer during communication. Specifically:

Summation is the process where the values of n variables are directly added together.

The XOR logical operation is described as follows:

- The variables are converted to binary numbers.
- Bit 0 = 1 occurrences in all variables are counted. If the counting result is an even number, the XOR operation result for bit 0 is 0; if the counting result is an odd number, the XOR operation result for bit 0 is 1.
- Count the number of variables of which bit 1 is 1. If it is an even number, bit 1 of the XOR result is 0; if it is an odd number, bit 1 of the XOR result is 1.

Calculate bit 2 to bit 7 in a similar way. Convert the resulting binary number into a hexadecimal equivalent, which is the XOR operation result (or called a polarity value).

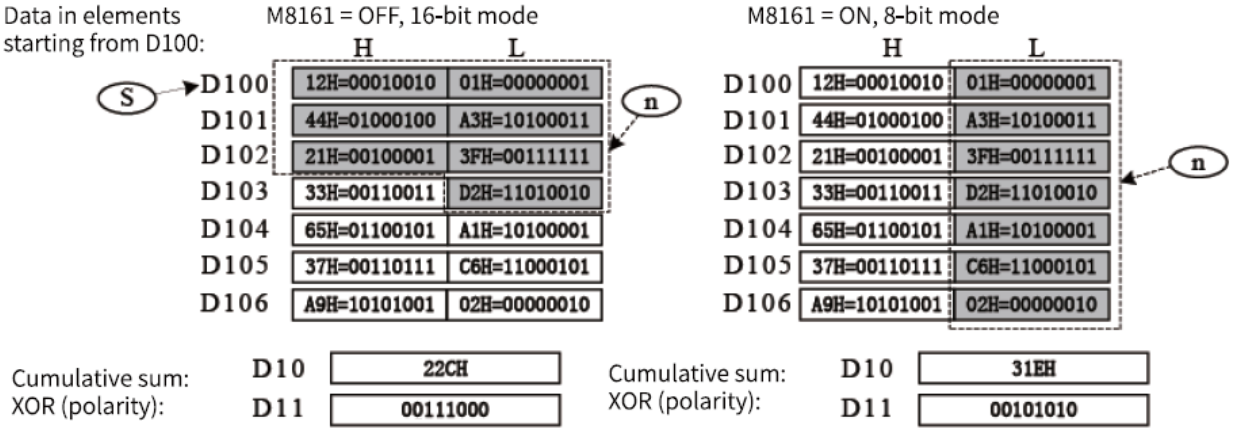
NOTE

Note during programming that instructions including HEX, ASCI, and CCD share M8161 mode flag.

Example



The M8161 flag determines which variable width mode to use. When M8161 = OFF, the 16-bit mode is enabled, whereby the high- and low-order bytes of variables are taken for the operation. When M8161 = ON, the 8-bit mode is enabled, whereby only the low-order bytes of variables are taken for the operation and the high-order bytes are discarded. Therefore, the length of the actually used variable area is increased. See the following figure.



5.7.5.9 CRC

Cyclic redundancy check (CRC) is commonly used during communication. The CRC instruction calculates the CRC code.

CRC -- Cumulative sum check code

16-bit Instruction	CRC: Continuous execution/CRCP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Source data	Start address of elements that store the data for CRC code calculation (RTU mode)	-	INT, array*n
n	Data count	Number of operated data entries (K1 to K256)	1 to 256	INT
D	Result	Start address of elements that store the operation result	-	INT, array*2

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	√	√	-	-
n	-	-	-	√	√	-	-	-	-

Function and Instruction Description

16-bit conversion mode: When M8161 is OFF, the CRC instruction takes the high-order 8 bits and low-order 8 bits (n data points in total) starting from [S] in the unit of 16 bits for CRC code calculation and stores the result in the high-order 8 bits and low-order 8 bits in [D].

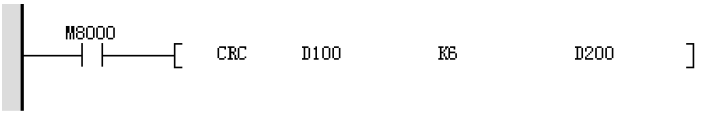
8-bit conversion mode: When M8161 is ON, the CRC instruction takes the low-order 8 bits (n data points in total) starting from [S] in the unit of 8 bits for CRC code calculation and stores the low-order 8 bits of the result in [D] and the high-order 8 bits of the result in [D+1].

An error is reported in the following cases:

Error 6706 is reported when n is out of range.

Instruction Example

When M8161 is ON, the 8-bit conversion mode is used. The low-order 8 bits of elements D100 to D105 are taken for CRC code calculation. The result is stored in the low-order 8 bits of D200 and D201.



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1
D101	16-bit INT	Hex	0x2
D102	16-bit INT	Hex	0x4
D103	16-bit INT	Hex	0x20
D104	16-bit INT	Hex	0x0
D105	16-bit INT	Hex	0x12
	16-bit INT	Dec	
M8161	BOOL	Bin	ON
	16-bit INT	Dec	
D200	16-bit INT	Hex	0xF8
D201	16-bit INT	Hex	0xFD

When M8161 is OFF, the 16-bit conversion mode is used. The low-order 8 bits of elements D100 to D105 are taken for CRC code calculation. The result is stored in the high-order 8 bits and low-order 8 bits of D200.

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1
D101	16-bit INT	Hex	0x2
D102	16-bit INT	Hex	0x4
D103	16-bit INT	Hex	0x20
D104	16-bit INT	Hex	0x0
D105	16-bit INT	Hex	0x12
	16-bit INT	Dec	
M8161	BOOL	Bin	OFF
	16-bit INT	Dec	
D200	16-bit INT	Hex	0xB202
D201	16-bit INT	Hex	0x0

5.7.5.10 LRC

The LRC instruction calculates the longitudinal redundancy check (LRC) code in ASCII mode.

LRC: LRC code calculation

16-bit instruction	LRC: Continuous execution LRCP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Source data	Head address of soft elements that store the data for LRC code calculation		INT, array*n
n	Data count	Number of operated data entries, which must be an even number; value range: K1 to K256	1 to 256	INT
D	Result	Register that stores the operation result		INT, array*2

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	-	-	-	-

Function

The LRC code is acquired by calculating the two's complement of the sum of values within the range from the communication address to the end of the data content.

Example: 01 H + 03 H + 21 H + 02 H + 00 H + 02 H = 29 H, and the two's complement of the sum is D7H (which corresponds to the ASCII codes 44H and 37H).

16-bit conversion mode: When M8161 is OFF, the LRC instruction takes the high-order 8 bits and low-order 8 bits (n data points in total) starting from [S] in the unit of 16bit for LRC code calculation and stores the result in the high-order 8 bits and low-order 8 bits in [D].

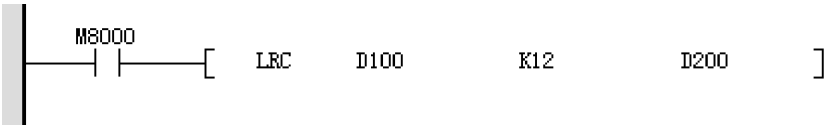
8-bit conversion mode: When M8161 = ON, the lower eight bits (n data points in total) from start address [S] are taken byte by byte for LRC code calculation. The lower eight bits of the result are stored in [D], and the upper eight bits are stored in [D+1].

Errors

An error is returned in the following conditions:

- n is out of the specified range.
- n is an odd number.

Example



- 16-bit mode (M8161 = OFF)

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x3130
D101	16-bit INT	Hex	0x3330
D102	16-bit INT	Hex	0x3132
D103	16-bit INT	Hex	0x3230
D104	16-bit INT	Hex	0x3030
D105	16-bit INT	Hex	0x3230
D200	16-bit INT	Hex	0x3744
M8161	BOOL	Bin	OFF
	16-bit INT	Dec	

- 8-bit mode (M8161 = ON)

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x30
D101	16-bit INT	Hex	0x31
D102	16-bit INT	Hex	0x30
D103	16-bit INT	Hex	0x33
D104	16-bit INT	Hex	0x32
D105	16-bit INT	Hex	0x31
D106	16-bit INT	Hex	0x30
D107	16-bit INT	Hex	0x32
D108	16-bit INT	Hex	0x30
D109	16-bit INT	Hex	0x30
D110	16-bit INT	Hex	0x30
D111	16-bit INT	Hex	0x32
D112	16-bit INT	Hex	0x0
D200	16-bit INT	Hex	0x44
D201	16-bit INT	Hex	0x37
M8161	BOOL	Bin	ON
	16-bit INT	Hex	

5.8 Matrix Instructions

5.8.1 Matrix Operation

5.8.1.1 List of Instructions

The following table lists the matrix operation instructions.

Instruction category	Name	Function
Matrix operations	BK+	Block data addition
	BK-	Block data subtraction
	MAND	Matrix AND operation
	MOR	Matrix OR operation
	MXOR	Matrix XOR operation
	MXNR	Matrix XNR operation
	MINV	Matrix inverse operation

5.8.1.2 BK+

The BK+ instruction adds binary block data.

BK+: Block data addition

16-bit instruction	BK+: Continuous execution BK+P: Pulse execution			
32-bit instruction	DBK+: Continuous execution DBK+P: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Source address 1	Start number of soft elements that store the data for which the addition operation is performed	-	INT/DINT, array*n
S2	Source address 2	Constant for which the addition operation is performed, or start number of soft elements that store the data for which the addition operation is performed	-	INT/DINT, array*n
D	Destination address	Start number of soft elements that store the operation result	-	INT/DINT, array*n
n	Data count	Number of data entries in an operation	1 to 256	INT/DINT

NOTE

n indicates the number of data entries to be operated. If it is a constant, only a fixed number of data entries can be operated; if it is a variable, the number of data entries to be operated can be changed by adjusting the value of n.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The BK+ instruction adds the n data entries (16- or 32-bit) starting from [S1] and the n data entries (16- or 32-bit) starting from [S2] together and stores the result in n units (16- or 32-bit) starting from [D].

[S1+0]	K1111		[S2+0]	K1111		[D+0]	K2222
[S1+1]	K1111		[S2+1]	K-2222		[D+1]	K-1111
...	...	+	...	...	=	...	...
[S1+n-2]	K1111		[S2+n-2]	K3333		[D+n-2]	K4444
[S1+n-1]	K1111		[S2+n-1]	K4444		[D+n-1]	K5555

A signed constant (16- or 32-bit) can be directly specified in [S2].

[S1+0]	K1111					[D+0]	K3333
[S1+1]	K1111					[D+1]	K3333
...	...	+	K2222		=	...	...
[S1+n-2]	K1111					[D+n-2]	K3333
[S1+n-1]	K1111					[D+n-1]	K3333

If the soft elements starting from [S1], [S2], or [D] are beyond the corresponding element range, an error is returned and the instruction is not executed.

If the set length of BK+ instruction is greater than that of the array variable, the operation runs cyclically with the consecutive address of array variable till the set length is reached.

Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	1111
D101	16-bit INT	Decimal	1111
D102	16-bit INT	Decimal	1111
D103	16-bit INT	Decimal	1111
D104	16-bit INT	Decimal	1111
D105	16-bit INT	Decimal	0
D200	16-bit INT	Decimal	2222
D201	16-bit INT	Decimal	2222
D202	16-bit INT	Decimal	2222
D203	16-bit INT	Decimal	2222
D204	16-bit INT	Decimal	2222
D205	16-bit INT	Decimal	0
R100	16-bit INT	Decimal	3333
R101	16-bit INT	Decimal	3333
R102	16-bit INT	Decimal	3333
R103	16-bit INT	Decimal	3333
R104	16-bit INT	Decimal	3333
R105	16-bit INT	Decimal	0

5.8.1.3 BK-

The BK- instruction subtracts binary block data.

BK-: Block data subtraction

16-bit instruction	BK-: Continuous execution/BK-P: Pulse execution

32-bit instruction	DBK: Continuous execution DBK: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Source address 1	Start number of soft elements that store the data subjected to a subtraction operation	-	INT/DINT, array*n
S2	Source address 2	Constant for which the subtraction operation is performed, or start number of elements that store the data for which the subtraction operation is performed	-	INT/DINT, array*n
D	Destination address	Start number of soft elements that store the operation result	-	INT/DINT, array*n
n	Data count	Number of data entries in an operation	1 to 256	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The BK- instruction subtracts the n data entries (16- or 32-bit) starting from [S2] from the n data entries (16- or 32-bit) starting from [S1] and stores the result in n units (16- or 32-bit) starting from [D].

[S1+0]	K1111		[S2+0]	K1111	=	[D+0]	K0
[S1+1]	K1111		[S2+1]	K-2222		[D+1]	K3333
...	...	—	...	...		...	...
[S1+n-2]	K1111		[S2+n-2]	K3333		[D+n-2]	K-2222
[S1+n-1]	K1111		[S2+n-1]	K4444		[D+n-1]	K-3333

A signed constant (16- or 32-bit) can be directly specified in [S2].

[S1+0]	K1111
[S1+1]	K1111
...	...
[S1+n-2]	K1111
[S1+n-1]	K1111

— K-2222 =

[D+0]	K3333
[D+1]	K3333
...	...
[D+n-2]	K3333
[D+n-1]	K3333

If the elements starting from [S1], [S2], or [D] are beyond the corresponding element range, an error is returned and the instruction is not executed.

Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Dec	3333
D101	16-bit INT	Dec	3333
D102	16-bit INT	Dec	3333
D103	16-bit INT	Dec	3333
D104	16-bit INT	Dec	3333
D105	16-bit INT	Dec	0
D200	16-bit INT	Dec	2222
D201	16-bit INT	Dec	2222
D202	16-bit INT	Dec	2222
D203	16-bit INT	Dec	2222
D204	16-bit INT	Dec	2222
D205	16-bit INT	Dec	0
R100	16-bit INT	Dec	1111
R101	16-bit INT	Dec	1111
R102	16-bit INT	Dec	1111
R103	16-bit INT	Dec	1111
R104	16-bit INT	Dec	1111
R105	16-bit INT	Dec	0

5.8.1.4 MAND

The MAND instruction performs an AND operation on the matrices and stores the result in D.

MAND: Matrix AND operation

16-bit instruction	MAND: Continuous execution MANDP: Pulse execution			
32-bit instruction	DMAND: Continuous execution DMANDP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Matrix 1	Operand element 1 in an operation	-	INT/DINT, array*n
S2	Matrix 2	Operand element 2 in an operation	-	INT/DINT, array*n
D			-	

	Operation result	Start number of elements that store the operation result		INT/DINT, array*n
n	Data count	Number of data entries in an operation; value range: 1 to 256	1 to 256	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

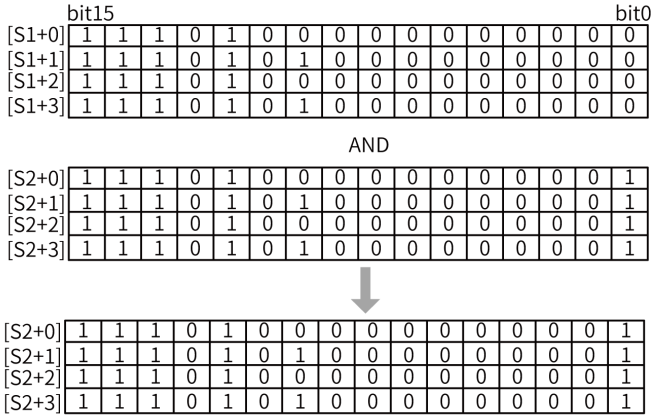
Function

The MAND instruction performs an AND operation by bit on the n groups of data starting from [S1] and the n groups of data starting from [S2] and stores the result in elements starting from [D].

The result is 1 when the values of two bits are both 1; otherwise, the result is 0.

Assume that n = 4. The result of a matrix XNR operation is as follows:

> **NOTE**
>
> For a 16-bit instruction, n indicates the number of words; for a 32-bit instruction, n indicates the number of dwords.



Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0xFFFF
D101	16-bit INT	Hex	0xFFFF
D102	16-bit INT	Hex	0xFFFF
	16-bit INT	Dec	
D200	16-bit INT	Hex	0xFF
D201	16-bit INT	Hex	0xFF00
D202	16-bit INT	Hex	0xAAAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0xFF
R101	16-bit INT	Hex	0xFF00
R102	16-bit INT	Hex	0xAAAA

5.8.1.5 MOR

The MOR instruction performs an OR operation on the matrix and stores the result in D.

MOR: Matrix OR operation

16-bit instruction	MOR: Continuous execution MORP: Pulse execution			
32-bit instruction	DMOR: Continuous execution DMORP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Matrix 1	Operand element 1 in an operation	-	INT/DINT, array*n
S2	Matrix 2	Operand element 2 in an operation	-	INT/DINT, array*n
D	Operation result	Start number of elements that store the operation result	-	INT/DINT, array*n
n	Data count	Number of data entries in an operation; value range: 1 to 256	1 to 256	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

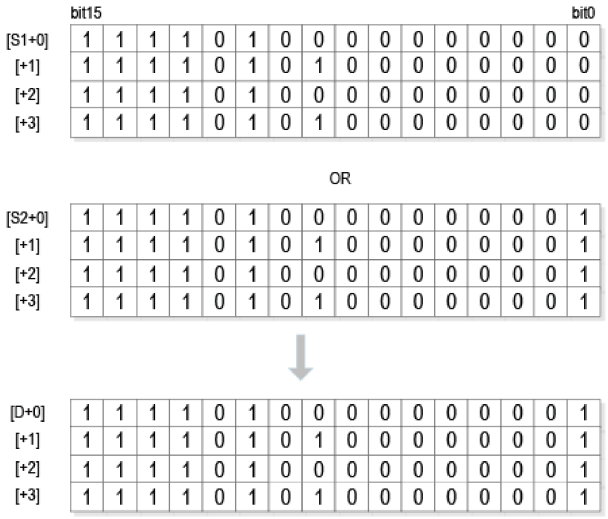
Function

The MOR instruction performs an OR operation by bit on the n groups of data starting from [S1] and the n groups of data starting from [S2] and stores the result in elements starting from [D].

The result is 1 when the value of any bit is 1; otherwise, the result is 0.

Assume that n = 4. The result of a matrix XNR operation is as follows:

> **NOTE**
>
> For the 16-bit instruction, n indicates the number of words; for the 32-bit instruction, n indicates the number of dwords.



Example



5.8.1.6 MXNR

The MXNR instruction performs an XNOR operation on the matrix and stores the result in D.

MXNR: Matrix XNOR operation

16-bit instruction	MXNR: Continuous execution MXNRP: Pulse execution			
32-bit instruction	DMXNR: Continuous execution DMXNRP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Matrix 1	Operand element 1 in an operation	-	INT/DINT, array*n

S2	Matrix 2	Operand element 2 in an operation	-	INT/DINT, array*n
D	Operation result	Start number of elements that store the operation result	-	INT/DINT, array*n
n	Data count	Number of data entries in an operation; value range: 1 to 256	1 to 256	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

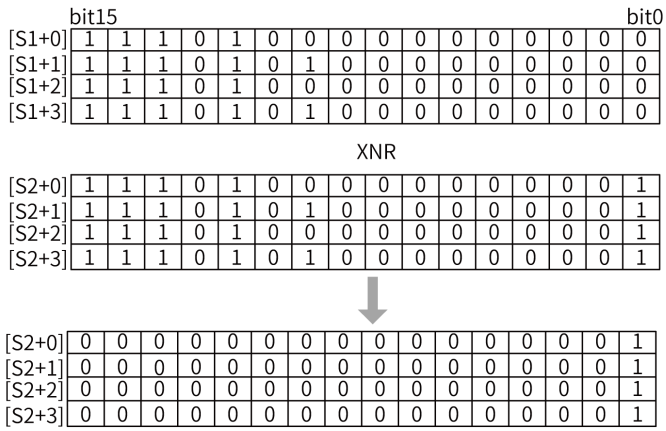
Function and Instruction Description

The MXNR instruction performs an XNOR operation by bit on the n groups of data starting from [S1] and the n groups of data starting from [S2] and stores the result in elements starting from [D].

The result of the XNOR operation is 1 when the values of the two bits are different; otherwise, the result is 0.

Assume that n = 4. The result of a matrix XNR operation is as follows:

> **NOTE**
>
> For a 16-bit instruction, n indicates the number of words; for a 32-bit instruction, n indicates the number of dwords.



Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x0
D101	16-bit INT	Hex	0x0
D102	16-bit INT	Hex	0x0
	16-bit INT	Dec	
D200	16-bit INT	Hex	0x0
D201	16-bit INT	Hex	0x0
D202	16-bit INT	Hex	0xAAAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0xFFFF
R101	16-bit INT	Hex	0xFFFF
R102	16-bit INT	Hex	0x5555

5.8.1.7 MXOR

The MXOR instruction performs an XOR operation on the matrix and stores the result in D.

MXOR: Matrix XOR

16-bit instruction	MXOR: Continuous execution MXORP: Pulse execution			
32-bit instruction	DMXOR: Continuous execution DMXORP: Pulse execution			
Operands	Name	Description	Value range	Data type
S1	Matrix 1	Operand element 1 in an operation	-	INT/DINT, array*n
S2	Matrix 2	Operand element 2 in an operation	-	INT/DINT, array*n
D	Operation result	Start number of elements that store the operation result	-	INT/DINT, array*n
n	Data count	Number of data entries in an operation	1 to 256	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

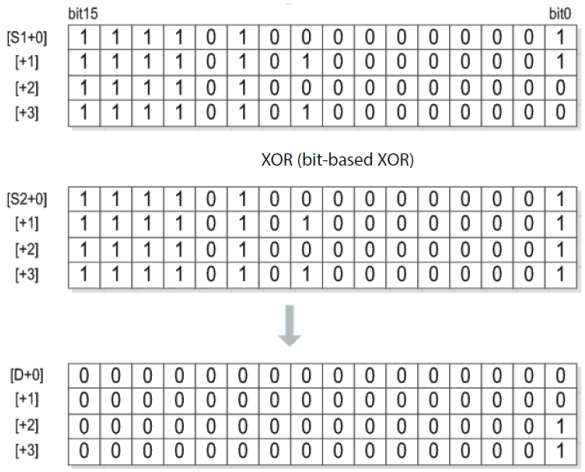
Function

The MXOR instruction performs an XOR operation by bit on the n groups of data starting from [S1] and the n groups of data starting from [S2] and stores the result in elements starting from [D].

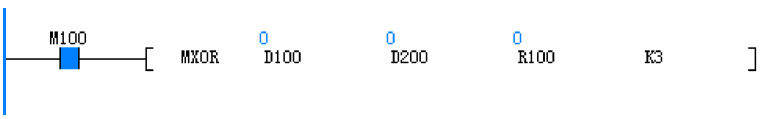
The result of the XOR operation is 1 when the values of the two bits are different; otherwise, the result is 0.

Assume that n = 4. The result of a matrix XNR operation is as follows:

> **NOTE**
>
> For a 16-bit instruction, n indicates the number of words; for a 32-bit instruction, n indicates the number of dwords.



Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x0
D101	16-bit INT	Hex	0x0
D102	16-bit INT	Hex	0x5555
	16-bit INT	Dec	
D200	16-bit INT	Hex	0x0
D201	16-bit INT	Hex	0x0
D202	16-bit INT	Hex	0xAAAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x0
R101	16-bit INT	Hex	0x0
R102	16-bit INT	Hex	0xFFFF

5.8.1.8 MINV

The MINV instruction inverts all bits of the specified matrix.

MINV: Matrix inverse

16-bit instruction	MINV: Continuous execution MINVP: Pulse execution			
32-bit instruction	DMINV: Continuous execution DMINVP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Matrix	Operand element in an operation	-	INT/DINT, array*n
D	Operation result	Start number of elements that store the operation result	-	INT/DINT, array*n
n	Data count	Number of data entries in an operation; value range: 1 to 256	1 to 256	INT/DINT

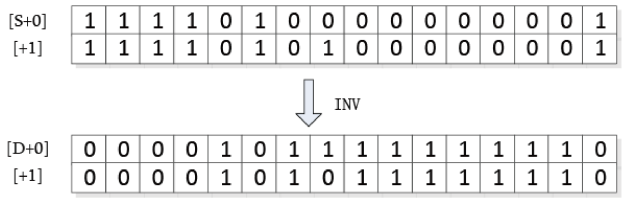
For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

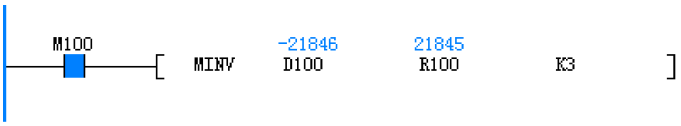
Function

The MINV instruction inverts the n groups of data starting from [S] by bit and stores the result in elements starting from [D].

> **NOTE**
>
> For a 16-bit instruction, n indicates the number of words; for a 32-bit instruction, n indicates the number of dwords.



Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0xAAAA
D101	16-bit INT	Hex	0xAAAA
D102	16-bit INT	Hex	0xAAAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x5555
R101	16-bit INT	Hex	0x5555
R102	16-bit INT	Hex	0x5555

5.8.2 Matrix Comparison

5.8.2.1 List of Instructions

The following table lists the matrix comparison instructions.

Category	Instruction	Function
Matrix comparison instruction	BKCMP=	Matrix comparison equal to (S1 = S2)
	BKCMP>	Matrix comparison greater than (S1 > S2)
	BKCMP<	Matrix comparison less than (S1 < S2)
	BKCMP<>	Matrix comparison not equal to (S1 ≠ S2)
	BKCMP<=	Matrix comparison less than or equal to (S1 ≤ S2)
	BKCMP>=	Matrix comparison greater than or equal to (S1 ≥ S2)

5.8.2.2 BKCMP#

The following instructions compare block data according to the comparison condition set in each instruction.

BKCMP= -- Matrix comparison equal to (S1 = S2)

BKCMP> -- Matrix comparison greater than (S1 > S2)

BKCMP< -- Matrix comparison less than (S1 < S2)

BKCMP<> -- Matrix comparison not equal to (S1 ≠ S2)

BKCMP<= -- Matrix comparison less than or equal to (S1 ≤ S2)

BKCMP>= -- Matrix comparison greater than or equal to (S1 ≥ S2)

16-bit Instruction	BKCMP=: Continuous execution/BKCMP=P: Pulse execution
32-bit Instruction	DBKCMP=: Continuous execution/DBKCMP=P: Pulse execution
16-bit Instruction	BKCMP>: Continuous execution/BKCMP>P: Pulse execution
32-bit Instruction	DBKCMP>: Continuous execution/DBKCMP>P: Pulse execution
16-bit Instruction	BKCMP<: Continuous execution/BKCMP<P: Pulse execution
32-bit Instruction	DBKCMP<: Continuous execution/DBKCMP<P: Pulse execution
16-bit Instruction	BKCMP<>: Continuous execution/BKCMP<>P: Pulse execution
32-bit Instruction	DBKCMP<>: Continuous execution/DBKCMP<>P: Pulse execution
16-bit Instruction	BKCMP>=: Continuous execution/BKCMP>=P: Pulse execution
32-bit Instruction	DBKCMP>=: Continuous execution/DBKCMP>=P: Pulse execution
16-bit Instruction	BKCMP<=: Continuous execution/BKCMP<=P: Pulse execution
32-bit Instruction	DBKCMP<=: Continuous execution/DBKCMP<=P: Pulse execution

Operand	Name	Description	Range	Data Type
S1	Comparand	Comparand, or number of the element that stores the comparand	-	INT, DINT, and array*n
S2	Compared value	Start number of elements that store the source data to be compared	-	INT, DINT, and array*n

D	Destination address	Start number of elements that store the comparison result	-	BOOL and array*n
n	Data count	Number of data entries involved in an operation	1 to 256	INT and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
D	√[1]	√	√	-	-	-	-	-	-
n	-	-	-	√	√	√	√	-	-

NOTE

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- [1]: The X element is not supported.

Function and Instruction Description

The BKCMP# instruction compares the n data entries (16-bit or 32-bit) starting from [S1] with the n data entries (16-bit or 32-bit) starting from [S2] and stores the comparison result in n units (16-bit or 32-bit) starting from [D].

Take the BKCMP> instruction as an example.



A signed constant (16-bit or 32-bit) can be directly specified in [S1].

Take the BKCMP> instruction as an example.



M8333 is set to ON when all of the n comparison results starting from [D] are ON.

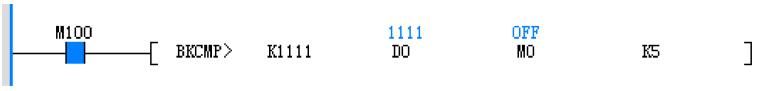
An error is reported in the following cases, in which the instruction is not executed:

1. The elements starting from [S1], [S2], or [D] are beyond the corresponding element range.

2. 32-bit variables are used in a 16-bit instruction.

3. You need to use 32-bit instructions (such as DBKCMP=, DBKCMP>, and DBKCMP<) to compare 32-bit variables.

Instruction Example



Element Name	Data Type	Display Format	Current Value
D0	16-bit INT	Dec	1111
D1	16-bit INT	Dec	-2222
D2	16-bit INT	Dec	0
D3	16-bit INT	Dec	500
D4	16-bit INT	Dec	3333
M100	BOOL	Bin	ON
	16-bit INT	Dec	
M0	BOOL	Bin	OFF
M1	BOOL	Bin	ON
M2	BOOL	Bin	ON
M3	BOOL	Bin	ON
M4	BOOL	Bin	OFF

5.9 String Instructions

5.9.1 List of Instructions

The following table lists the string instructions.

Category	Instruction	Description	Language Support
String instructions	STR	Conversion from integer into string	LD
	STRMOV	String assignment	LD
	VAL	Conversion from string into integer	LD
	ESTR	Conversion from binary floating-point into string	LD

Category	Instruction	Description	Language Support
	EVAL	Conversion from string into binary floating-point	LD
	$ADD	String combination	LD
	LEN	Detect string length	LD and LiteST
	INSTR	String search	LD
	RIGHT	Retrieve from string right	LD and LiteST
	LEFT	Retrieve from string left	LD and LiteST
	MIDR	Retrieve from any position of string	LD
	MIDW	String replacement at any position	LD
	$MOV	String transfer	LD
	ClearString	Clear string	LiteST
	CONCAT	Concatenate and output two strings	LiteST
	DELETE	Delete specific-length string at specified position and output processed string	LiteST
	FIND	Search input source string for position of target string	LiteST
	REPLACE	Replace characters in string	LiteST
	INSETR	Insert string	LiteST

NOTE

The STR, STRMOV, VAL, ESTR, EVAL, $ADD, LEN, INSTR, RIGHT, LEFT, MIDR, MIDW, and $MOV instructions are supported only when the firmware version is V6.3.0.3 and later.

5.9.2 STR

The STR instruction converts integers into character strings (ASCII codes).

STR -- Conversion from integer into string

16-bit Instruction	STR: Continuous execution/STRP: Pulse execution
32-bit Instruction	DSTR: Continuous execution/DSTRP: Pulse execution

Operand	Name	Description	Range	Data Type
S1	Number of digits to be converted	Start number of elements that store the total number of digits contained in the integer to be converted	-	INT, DINT, and array*2
S2	Data to be converted	Number of the element that stores the integer to be converted	-	INT and DINT
D	Output	Start number of elements that store the character string after conversion	-	INT, DINT, and array*indeterminate

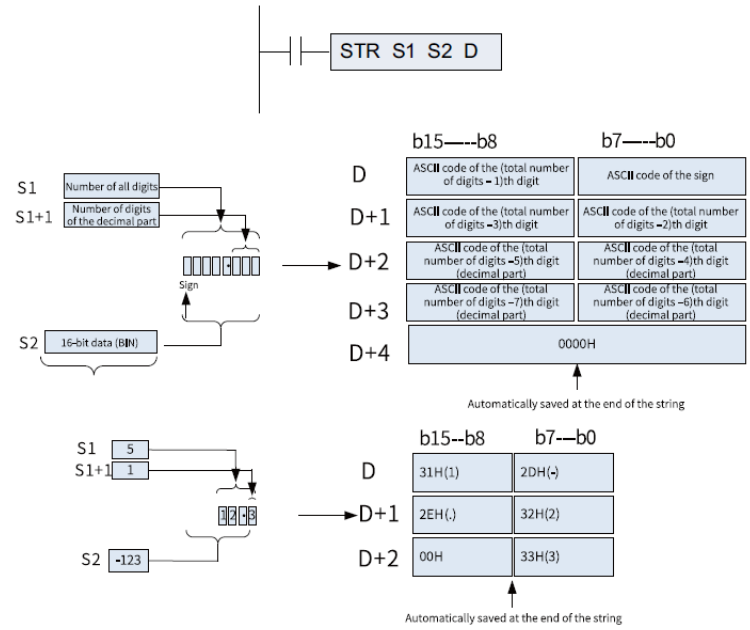
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

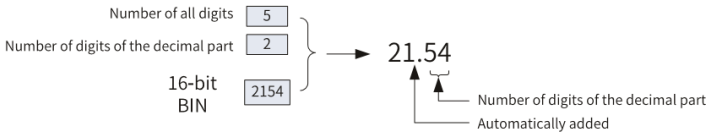
Function and Instruction Description

1. 16-bit operation (STR and STRP)

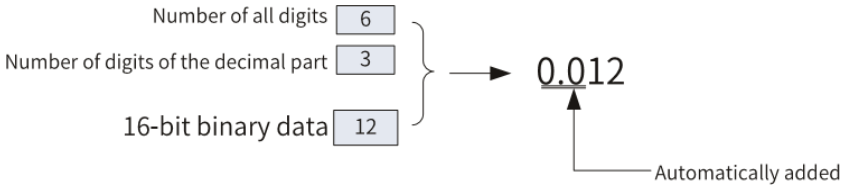
   The STR or STRP instruction inserts a decimal point in the position specified by [S1] and [S1+1], converts the 16-bit binary number in [S2] into a character string, and stores the result in elements starting from [D].



- The total number of digits specified in [S1] ranges from 2 to 8.
- The number of digits of the decimal part specified in [S1+1] ranges from 0 to 5. Make sure that the following condition is met: [S1+1] ≤ [S1] - 3.
- The 16-bit binary number to be converted ranges from -32768 to +32767. The character string after conversion is stored in elements starting from [D], as shown in the following figure.
- The sign bit stores space (20H) when the 16-bit binary number in [S2] is positive or - (2DH) when it is negative.
- When the number of digits of the decimal part in [S1+1] is set to any value other than 0, the decimal point . (2EH) is automatically added in the "number of digits of the decimal part + 1"th digit. No decimal point is inserted when the value in [S1+1] is 0.

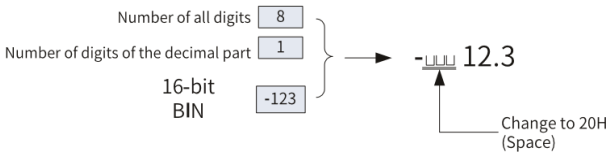


If the number of digits of the decimal part in [S1+1] is greater than the number of digits of 16-bit binary data in [S2], data is automatically aligned to the right and 0 (30H) is automatically added on the left during conversion.



If the number of all digits in [S1] excluding the sign and decimal point is greater than the number of digits of 16-bit binary data in [S2], space (20H) is inserted between the sign and the numeric value.

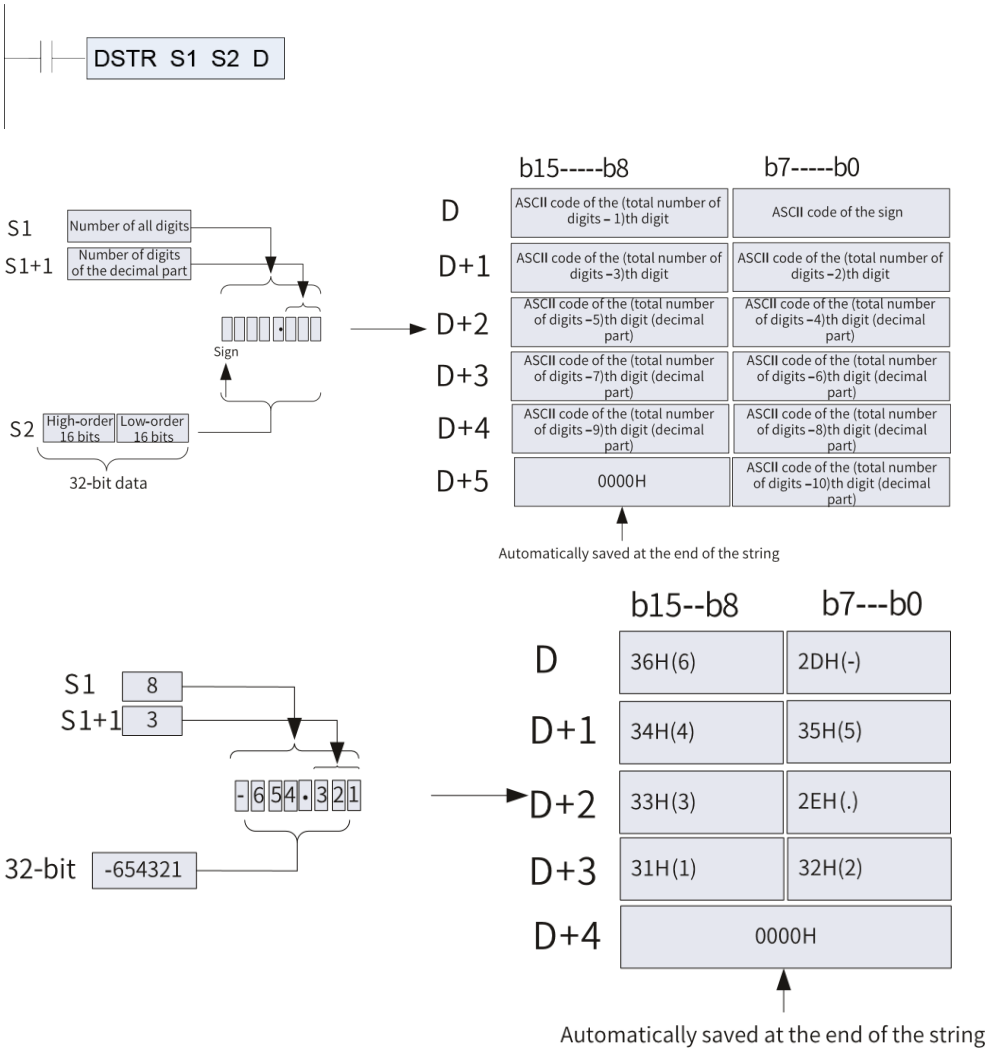
If the number of digits of 16-bit binary data in [S2] is larger, an error occurs.



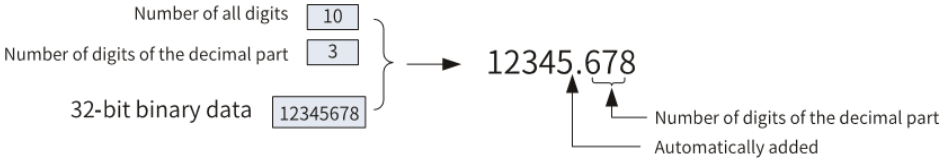
00H indicating the end of a character string is automatically added at the end of a converted character string.

When the total number of digits is even, 0000H is stored in the element after the one that stores the last character. When the total number of digits is odd, 00H is stored in the high-order byte (8 bits) of the element that stores the last character.

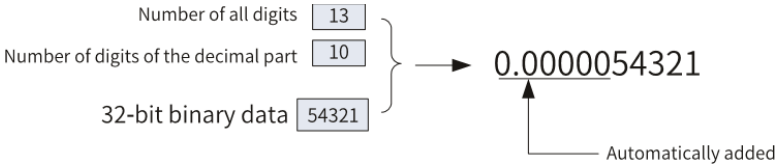
2. 32-bit operation (DSTR and DSTRP)
   The DSTR or DSTRP instruction inserts a decimal point in the position specified by [S1] and [S1+1], converts the 32-bit binary number in [S2+1, S2] into a character string, and stores the result in elements starting from [D].



- The total number of digits specified in [S1] ranges from 2 to 13.
- The number of digits of the decimal part specified in [S1+1] ranges from 0 to 10. Make sure that the following condition is met: [S1+1] ≤ [S1] - 3.
- The 32-bit binary number to be converted ranges from -2,147,483,648 to +2,147,483,647. The character string after conversion is stored in elements starting from [D], as shown in the following figure.
- The sign bit stores space (20H) when the 32-bit binary number in [S2] is positive or - (2DH) when it is negative.
- When the number of digits of the decimal part in [S1+1] is set to any value other than 0, the decimal point . (2EH) is automatically added in the "number of digits of the decimal part + 1"th digit. No decimal point is inserted when the value in [S1+1] is 0.

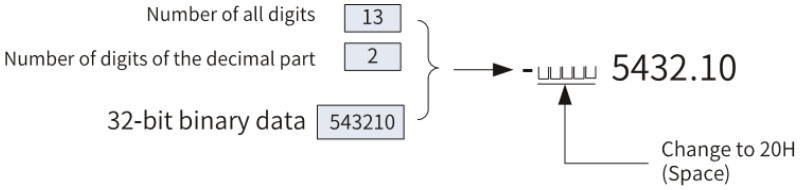


If the number of digits of the decimal part in [S1+1] is greater than the number of digits of 16-bit binary data in [S2], data is automatically aligned to the right and 0 (30H) is automatically added on the left during conversion.



If the number of all digits in [S1] excluding the sign and decimal point is greater than the number of digits of 32-bit binary data in [S2], space (20H) is inserted between the sign and the numeric value.

If the number of digits of 32-bit binary data in [S2] is larger, an error occurs.



00H indicating the end of a character string is automatically added at the end of a converted character string.

When the total number of digits is even, 0000H is stored in the element after the one that stores the last character. When the total number of digits is odd, 00H is stored in the high-order byte (8 bits) of the element that stores the last character.

Errors

An operation error occurs in the following cases:

- The value in [S1] is out of the following range.

Operation	Value Range
16-bit operation	2 to 8
32-bit operation	2 to 13

- The value in [S1+1] is out of the following range.

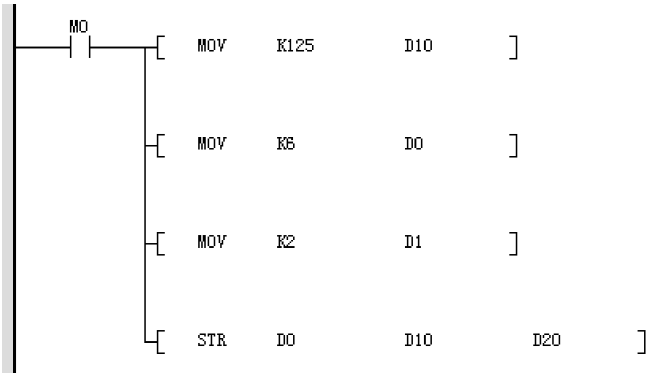
Operation	Value Range
16-bit operation	0 to 5
32-bit operation	0 to 10

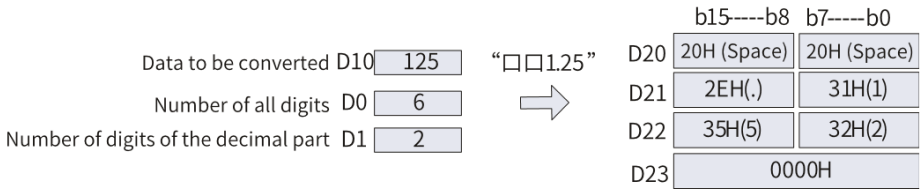
- The relationship between the number of all digits specified in [S1] and the number of digits of the decimal part specified in [S1+1] does not meet the following requirements:
  - Total number of digits - 3 ≥ Number of digits of the decimal part
  - Total number of digits ([S1]) including the digits for the sign and the decimal point < Number of digits of the binary data stored in [S2]

- The elements starting from [D] for storing the character string are beyond the corresponding element range.

Instruction Example

When M0 is ON, the 16-bit binary number in D10 is converted to a character string in accordance with the number of digits specified by D0 and D1. The result is stored in D20 to D23.





5.9.3 STRMOV

The STRMOV instruction directly assigns character strings.

Instruction	Name	LD Expression
STRMOV	String assignment	STRMOV ??? ???

16-bit Instruction	STRMOV: Continuous execution/STRMOVP: Pulse execution			
32-bit Instruction	-			
Operand	**Name**	**Description**	**Range**	**Data Type**
S	Character string data	Character string data to be assigned	1 to 127	STRING
D	Memory register	Target memory register	-	INT, array*strlen [1]

NOTE

[1]: strlen indicates the string length. One character occupies one byte.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	-	-	-	string[1]
D	-	-	-	√	√	-	-	-	-

NOTE

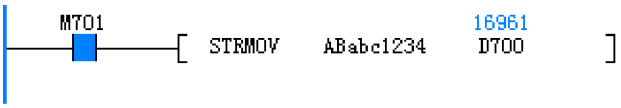
[1]: Only constants of the data type STRING can be entered directly.

Function and Instruction Description

The STRMOV instruction supports direct input of character strings. It can be used to receive and transmit character string data in communication.

The character string data is stored in sequence.

Instruction Example



	Element Name	Data Type	Display Format	Current Value
1	D700	16-bit INT	Hex	0x4241
2	D701	16-bit INT	Hex	0x6261
3	D702	16-bit INT	Hex	0x3163
4	D703	16-bit INT	Hex	0x3332
5	D704	16-bit INT	Hex	0x34
6	D705	16-bit INT	Dec	0
7	D706	16-bit INT	Dec	0

5.9.4 VAL

The VAL instruction converts character strings (ASCII codes) into integers.

VAL -- Conversion from string into integer

16-bit Instruction	VAL: Continuous execution/VALP: Pulse execution			
32-bit Instruction	DVAL: Continuous execution/DVALP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data to be converted	Start number of elements that store the string to be converted	-	INT, DINT, and array*n

D1	Number of bits to be converted	Start number of elements that store the number of bits in the string to be converted	-	INT, DINT, and array*2
D2	Output	Converted integer	-	INT and DINT

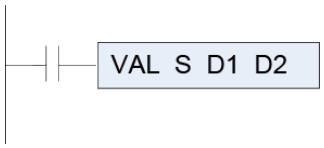
The following table lists soft elements in the instruction.

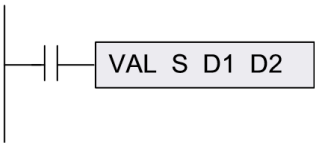
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D1	-	-	-	√	√	-	-	-	-
D2	-	-	-	√	√	-	-	-	-

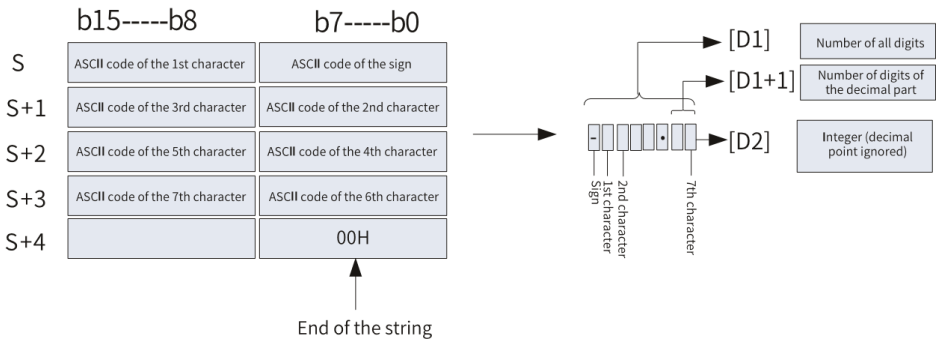
Function and Instruction Description

1. 16-bit operation (VAL and VALP)

- The VAL or VALP instruction converts the string stored in elements starting from [S] into a 16-bit binary number. The total number of digits of the obtained binary data is stored in [D1], the number of digits of the decimal part is stored in [D1+1], and the binary data is stored in [D2].
- During the conversion, the data stored within the range from [S] to the element that stores 00H is handled as a character string in the unit of byte.
- If the ASCII code of the sign is equal to 0x20, the converted data is a positive number. If the ASCII code of the sign is equal to 0x2D, the converted data is a negative number.







- String data to be converted
  The number of characters in and the value range (decimal point ignored) of the character string to be converted must meet the requirements in the following table.

Item	Value Range
Total number of characters	2 to 8
Number of characters of the decimal part	0 to 5
Value range (decimal point ignored)	-32,768 to +32,767 (For example, 123.45 is processed as 12,345.)

The following table describes the types of characters that can be used in the character string to be converted.

Item	Character Type
Positive number	Space (20H)
Negative number	- (2DH)
Decimal point	. (2EH)
Digit	0 (30H) to 9 (39H)

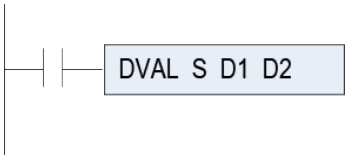
- [D1] stores the total number of digits, including the digits, sign, and decimal point.
- [D1+1] stores the number of digits of the decimal part, that is, the characters to the right of the decimal point . (2EH).
- [D2] stores the 16-bit binary data converted from a character string with the decimal point ignored.

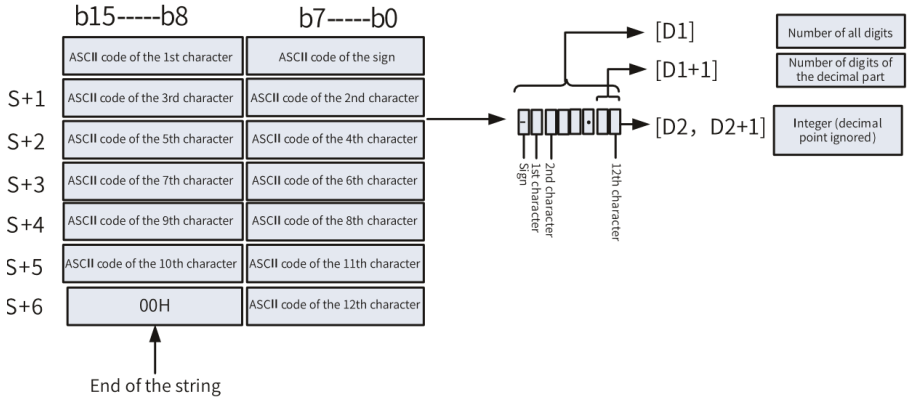
In the character string, space (20H) and 0 (30H) characters between the sign and the first number other than 0 are ignored in the conversion to 16-bit binary data.



2. 32-bit operation (DVAL and DVALP)

- The DVAL or DVALP instruction converts the string stored in elements starting from [S] into a 32-bit binary number. The total number of digits of the obtained binary data is stored in [D1], the number of digits of the decimal part is stored in [D1+1], and the binary data is stored in [D2+1, D2].
- During the conversion, the data stored within the range from [S] to the element that stores 00H is handled as a character string in the unit of byte.
- If the ASCII code of the sign is equal to 0x20, the converted data is a positive number. If the ASCII code of the sign is equal to 0x2D, the converted data is a negative number.





- String data to be converted
  The number of characters in and the value range (decimal point ignored) of the character string to be converted must meet the requirements in the following table.

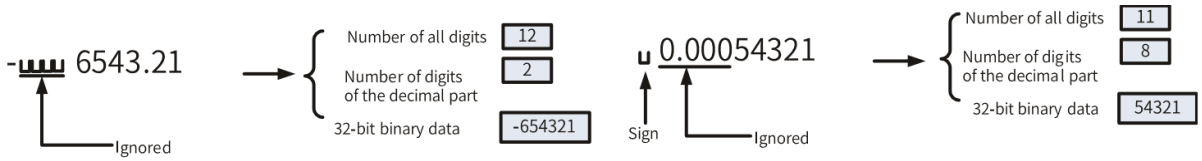
Item	Value Range
Total number of characters	2 to 8
Number of characters of the decimal part	0 to 10
Value range (decimal point ignored)	-2,147,483,648 to +2,147,483,647 (For example, 123.45 is processed as 12345.)

  The following table describes the types of characters that can be used in the character string to be converted.

Item	Character Type
Positive number	Space (20H)
Negative number	- (2DH)
Decimal point	. (2EH)
Digit	0 (30H) to 9 (39H)

- [D1] stores the total number of digits, including the digits, sign, and decimal point.
- [D1+1] stores the number of digits of the decimal part, that is, the characters to the right of the decimal point . (2EH).
- [D2+1, D2] stores the 32-bit binary data converted from a character string with the decimal point ignored.

In the character string, space (20H) and 0 (30H) characters between the sign and the first number other than 0 are ignored in the conversion to 32-bit binary data.



NOTE

- The sign data, spaces (20H) or - (2DH), must be stored in the first byte (low-order 8 bits of the elements in [S]).
- Only digits 0 (30H) to 9 (39H), spaces (20H), and decimal points (2EH) can be stored in the ASCII code data area within the range from the second byte of [S] to the string end 00H. An operation error occurs if - (2DH) is stored after the second byte.

Errors

An operation error occurs in the following cases:

- The total number of digits of the character string to be converted is out of the following range.

Operation	Value Range
16-bit operation	2 to 8
32-bit operation	2 to 13

- The number of digits of the decimal part of the character string to be converted is out of the following range.

Operation	Value Range
16-bit operation	0 to 5
32-bit operation	0 to 10

- The relationship between the number of all digits and the number of digits of the decimal part of the character string to be converted (starting from [S]) does not meet the following requirements:
  - Total number of digits - 3 ≥ Number of digits of the decimal part
  - The sign is set to any ASCII code other than space (20H) or - (2DH).
  - A digit of a number is set to any ASCII code other than 0 (30H) to 9 (39H) or a decimal point . (2EH).
  - The character string (starting from [S]) to be converted contains multiple decimal points . (2EH).
- The binary data after conversion is out of the following range.

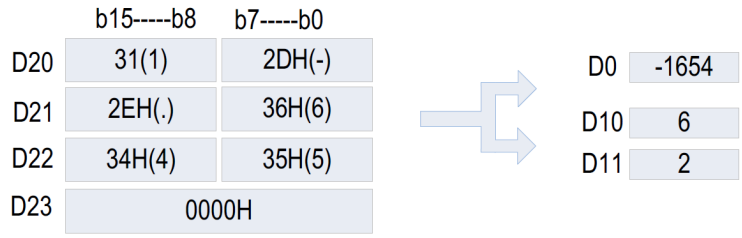
Operation	Value Range
16-bit operation	-32,768 to +32,767
32-bit operation	-2,147,483,648 to +2,147,483,647

- 00H does not exist in elements starting from [S].

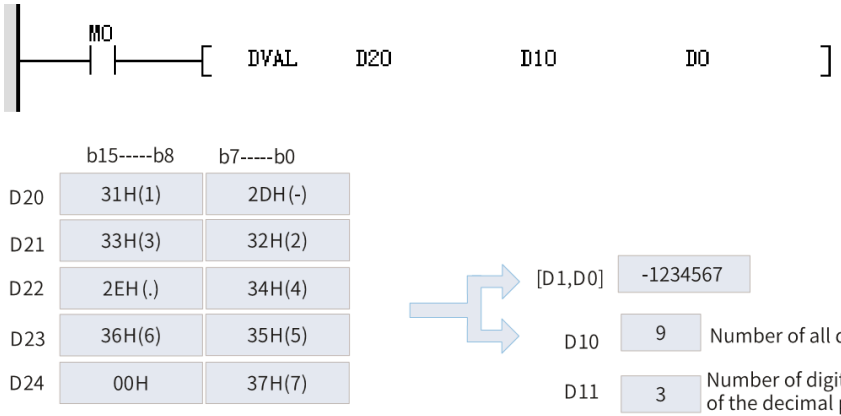
Instruction Example

- When M0 is ON, the character string data stored in D20 to D22 is regarded as an integer value, and it is converted into a binary value and stored in D0.





- When M0 is ON, the character string data stored in D20 to D24 is regarded as an integer value, and it is converted into a binary value and stored in [D1, D0].



5.9.5 ESTR

The ESTR instruction converts binary floating-point data (real number) into a character string (ASCII codes) with specified bits.

16-bit Instruction	-			
32-bit Instruction	DESTR: Continuous execution/DESTRP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Operand	Start number of elements that store the binary floating-point number to be converted	-	REAL
S2	Start number	Start number of elements that store the display format of the value to be converted	-	INT, array*3
D	Result	Start number of elements that store the character string after conversion	-	DINT, array*n

The following table lists soft elements in the instruction.

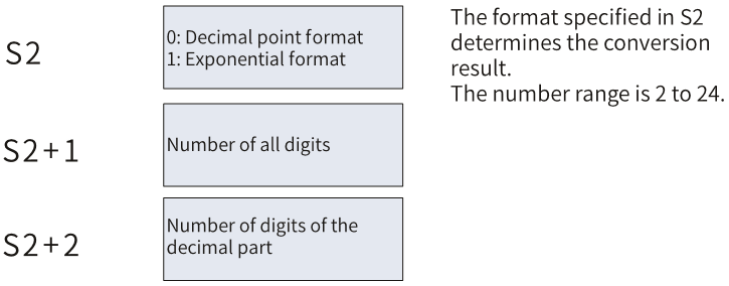
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

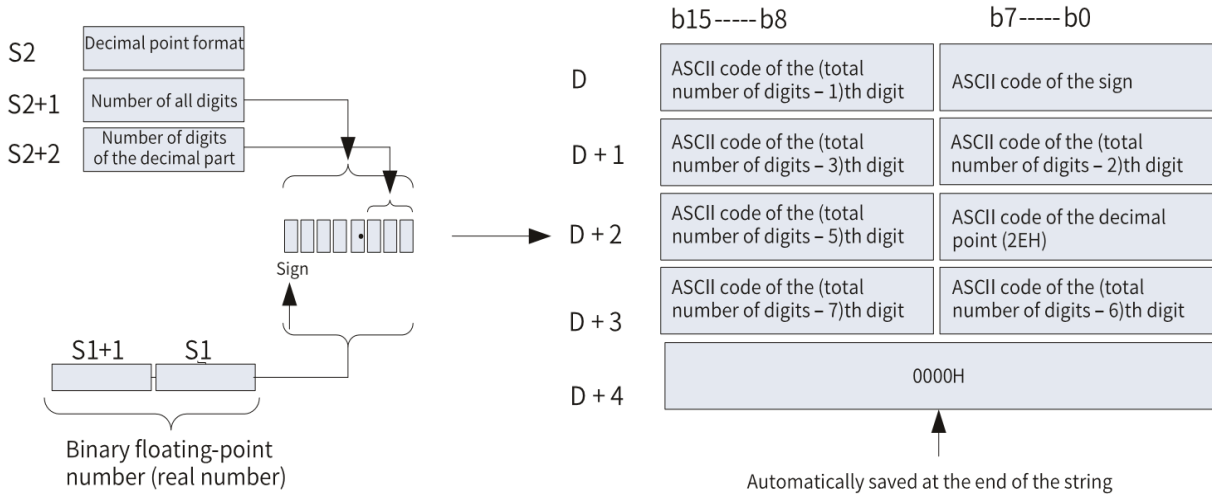
Function and Instruction Description

32-bit operation (DESTR)

The DESTR instruction converts the binary floating-point number in [S1+1, S] into a character string based on the content of [S2, S2+1, S2+2] and stores the result in elements starting from D.



- Decimal point format



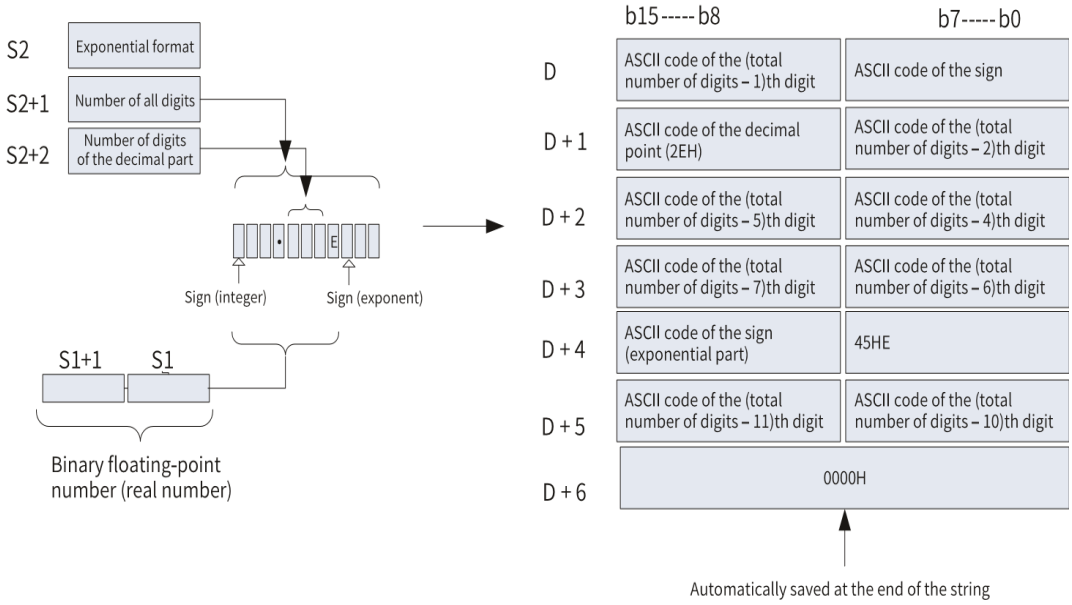
The total number of bits (maximum of 24) is specified in [S2+1] based on the following rules:

- When the number of bits of the decimal part is 0, the total number of bits is greater than or equal to 2.
- When the number of bits of the decimal part is not 0, the total number of bits is greater than or equal to the number of bits of the decimal part plus 3.

The number of bits of the decimal part specified in [S2+2] ranges from 0 to 7. Meanwhile, it cannot be greater than the total number of bits minus 3.

- Exponent format

- Exponent format



The total number of bits (maximum of 24) is specified in [S2+1] based on the following rules:

- When the number of bits of the decimal part is 0, the total number of bits is greater than or equal to 6.
- When the number of bits of the decimal part is not 0, the total number of bits is greater than or equal to the number of bits of the decimal part plus 7.

The number of bits of the decimal part specified in [S2+2] ranges from 0 to 7. Meanwhile, it cannot be greater than the total number of bits minus 3.

Errors

An operation error occurs in the following cases:

- The value in [S1] is out of range. (Error code: K6706)
- The value in [S2] is neither 0 nor 1. (Error code: K6706)
- The total number of bits specified in [S2+1] is out of range. (Error code K6706)
  Decimal point format

  When the number of bits of the decimal part is 0, the total number of bits is greater than or equal to 2.

  When the number of bits of the decimal part is not 0, the total number of bits is greater than or equal to the number of bits of the decimal part plus 3.

  Exponent format

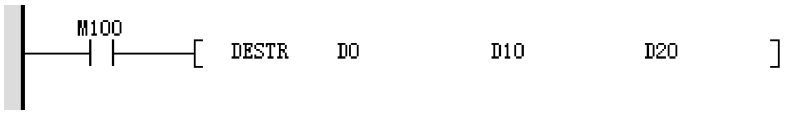
  When the number of bits of the decimal part is 0, the total number of bits is greater than or equal to 6.

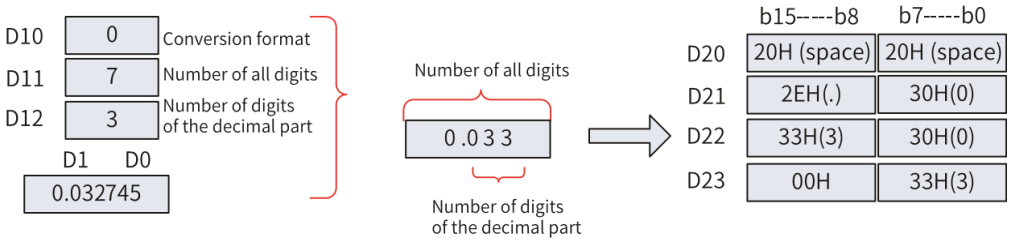
  When the number of bits of the decimal part is not 0, the total number of bits is greater than or equal to the number of bits of the decimal part plus 7.
- The number of bits of the decimal part specified in [S2+2] is out of range. (Error code: K6706)
  Decimal point format Number of bits of the decimal part ≤ Total number of bits - 3

  Exponent format Number of bits of the decimal part ≤ Total number of bits - 7
- The elements starting from [D] for storing the character string are out of range. (Error code: K6705)
- The number of bits in the conversion result exceeds the specified total number of bits. (Error code: K6705)

Instruction Example

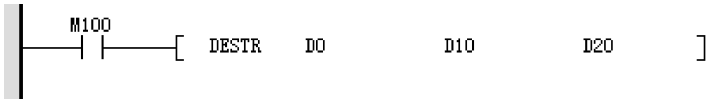
- When M100 is ON, the binary floating-point number in D0 and D1 is converted based on the content (decimal point format) of D10 to D12. The result is stored in elements starting from D20.

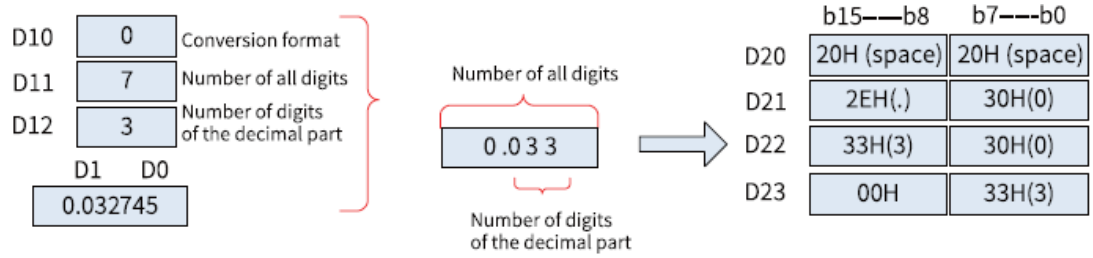




Element Name	Data Type	Display Format	Current Value
D0	Float	Dec	0.032745
D10	16-bit INT	Dec	0
D11	16-bit INT	Dec	7
D12	16-bit INT	Dec	3
M100	BOOL	Bin	ON
D20	16-bit INT	Hex	0x2020
D21	16-bit INT	Hex	0x2E30
D22	16-bit INT	Hex	0x3330
D23	16-bit INT	Hex	0x33

- When M100 is ON, the binary floating-point number in D0 and D1 is converted based on the content (exponential format) of D10 to D12. The result is stored in elements starting from D20.





Element Name	Data Type	Display Format	Current Value
D0	Float	Dec	0.032745
D10	16-bit INT	Dec	1
D11	16-bit INT	Dec	12
D12	16-bit INT	Dec	4
M100	BOOL	Bin	ON
D20	16-bit INT	Hex	0x2020
D21	16-bit INT	Hex	0x2E33
D22	16-bit INT	Hex	0x3732
D23	16-bit INT	Hex	0x3534
D24	16-bit INT	Hex	0x2D45
D25	16-bit INT	Hex	0x3230
D26	16-bit INT	Hex	0x0

5.9.6 EVAL

The EVAL instruction converts a character string (ASCII codes) into binary floating-point data.

EVAL -- Conversion from string into floating-point number

16-bit Instruction	-			
32-bit Instruction	DEVAL: Continuous execution/DEVALP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Operand	Start number of elements that store the character string to be converted	-	DINT, array*indeterminate
D	Result	Start number of elements that store the binary floating-point number after conversion	-	REAL

The following table lists soft elements in the instruction.

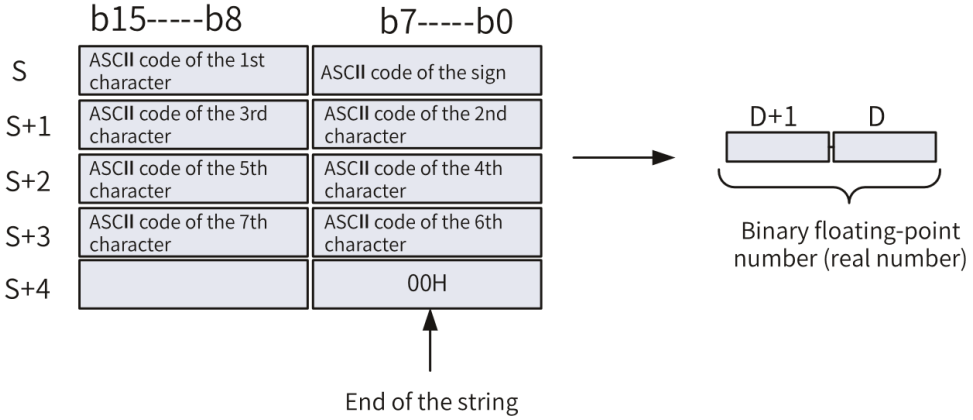
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

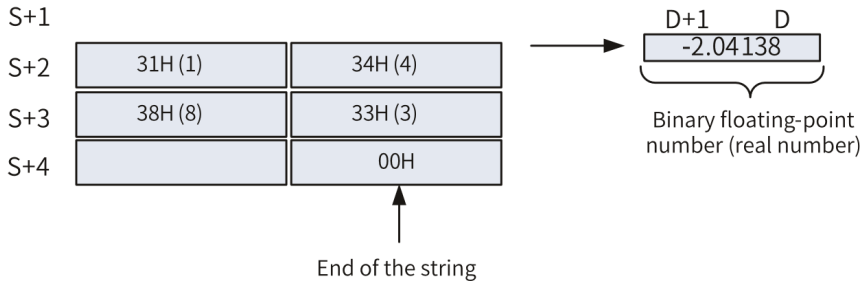
32-bit Operation (DEVAL)

The DEVAL instruction converts the character string stored in elements starting from [S] into a binary floating-point number and stores the result in [D+1, D].

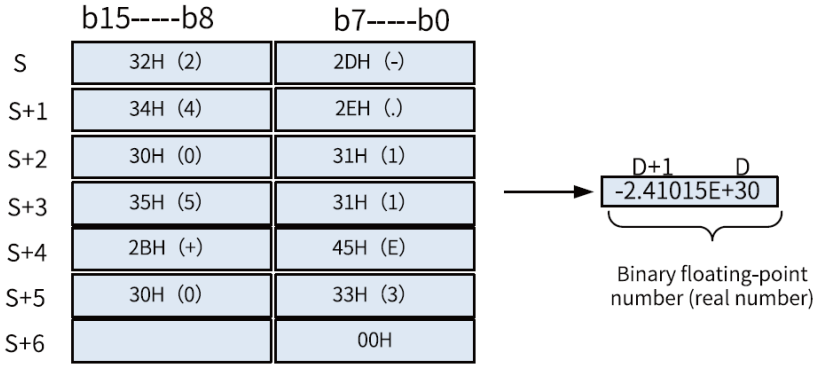
A specified character string, either in the decimal point format or exponent format, can be converted into binary floating-point data.



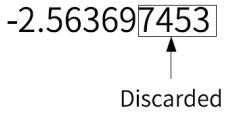
- Decimal point format

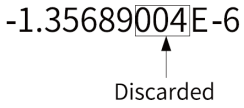


- Exponent format



When a character string to be converted into binary floating-point data specified by [S] has 7 bits or more excluding the sign, decimal point, and exponent part, the bits after the 7th bit are discarded.

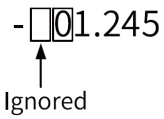




In the decimal point format, when 2BH (+) is specified as the sign in [S] or when the sign is omitted, a character string is converted into a positive value. When 2DH (-) is specified as the sign, a character string is converted into a negative value.

In the exponent format, when 2BH (+) is specified as the sign or when the sign is omitted, a character string is converted into a positive exponent. When 2DH (-) is specified as the sign, a character string is converted into a negative exponent.

If the source string specified in [S] contains 20H (space) or 30H (0) between bits other than the first 0, 20H or 30H is ignored when the string is converted.



The source string can contain a maximum of 24 characters, including 20H (space) and 30H (0).

Related soft elements

Element	Name	Condition	Action
M8020	Zero flag	The conversion result is TRUE (mantissa = 0).	Set the zero flag (M8020) to ON.
M8021	Borrow flag	The absolute value of the conversion result is less than 2^{-126}.	Discard the part in the D value less than the 32-bit minimum absolute value of 2-126 and set the borrow flag (M8021) to ON.
M8022	Carry flag	The absolute value of the conversion result is greater than or equal to 2128.	Discard the part in the D value greater than the 32-bit maximum absolute value of 2128 and set the carry flag (M8022) to ON.

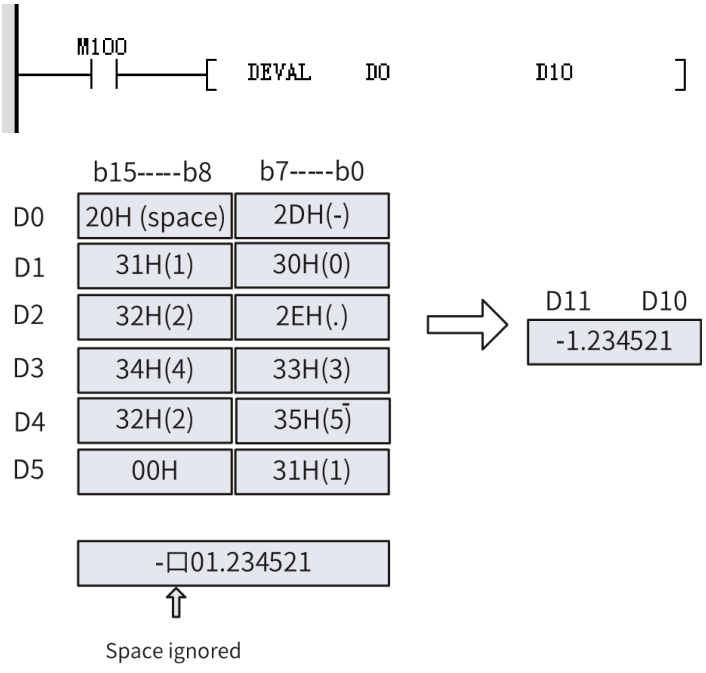
Errors

An operation error occurs in the following cases:

- The integer or decimal part contains characters other than 30H (0) to 39H (9).
- The character string starting from [S] contains two or more decimal points 2EH (.).
- The exponent contains characters other than 45H (E), 2BH (+), and 2DH (-), or multiple exponents exist.
- 00H does not exist in the corresponding element range starting from [S].
- The number of characters after [S] is 0 or more than 24.

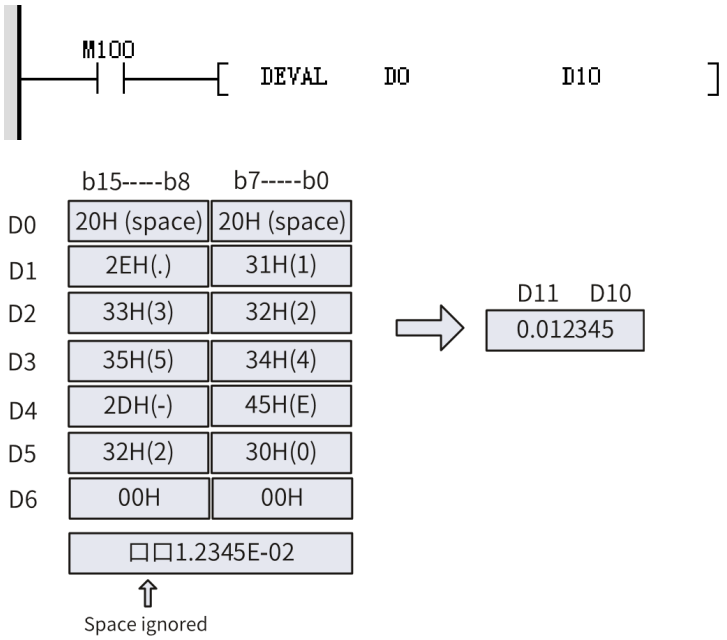
Instruction Example

- When M101 is ON, the character string stored in elements starting from D0 is converted into a binary floating-point number (in decimal point format). The result is stored in D10 and D11.



Element Name	Data Type	Display Format	Current Value
D10	Float	Dec	-1.234521
M100	BOOL	Bin	ON
D0	16-bit INT	Hex	0x202D
D1	16-bit INT	Hex	0x3130
D2	16-bit INT	Hex	0x322E
D3	16-bit INT	Hex	0x3433
D4	16-bit INT	Hex	0x3235
D5	16-bit INT	Hex	0x31

- When M100 is ON, the character string stored in elements starting from D0 is converted into a binary floating-point number (in exponent format). The result is stored in D10 and D11.



Element Name	Data Type	Display Format	Current Value
D10	Float	Dec	0.012345
M100	BOOL	Bin	ON
D0	16-bit INT	Hex	0x2020
D1	16-bit INT	Hex	0x2E31
D2	16-bit INT	Hex	0x3332
D3	16-bit INT	Hex	0x3534
D4	16-bit INT	Hex	0x2D45
D5	16-bit INT	Hex	0x3230
D6	16-bit INT	Hex	0x0

5.9.7 $ADD

The $ADD instruction links a character string to another character string.

16-bit instruction	$ADD: Continuous execution/$ADDP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type

S1	String to be appended with another one	Start number of soft elements that store the source data (character string) to be linked, or a directly specified character string	-	INT, array*n
S2	String to be appended to the source string	Start number of soft elements that store the data (character string) to be linked to the source string, or a directly specified character string	-	INT, array*n
D	Connection result	Start number of soft elements that store the data (character string) after linking	-	INT, array*n

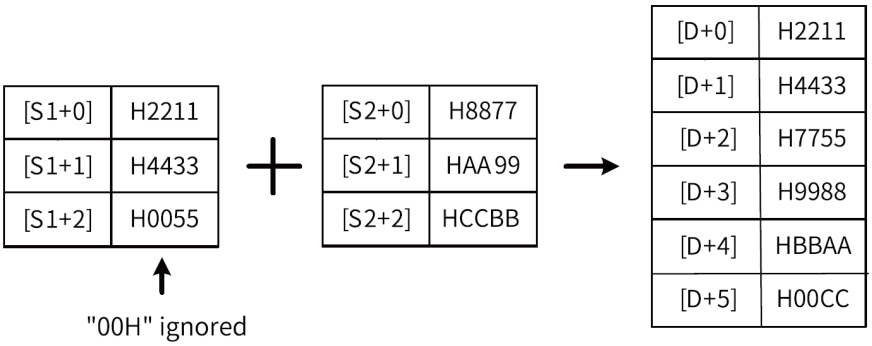
For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function

The string from start address [S2] is appended to the string from start address [S1]. The resulting string is stored in [D].

A character string stored in [S1] or [S2] is organized by byte and ends with the first "00H" byte.



During linking, "00H" indicating the end of a string is ignored and the last character of a character string is linked to the last character of another specified character string. After a character string is linked, "00H" is automatically added at the end.

When the number of characters in the new character string after linking is odd, "00H" is stored in the high-order byte of the soft element that stores the last character.

When the number of characters in the new character string after linking is even, "0000H" is stored in the element after the one that stores the last character.

Errors

An error is returned in the following conditions:

- "00H" is not found within the corresponding element range starting from[S1] or [S2].
- The number of elements required to store the linking result is beyond the element range starting from [D].

Example



Program running flag

- Running: ON
- Stopped: OFF

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x2211
D101	16-bit INT	Hex	0x4433
D102	16-bit INT	Hex	0x55
	16-bit INT	Dec	
D200	16-bit INT	Hex	0x8877
D201	16-bit INT	Hex	0xAA99
D202	16-bit INT	Hex	0xCCBB
D203	16-bit INT	Hex	0xDD
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x2211
R101	16-bit INT	Hex	0x4433
R102	16-bit INT	Hex	0x7755
R103	16-bit INT	Hex	0x9988
R104	16-bit INT	Hex	0xBBAA
R105	16-bit INT	Hex	0xDDCC
R106	16-bit INT	Dec	0

5.9.8 LEN

Function and Instruction Description

The LD expression of the LEN instruction is different from its LiteST expression.

- **LD**

The LEN instruction detects the number of characters (bytes) of a specified character string.

16-bit Instruction	LEN: Continuous execution/LENP: Pulse execution			
32-bit Instruction	---			
Operand	Name	Description	Range	Data Type
S	Checked data	Start number of soft elements that store a character string to be detected	-	INT and array*indeterminate
D	Detection result	Number of the soft element that stores the detected number of	-	INT and array*indeterminate

		characters (bytes) contained in the string		

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

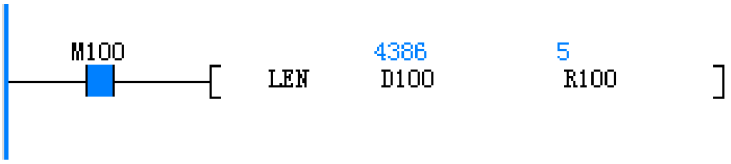
The LEN instruction detects the number of characters in the character string stored in elements starting from [S] and stores the result in [D]. Characters are counted byte by byte within the range from [S] to the first 00H.

An error is reported in the following cases:

1. "00H" is not found within the corresponding element range starting from [S].

2. The detected number of characters is greater than 32,767.

Instruction Example



Element Name	Data Type	Display Format	Comment
D100	INT	Hex	0x1122
D101	INT	Hex	0x3344
D102	INT	Hex	0x55
D103	INT	Hex	0x0
	INT	Hex	

- **LiteST**

Return Value Type	Get string length		
INT			
Operand	Description	Range	Data Type
S1	Source string	-	STRING

The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	-	√	-	-	-	-	-	-	√	-

Function and Instruction Description

The LEN instruction gets the number of characters in a target text string, excluding the end character.

Instruction Example

```
Out_Int1 := LEN(In_Str1);
```

5.9.9 INSTR

The INSTR instruction searches a specified character string within another character string.

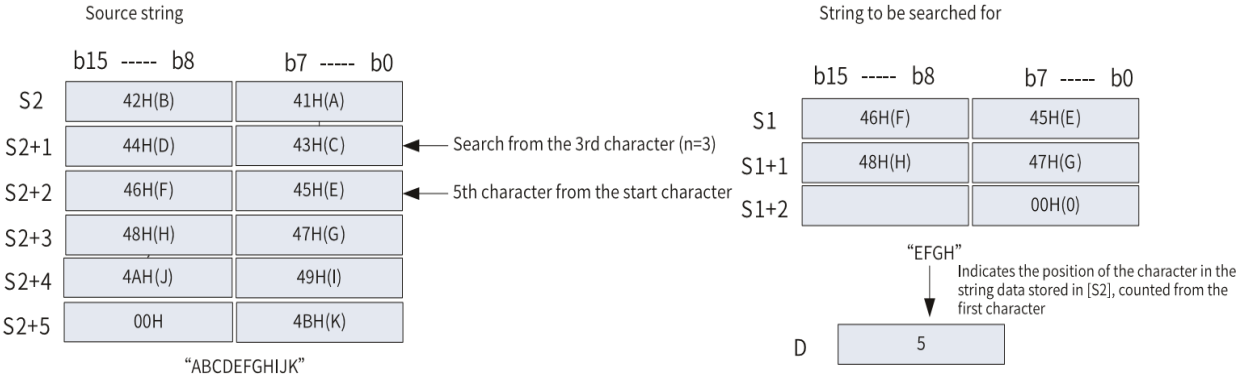
16-bit instruction	INSTR: Continuous execution INSTRP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S1	Source data	Start number of soft elements that store the string to be retrieved	-	INT, array*indeterminate
S2	Retrieval source	Start number of soft elements that store the string from which another string will be retrieved	-	INT, array*indeterminate
D	Retrieval result	Start number of soft elements that store the retrieval result	-	INT
n	Retrieval start point	Position from which a string will be retrieved	1 to 32767	INT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

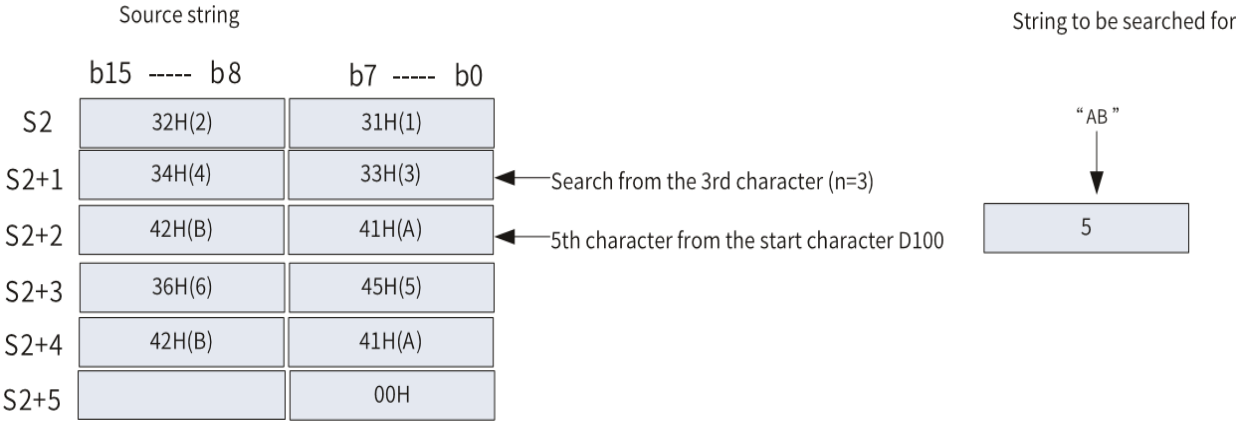
Function

The INSTR instruction searches from the nth character on the left (beginning of a string) of a range of soft elements from the retrieval source [S2], to identify the same string as that stored in soft elements from start address [S1]. Information about the beginning (position of the first matched character on the left) of the retrieved string is stored in [D].



If no portion of the string from start address [S2] matches the string from start address [S1], 0 is stored in [D].

If n (retrieval start point) is negative or 0, the instruction is not executed.



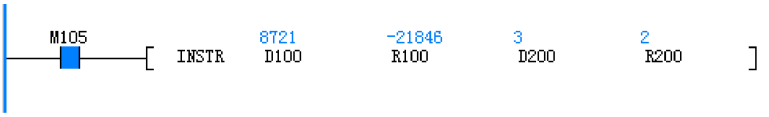
Errors

An error is returned in the following conditions:

1. The value of n (search start position) is greater than the number of characters stored in elements starting from [S2].

2. "00H" is not found within the corresponding element range starting from [S1] or [S2].

Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x2211
D101	16-bit INT	Hex	0x4433
D102	16-bit INT	Hex	0x0
	16-bit INT	Hex	
D200	16-bit INT	Hex	0x3
R200	16-bit INT	Hex	0x2
	16-bit INT	Hex	
R100	16-bit INT	Hex	0xAAAA
R101	16-bit INT	Hex	0x2211
R102	16-bit INT	Hex	0x4433
R103	16-bit INT	Hex	0xAAAA
R104	16-bit INT	Hex	0xAAAA
R105	16-bit INT	Hex	0x0

5.9.10 RIGHT

The LD expression of the RIGHT instruction is different from its LiteST expression.

- **LD**

The RIGHT instruction extracts a specified number of characters from the right end of a specified character string.

16-bit Instruction	RIGHT: Continuous execution/RIGHTP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Source data	Start number of soft elements that store a character string	-	INT and array*indeterminate
D	Extraction result	Start number of soft elements that store the extracted character string	-	INT and array*indeterminate
n	Extracted character count	Number of characters to be extracted	1 to 32767	INT

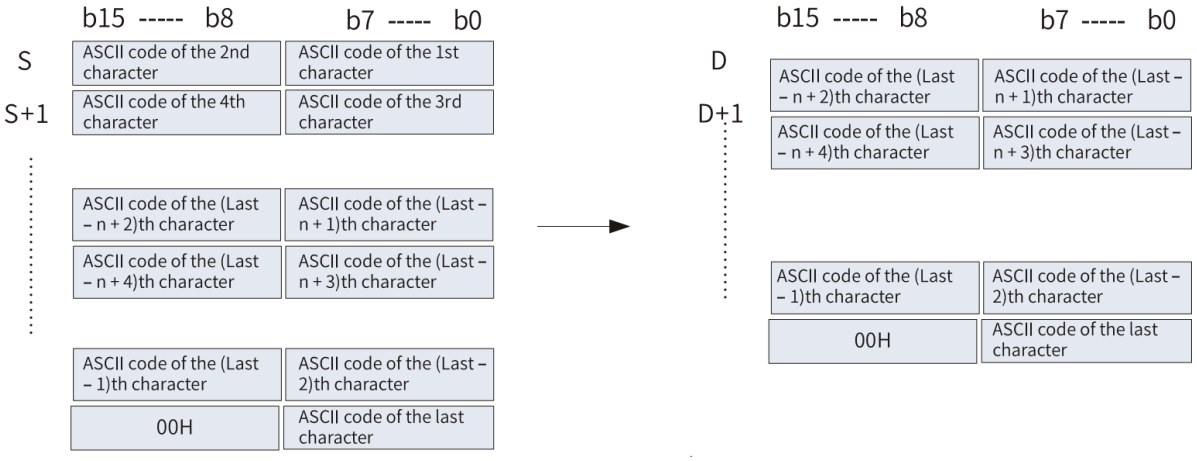
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The RIGHT instruction extracts n characters from the right end (that is, from the end) of the character string stored in elements starting from [S] and stores the extraction result to elements starting from [D].



"00H" is automatically added at the end of the extracted characters.

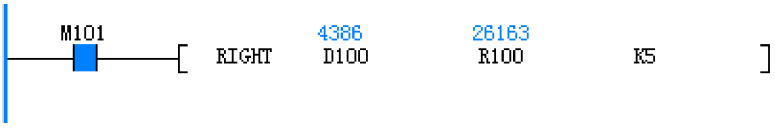
- When the number of extracted characters is odd, "00H" is stored in the high-order byte of the soft element that stores the last character.
- When the number of extracted characters is even, "0000H" is stored in the soft element after the one that stores the last character.
- When the number of bytes to be extracted is 0, "0000H" is stored in [D].

Errors

An error is reported in the following cases:

- "00H" is not found within the corresponding element range starting from [S].
- The number of elements starting from [D] is smaller than the number of elements required to store the extracted n characters.
- n is greater than the number of characters stored in elements starting from [S].
- n is a negative number.

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x5566
D103	16-bit INT	Hex	0x7788
D104	16-bit INT	Hex	0x0
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x6633
R101	16-bit INT	Hex	0x8855
R102	16-bit INT	Hex	0x77
R103	16-bit INT	Hex	0x0

- **LiteST**

Return Value Type	Intercept string from right		
STRING			
Operand	Description	Range	Data Type
S1	Source string	-	STRING
S2	Data length	0 to 255	BYTE, INT, and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	-	√	-	-	-	-	-	-	√	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-

Function and Instruction Description

The RIGHT instruction extracts and returns characters of a specified length from the right (end) of the text string.

Instruction Example

```
Out_Str1 := RIGHT(
    In_Str1, In_Int1);
```

NOTE

- If the value of S2 is 0, the output text string is null.
- If S2 is a constant greater than the length of the input text string, a compilation error is reported.
- If S2 is a variable less than 0, the output text string is null and a running error is reported.
- If S2 is a variable greater than the length of the input text string, the output text string is the original input text string and a running error is reported.

5.9.11 LEFT

The LD expression of the LEFT instruction is different from its LiteST expression.

- **LD**

The LEFT instruction extracts a specified number of characters from the left end of a specified character string.

16-bit Instruction	LEFT: Continuous execution/LEFTP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Source data	Start number of soft elements that store a character string	-	INT and array*indeterminate
D	Extraction result	Start number of soft elements that store the extracted character string	-	INT and array*indeterminate
n	Extracted character count	Number of characters to be extracted	1 to 32767	INT

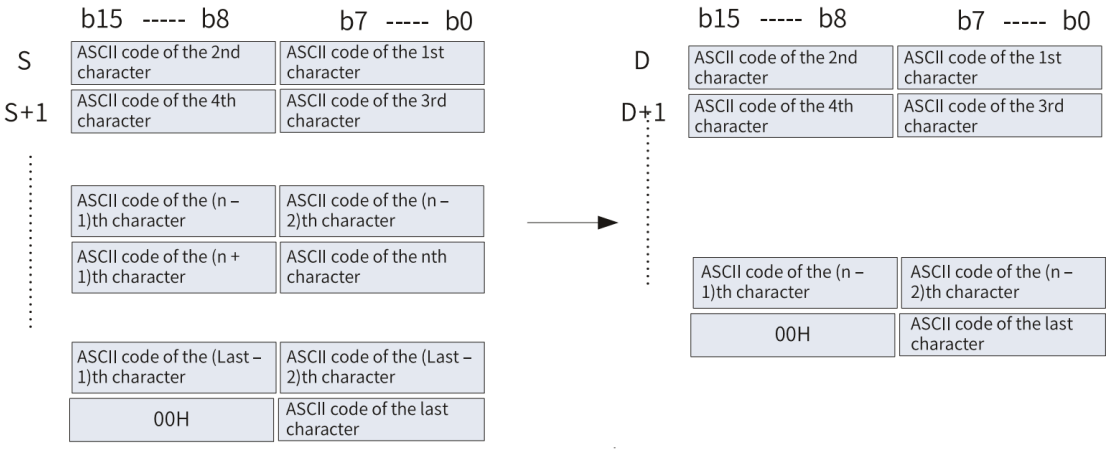
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The LEFT instruction extracts n characters from the left end (that is, from the start) of the character string stored in elements starting from [S] and stores the extraction result to elements starting from [D].



"00H" is automatically added at the end of the extracted characters.

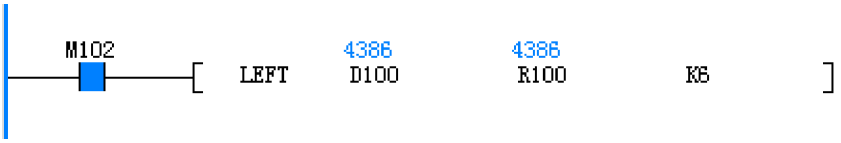
- When the number of extracted characters is odd, "00H" is stored in the high-order byte of the soft element that stores the last character.
- When the number of extracted characters is even, "0000H" is stored in the soft element after the one that stores the last character.
- When the number of bytes to be extracted is 0, "0000H" is stored in [D].

Errors

An error is reported in the following cases:

- "00H" is not found within the corresponding element range starting from [S].
- The number of elements starting from [D] is smaller than the number of elements required to store the extracted n characters.
- n is greater than the number of characters stored in elements starting from [S].
- n is a negative number.

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x5566
D103	16-bit INT	Hex	0x7788
D104	16-bit INT	Hex	0x0
	16-bit INT		
R100	16-bit INT	Hex	0x1122
R101	16-bit INT	Hex	0x3344
R102	16-bit INT	Hex	0x5566
R103	16-bit INT	Hex	0x0

- **LiteST**

Return Value Type	Intercept string from left		
STRING			
Operand	Description	Range	Data Type
S1	Source string	-	STRING
S2	Data length	0 to 255	BYTE, INT, and DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	-	√	-	-	-	-	-	-	√	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-

Function and Instruction Description

The LEFT instruction extracts and returns characters of a specified length from the left (beginning) of the text string.

Instruction Example

```
Out_Str1 := LEFT(
        In_Str1, In_Int1);
```

NOTE

- If the value of S2 is 0, the output text string is null.
- If S2 is a constant greater than the length of the input text string, a compilation error is reported.
- If S2 is a variable less than 0, the output text string is null and a running error is reported.
- If S2 is a variable greater than the length of the input text string, the output text string is the original input text string and a running error is reported.

5.9.12 MIDW

The MIDW instruction replaces the characters in arbitrary positions of a specified character string with characters in another specified character string.

16-bit instruction	MIDW: Continuous execution MIDWP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S1	Source data	Start number of soft elements that store the string that will replace another string	-	INT, array*indeterminate
D	Replacement result	Start number of soft elements that store the string after replacement	-	INT, array*indeterminate
S2	Replacement position	Start number of soft elements that indicate the start position of replacement and the number of replaced characters S2: Initial character to be replaced S2+1: Number of characters to be replaced	-	INT, array*2

For soft elements, see the following table.

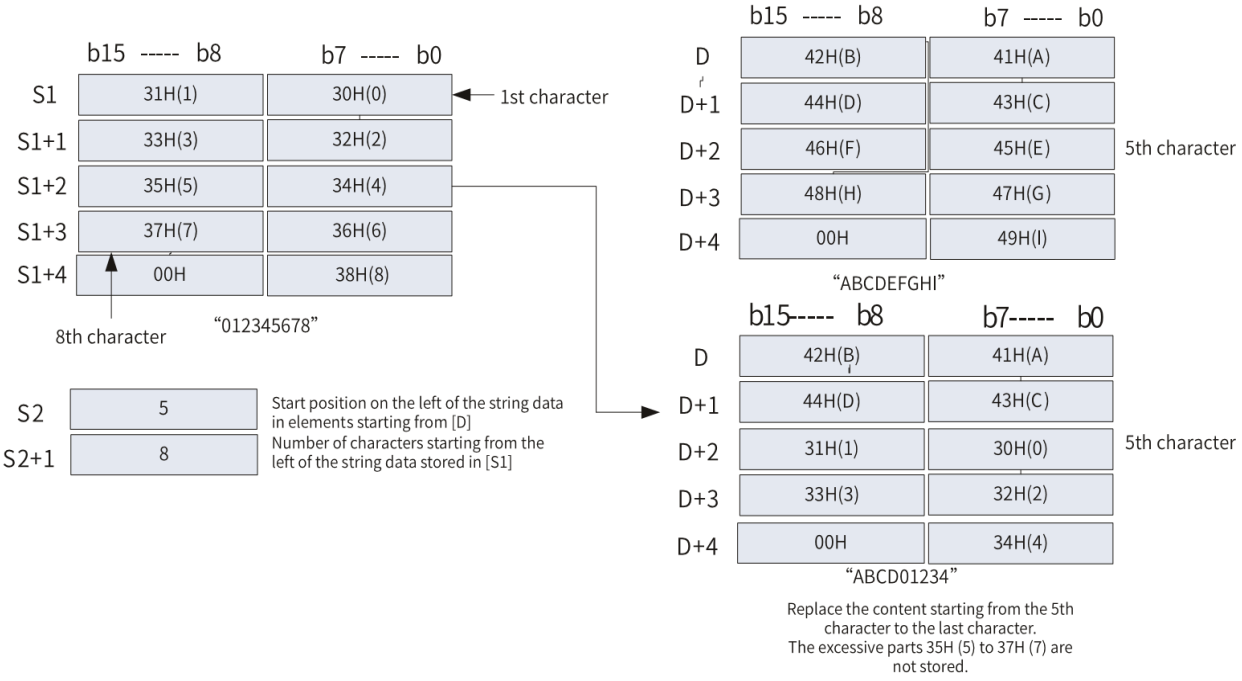
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S2	-	-	-	√	√	-	-	-	-

Function

The MIDW instruction extracts [S2+1] characters from the left (that is, the start) of the character string stored in elements starting from [S1] and stores the extracted data to the position specified by [S2] of the character string stored in elements starting from [D].

- The source string starts with [S1] and ends with 00H.
- When the value in [S2+1] is 0, the instruction is not executed.
- When the value in [S2+1] is -1, all characters of the string from start address [S1] replace those of the string from start address [D].
- If the value in [S2+1] exceeds the number of characters starting from the character specified by [S2] in elements starting from [D], data is stored up to the last character in elements starting from [D], and redundant characters of the source string are discarded.

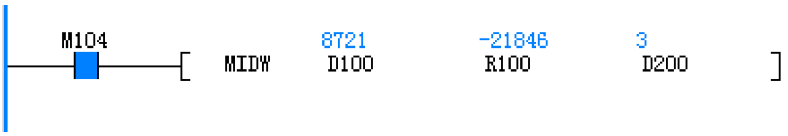


Errors

An error is returned in the following conditions:

- "00H" is not found within the corresponding element range starting from [S1] or [D].
- The value specified in [S2] is greater than the number of characters of the character string stored in elements starting from [D].
- The value specified in [S2] is negative.
- The value specified in [S2+1] is -2 or less.
- The value specified in [S2+1] exceeds the number of characters stored in elements starting from [S1].

Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x2211
D101	16-bit INT	Hex	0x4433
D102	16-bit INT	Hex	0x6655
D103	16-bit INT	Hex	0x0
D200	16-bit INT	Hex	0x3
D201	16-bit INT	Hex	0x4
D202	16-bit INT	Hex	0x0
R100	16-bit INT	Hex	0xAAAA
R101	16-bit INT	Hex	0x2211
R102	16-bit INT	Hex	0x4433
R103	16-bit INT	Hex	0xAAAA
R104	16-bit INT	Hex	0xAAAA
R105	16-bit INT	Hex	0x0

5.9.13 MIDR

The MIDR instruction extracts a specified number of characters from arbitrary positions of a specified character string.

16-bit instruction	MIDR: Continuous execution MIDRP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S1	Source data	Start number of soft elements that store a string	-	INT, array*indeterminate
D	Retrieval result	Start number of soft elements that store a retrieved string	-	INT, array*indeterminate
S2	Reading position	Start number of soft elements that indicate the start position of reading and the number of read characters S2: Position of the initial character to be read S2+1: Number of read characters	-	INT, array*2

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-

Function

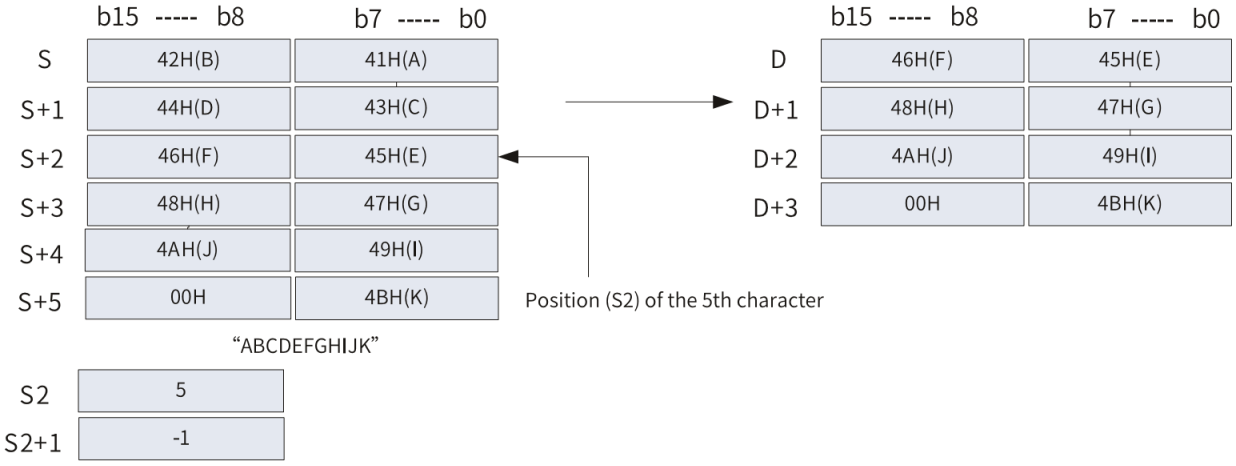
A number (indicated by [S2+1]) of characters starting from the character indicated by [S2] are retrieved at the left end (beginning) of a string from start address [S1]. The result is stored in soft elements from start address [D].

- When the number of extracted characters specified in [S2+1] is odd, "00H" is stored in the high-order byte of the element that stores the last character.
- When the number of extracted characters specified in [S2+1] is even, "0000H" is stored in the element after the one that stores the last character.

The source string starts with [S1] and ends with 00H.

When the value in [S2+1] is 0, the instruction is not executed.

When the value in [S2+1] is -1, all data within the range from the character specified by [S2] to the last character stored in elements starting from [S1] is stored to elements starting from [D].



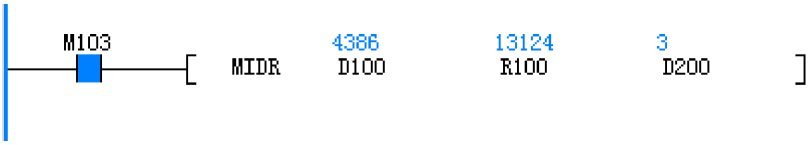
Errors

An error is returned in the following conditions:

- "00H" is not found within the corresponding element range starting from [S1].
- The value specified in [S2] is greater than the number of characters of the character string stored in elements starting from [S1].
- The number of elements starting from [D] is smaller than the number of elements required to store the extracted [S2+1] characters.
- The value specified in [S2] is negative.

- The value specified in [S2+1] is -2 or less.
- The value specified in [S2+1] exceeds the number of characters stored in elements starting from [S1].

Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x5566
D103	16-bit INT	Hex	0x7788
D104	16-bit INT	Hex	0x0
D200	16-bit INT	Dec	3
D201	16-bit INT	Dec	4
R100	16-bit INT	Hex	0x3344
R101	16-bit INT	Hex	0x5566
R102	16-bit INT	Hex	0x0

5.9.14 $MOV

The $MOV instruction transfers character string data.

16-bit instruction	$MOV: Continuous execution/$MOVP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S	Source address	Character string (a maximum of 32 characters) directly specified in the transfer source, or start number of elements that store the character string	-	INT, array*indeterminate
D	Destination address	Start number of soft elements that store a string transferred to a destination	-	INT, array*indeterminate

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function

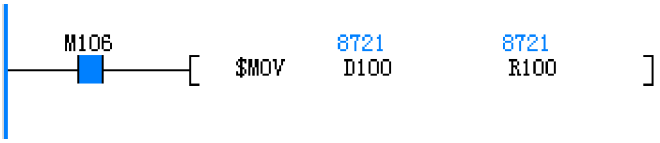
The string from start address [S] is copied to soft elements from start address [D]. All characters of the string that starts from [S] and ends with the first 00H are transferred at a time, together with the terminator 00H or 0000H.

Errors

An error is returned in the following conditions:

- "00H" is not found within the corresponding element range starting from [S]. An error is reported due to out-of-range search.
- The number of elements starting from [D] is smaller than the number of elements required to store the transferred character string data.

Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x2211
D101	16-bit INT	Hex	0x4433
D102	16-bit INT	Hex	0x6655
D103	16-bit INT	Hex	0x0
R100	16-bit INT	Hex	0x2211
R101	16-bit INT	Hex	0x4433
R102	16-bit INT	Hex	0x6655
R103	16-bit INT	Hex	0x0

5.9.15 ClearString

Return value type	Clear string		
BOOL			
Operands	Description	Value range	Data type
S	Text string to clear	-	STRING

For soft elements, see the following table.

Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom string variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S	-	-	-	-	-	√	-	-	-	-	-	-	-	-

Function and Instruction Description

This instruction clears text string S. The '\0' character is stored in the entire range of *S*.

Example

```
Return_BOOL := ClearString(
                  InOut_Str);
```

> **NOTE**
>
> The return value is always TRUE.

5.9.16 CONCAT

Return value type	This instruction joins two input text strings and provides the output.		
STRING			
Operands	Description	Value range	Data type
S1	Source string 1	-	STRING
S2	Source string 2	-	STRING

See the following table for soft elements.

Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom variables	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	-	√	-	-	-	-	-	-	√	-

Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom variables	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S2	-	-	-	-	-	√	-	-	-	-	-	-	√	-

Function

This instruction joins the input text strings S1 and S2 and provides the result.

Example

```
Out_Str := CONCAT(
           In_Str1, In_Str2);
```

> **NOTE**
>
> When the combined length of strings S1 and S2 is greater than 255, only the first 255 characters are returned.

5.9.17 DELETE

Return value type	This instruction deletes a specified-length text string at a specified position from the input text string, outputting the processed text string.
STRING	

Operands	Description	Value range	Data type
S1	Source string	-	STRING
S2	Length for deletion	0 to 255	BYTE/INT/DINT
S3	Position for deletion	0 to 255	BYTE/INT/DINT

For soft elements, see the following table.

Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom string variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	-	√	-	-	-	-	-	-	√	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-

Function

This instruction deletes a specified-length text string at a specified position from the input text string, outputting the processed text string.

Example

```
Out_Str := DELETE(
        In_Str1,In_Int1,In_Int2);
```

S3 indicates the position of the character to be deleted, where the first character at the leftmost of the string is assigned position 1, and each subsequent character increments the position by 1.

5.9.18 FIND

Return value type	This instruction searches the input source text string for the position of a specified text string.		
BYTE			
Operands	**Description**	**Value range**	**Data type**
S1	Source string	-	STRING
S2	String lookup	-	STRING

For soft elements, see the following table.

Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom variables	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	-	√	-	-	-	-	-	-	√	-
S2	-	-	-	-	-	√	-	-	-	-	-	-	√	-

Function

This instruction searches the input text string S1 for the position of the first character of text string S2.

Example

```
Out_Int := FIND(
    In_Str1, In_Str2);
```

NOTE

- If the text string to search for does not exist, the return value is 0.
- When the source string contains multiple matches, the return value is the position of the leftmost occurrence.
- The return value indicates the position of the search result, where the first character at the leftmost of the string is assigned position 1, and each subsequent character increments the position by 1.

5.9.19 REPLACE

Return value type	Replace some characters in a string
STRING	

Operands	Description	Value range	Data type
S1	Source string	-	STRING
S2	Target text string	-	STRING
S3	Replacement length	0 to 255	BYTE/INT/DINT
S4	Replacement position	0 to 255	BYTE/INT/DINT

For soft elements, see the following table.

Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom string variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	-	√	-	-	-	-	-	-	√	-
S2	-	-	-	-	-	√	-	-	-	-	-	-	√	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S4	-	-	-	√	√	-	-	√	√	√	-	-	-	-

Function and Instruction Description

This instruction replaces a part of the source text string with a specified text string, obtaining a new text string as the return value.

Example

```
Out_Str1 := REPLACE(
    In_Str1, In_Str2, In_Int1, In_Int2);
```

NOTE

- If the actual length of the character string after replacement is grerater than 255, only the first 255 characters are returned.
- S4 indicates the position of characters replaced. The first character at the leftmost of the string is assigned position 1, and each subsequent character increments the position by 1.

5.9.20 INSERT

Return value type	Insert string		
STRING			
Operands	**Description**	**Value range**	**Data type**
S1	Source string	-	STRING
S2	String insertion	-	STRING
S3	Insertion position	0 to 255	BYTE/INT/DINT

See the following table for soft elements.

Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom string variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	-	√	-	-	-	-	-	-	√	-
S2	-	-	-	-	-	√	-	-	-	-	-	-	√	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-

Function

This instruction inserts a text string within the input source text string on the left, and outputs the resulted text string.

Example

```
Out_Str1 := INSERT(
        In_Str1, In_Str2, In_Int1);
```

NOTE

- S3 indicates the insertion position. The leftmost character in the string starts at position 1, and each subsequent character increments the position by 1.
- If the combined actual length of strings S1 and S2 exceeds 255, only the first 255 characters will be returned.

5.10 Array Instructions

5.10.1 List of Instructions

The following table lists the array instructions.

Category	Instruction	Function	Language Support
Array instructions	AryMax and AryMin	Search maximum or minimum value in array variable	LiteST
	ArySearch	Search value in one-dimensional array	LiteST
	SizeOfAry	Get number of array elements	LiteST

Category	Instruction	Function	Language Support
	AryMove	Array move	LiteST
	AryExchange	Array data exchange	LiteST
	ClEAR	Initialize variable	LiteST
	AryAddV and ArySubV	Array value addition and subtraction	LiteST
	ArySHL and ArySHR	Shift N array elements left or right	LiteST
	AryAnd	Array logic AND	LiteST
	AryOr	Array logic OR	LiteST
	AryXor	Array logic XOR	LiteST
	AryXorN	Array logic XNOR	LiteST
	AryMean	Mean of array elements	LiteST
	ArySD	Standard deviation of array elements	LiteST

5.10.2 AryMax

Return Value Type	Array maximum		
BYTE, INT, DINT, REAL, and STRING			
Operand	Description	Range	Data Type
S1	Start member of the object array to be searched	-	BYTE, INT, DINT, REAL, and STRING
S2	Length of the object array to be searched	-	BYTE, INT, and DINT
S3	Position of the search result	-	INT and DINT
S4	Number of entries in the search result	-	INT and DINT

The following table lists soft elements in the instruction.

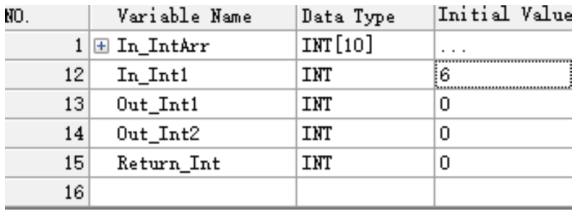
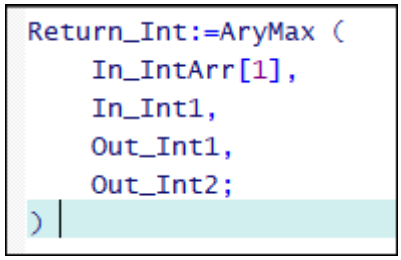
Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	√	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S3	-	-	-	√	√	-	-	-	-	-	-	-	-	-
S4	-	-	-	√	√	-	-	-	-	-	-	-	-	-

Function and Instruction Description

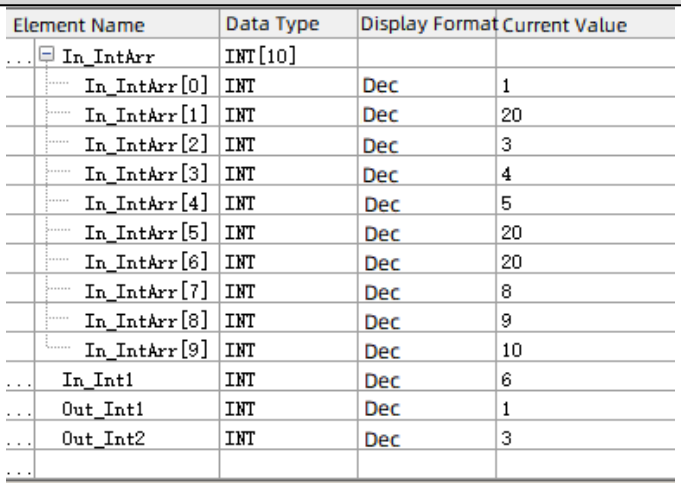
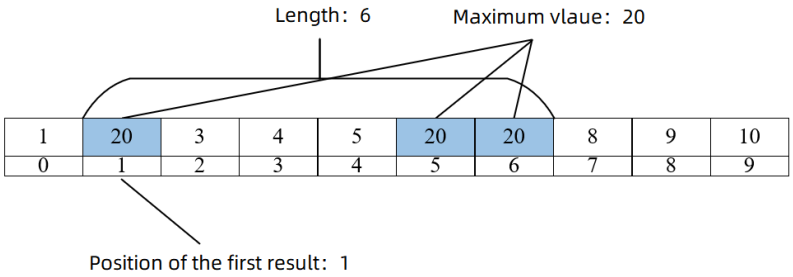
Searches for the maximum value of an element in a one-dimensional array.

- S1 is a member in an array, such as a[2].
- A compilation error occurs if S1 is not an array member.
- S2 specifies the length of data to be searched, and the search range includes S1 and elements of the number specified by S2. For example, if S1 is a[2] and S2 is 3, entries to be searched are a[2], a[3], and a[4].
- S3 is an InOut parameter, and the first matching subscript position is assigned to S3.
- S4 is an Out parameter, and the number of matched entries is assigned to S4.
- The value of the matched array member is returned, and its data type is the same as that of S1.

Instruction Example

Item	ST
Defined variable	
Program	
Operation result	


```
Return_Int:=AryMax (
    In_IntArr[1],
    In_Int1,
    Out_Int1,
    Out_Int2;
)
```

Item	ST
	
Principle	

NOTE

If the data type of S1 is not supported, an error occurs during compilation.

5.10.3 AryMin

Return Value Type	Array minimum		
BYTE, INT, DINT, REAL, and STRING			
Operand	Description	Range	Data Type
S1	Start member of the object array to be searched	-	BYTE, INT, DINT, REAL, and STRING
S2	Length of the object array to be searched	-	BYTE, INT, and DINT
S3	Position of the search result	-	INT and DINT

S4	Number of entries in the search result	-	INT and DINT

The following table lists soft elements in the instruction.

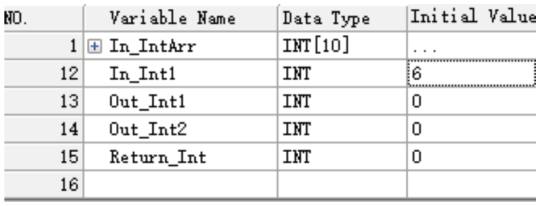
Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	√	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S3	-	-	-	√	√	-	-	-	-	-	-	-	-	-
S4	-	-	-	√	√	-	-	-	-	-	-	-	-	-

Function and Instruction Description

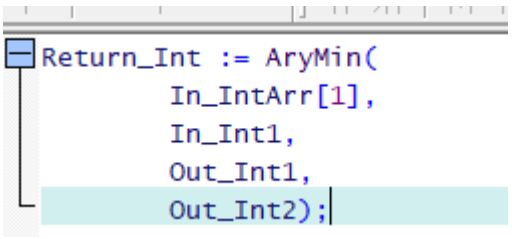
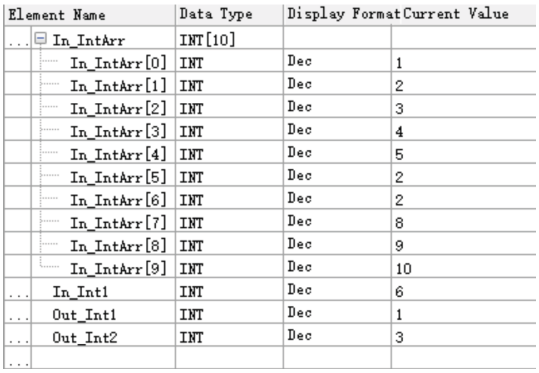
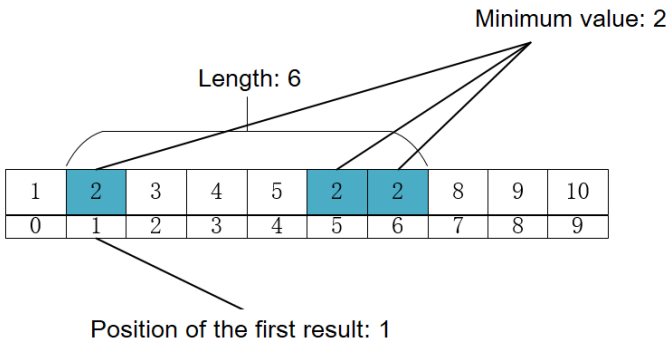
AryMin: Searches for the minimum value of an element in a one-dimensional array.

- S1 is a member in an array, such as a[2].
- A compilation error occurs if S1 is not an array member.
- S2 specifies the length of data to be searched, and the search range includes S1 and elements of the number specified by S2. For example, if S1 is a[2] and S2 is 3, entries to be searched are a[2], a[3], and a[4].
- S3 is an InOut parameter, and the first matching subscript position is assigned to S3.
- S4 is an Out parameter, and the number of matched entries is assigned to S4.
- The value of the matched array member is returned, and its data type is the same as that of S1.

Instruction Example

Item	ST
Defined variable	

NO.	Variable Name	Data Type	Initial Value
1	In_IntArr	INT[10]	...
12	In_Int1	INT	6
13	Out_Int1	INT	0
14	Out_Int2	INT	0
15	Return_Int	INT	0
16			

Item	ST
Program	
Operation result	
Principle	

Element Name	Data Type	Display Format	Current Value
In_IntArr	INT[10]		
In_IntArr[0]	INT	Dec	1
In_IntArr[1]	INT	Dec	2
In_IntArr[2]	INT	Dec	3
In_IntArr[3]	INT	Dec	4
In_IntArr[4]	INT	Dec	5
In_IntArr[5]	INT	Dec	2
In_IntArr[6]	INT	Dec	2
In_IntArr[7]	INT	Dec	8
In_IntArr[8]	INT	Dec	9
In_IntArr[9]	INT	Dec	10
In_Int1	INT	Dec	6
Out_Int1	INT	Dec	1
Out_Int2	INT	Dec	3

NOTE

If the data type of S1 is not supported, an error occurs during compilation.

5.10.4 ArySearch

Return value type	Array search		
BOOL			
Operands	Description	Value range	Data type
S1	Start member of the object array to search	-	BOOL/BYTE/INT/DINT/ REAL/STRING
S2		-	BYTE/INT/DINT

	Length of object array to search		
S3	Search keyword		BOOL/BYTE/INT/DINT/ REAL/STRING
S4	Location of search result	-	INT/DINT
S5	Number of search results	-	INT/DINT

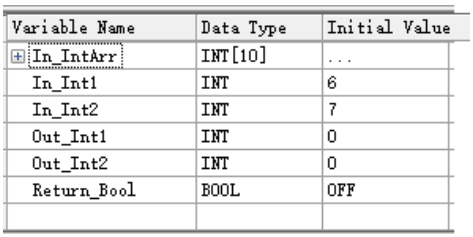
For soft elements, see the following table.

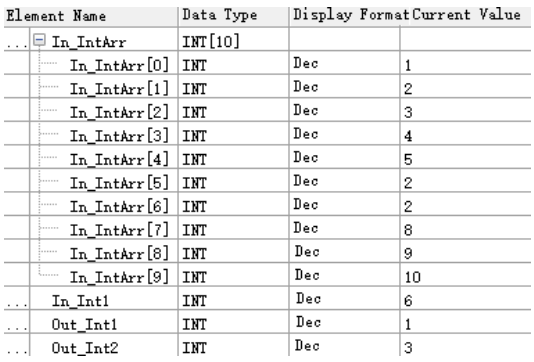
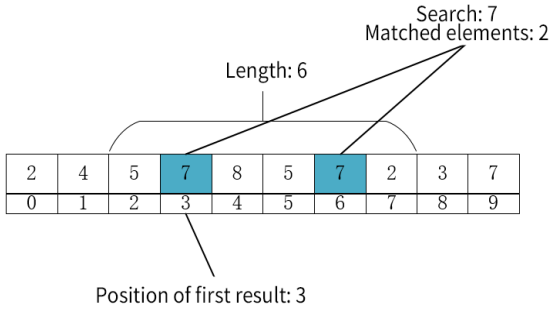
Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom string variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	√	-	√	√	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	-	-	-	-	-	-	-
S3	√	√	√	√	√	√	-	-	-	-	-	-	-	-
S4	-	-	-	√	√	-	-	-	-	-	-	-	-	-
S5	-	-	-	√	√	-	-	-	-	-	-	-	-	-

Function

- S1 include array members, such as a[2].
- A compiling error is reported when S1 is not an array member.
- S2 indicate the search length. The search range covers S1 and its subsequent S2-1 members. For example, if S1 is a[2], S2 is 3, the search range include a[2], a[3], and a[4].
- S4 is the InOut parameter. The index position of the first matching element searched is assigned to S3.
- S5 is the Out parameter. The total number of matching members searched is assigned to S4.

Example

Items	ST
Define variables	

Items	ST
Program	`Return_Bool := ArySearch(In_IntArr[2], In_Int1, In_Int2, Out_Int1, Out_Int2);`
Running result	
Work principle	

NOTE

- An error is reported when the data types of S3 and S1 are inconsistent.
- S3 does not support constants.
- When the search yields matching results, the return value is TRUE. Otherwise, the return value is FALSE.
- If the search range is beyond the array range, the return value is FALSE.

5.10.5 SizeOfAry

Return Value Type	Get number of array elements		
BYTE, INT, and DINT			
Operand	**Description**	**Range**	**Data Type**
S1	Object array	-	ARRAY

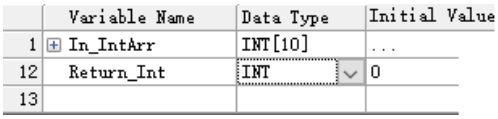
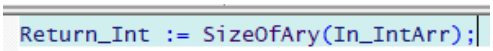
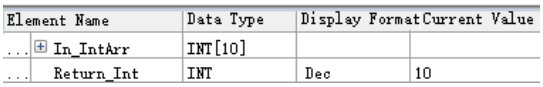
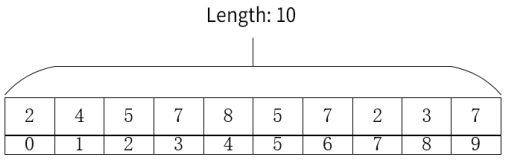
The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	√	-	√	√	-	-	-	-	-	-	-	Array

Function and Instruction Description

- The SizeOfAry instruction gets the number of elements in an array.
- S1 is a member in an array, such as a[2]. S1 can also be an array variable, such as a.
- The type of the return value is determined by the length. The type is BYTE if the length is less than 255. The type is INT if the length is in the range of 256 to 32767. The type is DINT if the length is 32767.

Instruction Example

Item	ST
Defined variable	
Program	
Operation result	
Principle	

5.10.6 AryMove

Return Value Type	Move multiple array elements		
BOOL			
Operand	Description	Range	Data Type

S1	Source object array to be moved	-	BOOL, BYTE, INT, DINT, REAL, and STRING
S2	Move result array	-	BOOL, BYTE, INT, DINT, REAL, and STRING
S3	Number of members to be moved	-	BYTE, INT, and DINT

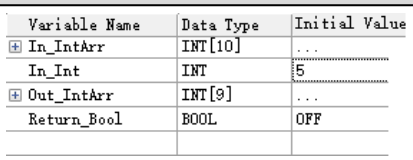
The following table lists soft elements in the instruction.

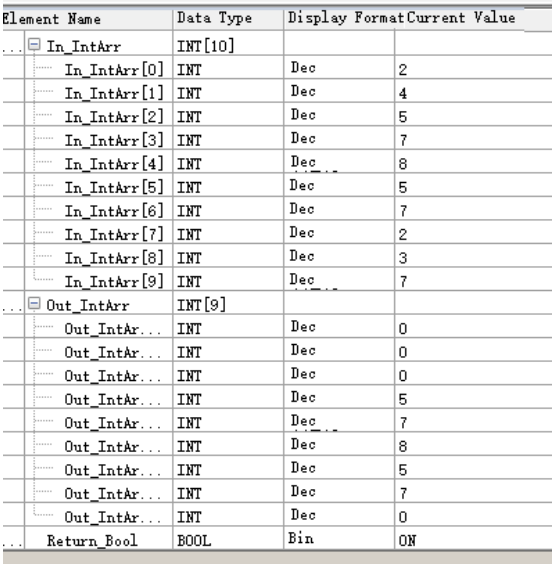
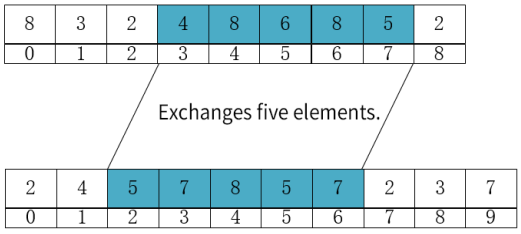
Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	√	-	√	√	-	-	-	-	-	-	-	-
S2	-	-	√	-	√	√	-	-	-	-	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-

Function and Instruction Description

- S1 is a member in an array, such as a[2].
- S2 is a member in an array, such as b[2].
- The AryMove instruction moves array elements of the number specified by S3 starting from S1 to S2.

Instruction Example

Item	ST
Defined variable	
Program	`Return_Bool := AryMove(` `In_IntArr[2],Out_IntArr[3],In_Int);`
Operation result	

Item	ST
	
Principle	

NOTE

- A compilation error is reported if the S2 type is different from the S1 type.
- When the S3 value is 0, the function returns TRUE and S2 remains unchanged.
- If the data type of S1 and S2 elements is STRING and the string length of S1 is greater than that of S2, the function returns FLASE and S2 remains unchanged.
- If the S3 value is beyond the array area of S1 or S2, the function returns FALSE and S2 remains unchanged.

5.10.7 AryExchange

Return Value Type	Exchange elements of two arrays
BOOL	

Operand	Description	Range	Data Type
S1	Source object arrays to be exchanged	-	BOOL, BYTE, INT, DINT, REAL, and STRING
S2	Exchange result array	-	BOOL, BYTE, INT, DINT, REAL, and STRING

S3	Number of exchanged members	-	BYTE, INT, and DINT

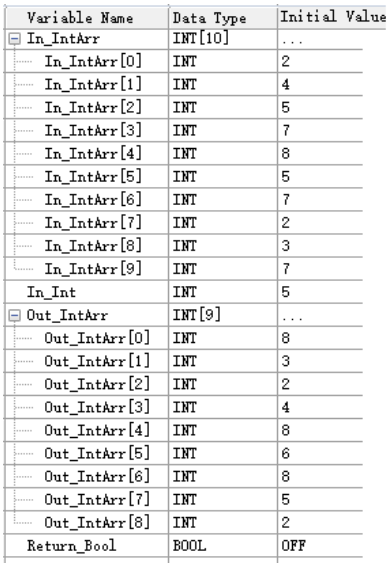
The following table lists soft elements in the instruction.

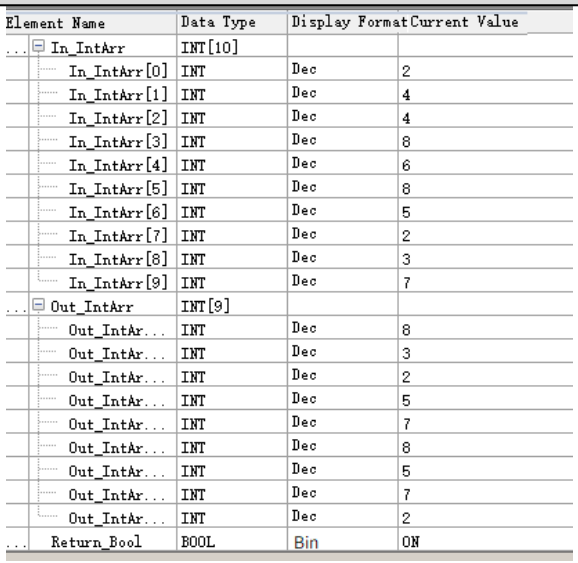
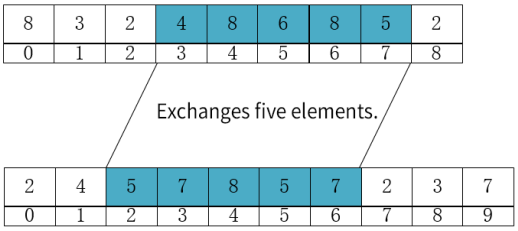
Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	√	-	√	√	-	-	-	-	-	-	-	-
S2	-	-	√	-	√	√	-	-	-	-	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-

Function and Instruction Description

- S1 is a member in an array, such as a[2].
- S2 is a member in an array, such as b[2].
- The AryExchange instruction exchanges the values of array members of the number specified by S3 starting from S1 with the same number of array members starting from S2.

Instruction Example

Item	ST
Defined variable	
Program	`Return_Bool := AryExchange(` `In_IntArr[2],Out_IntArr[3],In_Int);`
Operation result	

Item	ST
	
Principle	

NOTE

- A compilation error is reported if the S2 type is different from the S1 type.
- When the S3 value is 0, the function returns TRUE, and S1 and S2 remain unchanged.
- If the data type of S1 and S2 elements is STRING and the string length of S1 is not equal to that of S2, the function returns FLASE, and S1 and S2 remain unchanged.
- If the S3 value is beyond the array area of S1 or S2, the function returns FALSE, and S1 and S2 remain unchanged.

5.10.8 CLEAR

Return Value Type	Initialize variable		
BOOL			
Operand	**Description**	**Range**	**Data Type**
S1	Data to initialize	-	ARRAY, BOOL, BYTE, INT, DINT, REAL, and STRING

The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	√	√	√	√	√	√	-	-	-	-	-	-	-	Array

Function and Instruction Description

- S1 must be a variable.
- S1 can be an array or a member in an array of the basic types.
- The Clear instruction initializes the value of the data to initialize specified by S1 to the default initial value of each data type.

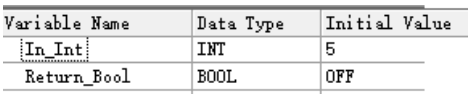
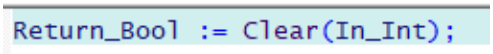
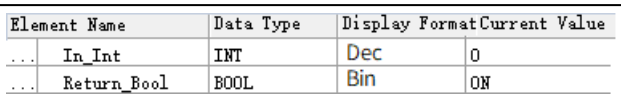
The following table describes default values for various data types.

Data Type	Initial Value
BOOL	FALSE
BYTE, INT, and DINT	0
REAL	0.0
STRING	''

When S1 is an array or a variable in an element of the array, S1 is processed according to the following table.

S1 Type	Operation
Array	Initialize all members in the array.
Array member	Initialize the member.

Instruction Example

Item	ST
Defined variable	
Program	
Operation result	

5.10.9 AryAddV

Return Value Type	Add same numerical value to one-dimensional array		
BOOL			
Operand	Description	Range	Data Type
S1	Source object array	-	BYTE, INT, DINT, and REAL
S2	Value to be added or subtracted	-	Same as that of S1
S3	Number of elements	-	BYTE, INT, and DINT
S4	Result array	-	Same as that of S1

The following table lists soft elements in the instruction.

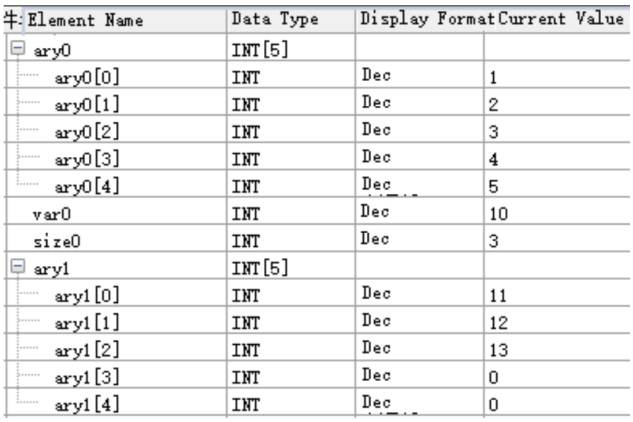
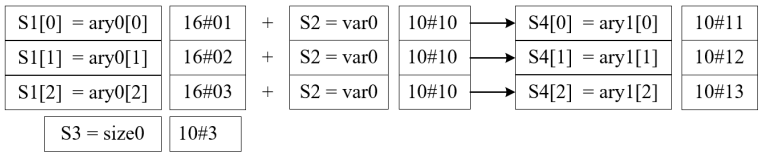
Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	-	-	-	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S4	-	-	-	-	√	-	-	-	-	-	-	-	-	-

Function and Instruction Description

The AryAddV instruction adds elements of the number specified by S3 starting from S1[0] to S2 and stores the result to the elements of the number specified by S3 starting from S4[0].

Instruction Example

Item	ST
Defined variable	NO. | Variable Name | Data Type | Initial Value 1 ⊞ ary0 | INT[5] | ... 7 var0 | INT | 0 8 size0 | INT | 0 9 ⊞ ary1 | INT[5] | ... 15 Return_Bool | BOOL | OFF
Program	

Item	ST
	`Return_Bool := AryAddV(ary0[0], var0, size0, ary1[0]);`
Operation result	
Principle	

Element Name	Data Type	Display Format	Current Value
ary0	INT[5]		
ary0[0]	INT	Dec	1
ary0[1]	INT	Dec	2
ary0[2]	INT	Dec	3
ary0[3]	INT	Dec	4
ary0[4]	INT	Dec	5
var0	INT	Dec	10
size0	INT	Dec	3
ary1	INT[5]		
ary1[0]	INT	Dec	11
ary1[1]	INT	Dec	12
ary1[2]	INT	Dec	13
ary1[3]	INT	Dec	0
ary1[4]	INT	Dec	0

NOTE

- If the S3 value is beyond the array area of S1[] or S4[], the function returns FALSE and AryOut[] remains unchanged.
- A compilation error occurs if the data types of S1[], S2, and S4[] are different.
- If the addition result is beyond the valid range of S4[], the elements in S4[] are invalid.
- When the S3 value is 0, S4[] remains unchanged and the function returns TRUE.
- When the S3 value is negative, S4[] remains unchanged and the function returns TRUE.

5.10.10 ArySubV

Return Value Type	Subtract same numerical value from one-dimensional array		
BOOL			
Operand	Description	Range	Data Type
S1	Source object array	-	BYTE, INT, DINT, and REAL
S2	Value to be added or subtracted	-	Same as that of S1
S3	Number of elements	-	BYTE, INT, and DINT
S4	Result array	-	Same as that of S1

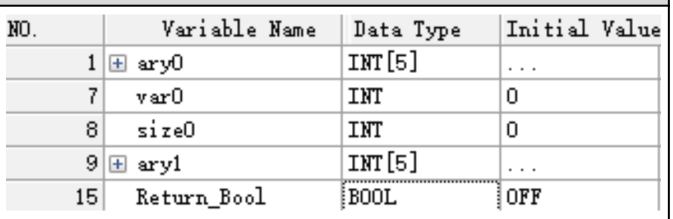
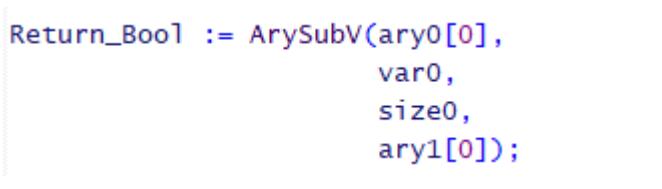
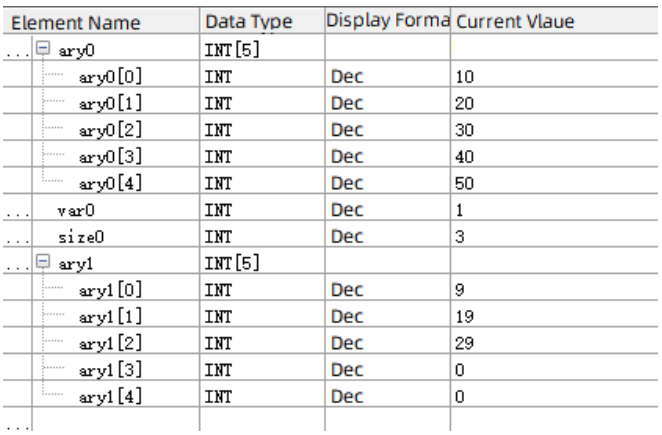
The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	-	-	-	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S4	-	-	-	-	√	-	-	-	-	-	-	-	-	-

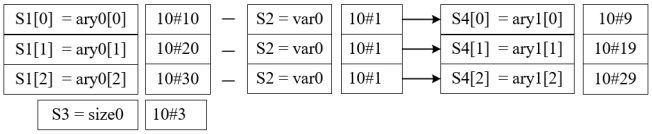
Function and Instruction Description

The ArySubV instruction subtracts S2 from elements of the number specified by S3 starting from S1[0] and stores the result to the elements of the number specified by S3 starting from S4[0].

Instruction Example

Item	ST
Defined variable	
Program	
Operation result	
Principle	


```
Return_Bool := ArySubV(ary0[0],
                       var0,
                       size0,
                       ary1[0]);
```

Item	ST
	

NOTE

- If the S3 value is beyond the array area of S1[] or S4[], the function returns FALSE and AryOut[] remains unchanged.
- A compilation error occurs if the data types of S1[], S2, and S4[] are different.
- If the subtraction result is beyond the valid range of S4[], the elements in S4[] are invalid.
- When the S3 value is 0, S4[] remains unchanged and the function returns TRUE.
- When the S3 value is negative, S4[] remains unchanged and the function returns TRUE.

5.10.11 ArySHL

Return Value Type	Move one-dimensional array left for N elements		
BOOL			
Operand	Description	Range	Data Type
S1	Shifted object array	-	BOOL, BYTE, INT, DINT, REAL, STRING, and STRUCT
S2	Number of elements	-	BYTE, INT, and DINT
S3	Number of bits to shift	-	BYTE, INT, and DINT

The following table lists soft elements in the instruction.

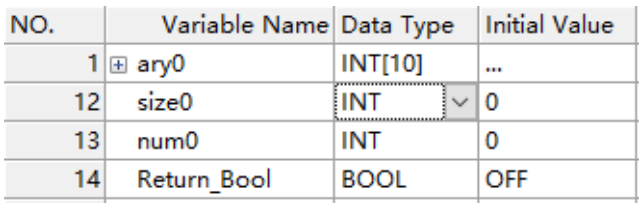
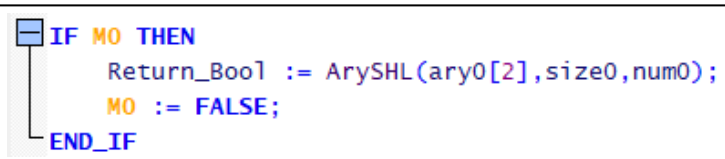
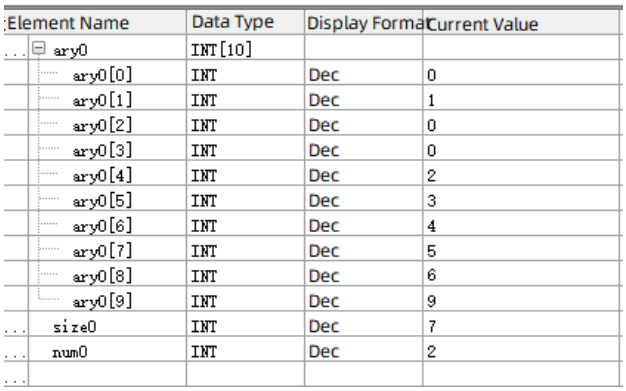
Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-

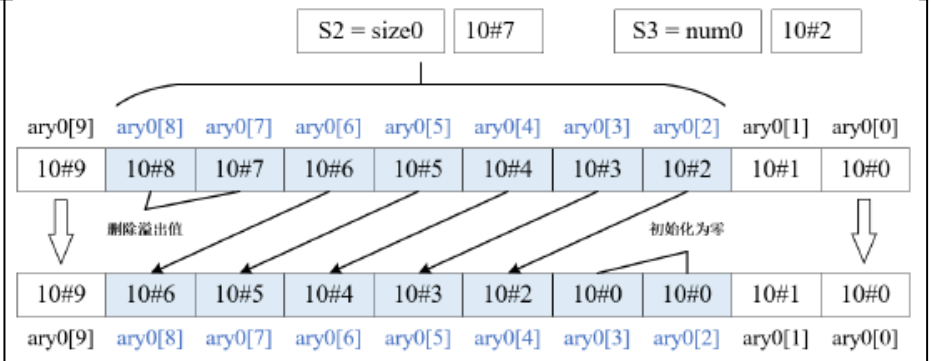
Function and Instruction Description

- ArySHL: Shifts multiple array elements to the left (toward the higher elements) for N elements.
- The lower or higher elements of the number specified by S2 in the object array S1[] to be shifted are shift to the left for the number of elements specified by S3.
- Elements that are shifted out of the array are discarded.
- After shift, null elements in the object array S1[] are initialized to the default values.

Instruction Example

Item	ST
Defined variable	
Program	
Operation result	
Principle	


```
IF M0 THEN
    Return_Bool := ArySHL(ary0[2],size0,num0);
    M0 := FALSE;
END_IF
```

Item	ST
	

NOTE

- When the S2 value is greater than the S1[] array area, S1[ ] remains unchanged and the function returns FALSE.
- When the S3 value or S2 value is 0, shift operation is not performed, S1[] remains unchanged, and the function returns TRUE.
- When the S3 value is greater than the S2 value, all the values in S1[S2 – 1] starting from S1[0] are initialized to the default values.
- When the S2 value or S3 value is negative, S1[] remains unchanged and the function returns FALSE.

5.10.12 ArySHR

Return Value Type	Move one-dimensional array right for N elements		
BOOL			
Operand	Description	Range	Data Type
S1	Shifted object array	-	BOOL, BYTE, INT, DINT, REAL, STRING, and STRUCT
S2	Number of elements	-	BYTE, INT, and DINT
S3	Number of bits to shift	-	BYTE, INT, and DINT

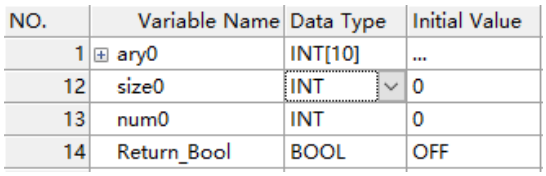
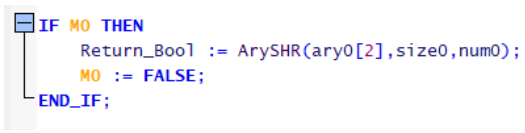
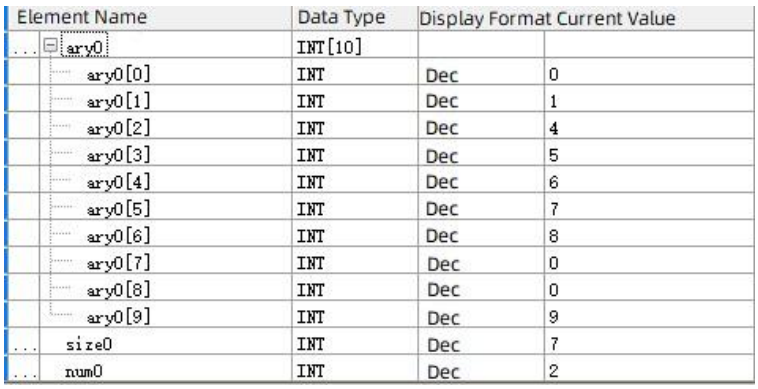
The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-

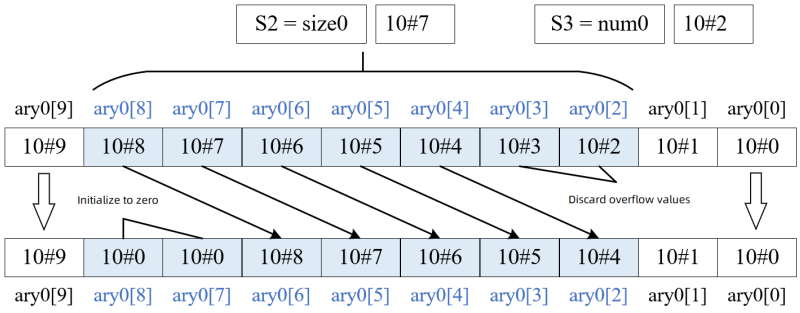
Function and Instruction Description

- ArySHR: Shifts multiple array elements to the right (toward the lower elements) for N elements.
- The lower or higher elements of the number specified by S2 in the object array S1[ ] to be shifted are shift to the left or right for the number of elements specified by S3.
- Elements that are shifted out of the array are discarded.
- After shift, null elements in the object array S1[ ] are initialized to the default values.

Instruction Example

Item	ST
Defined variable	
Program	
Operation result	
Principle	


```
IF M0 THEN
    Return_Bool := ArySHR(ary0[2],size0,num0);
    M0 := FALSE;
END_IF;
```

Item	ST
	

NOTE

- When the S2 value is greater than the S1[ ] array area, S1[ ] remains unchanged and the function returns FALSE.
- When the S3 value or S2 value is 0, shift operation is not performed, S1[ ] remains unchanged, and the function returns TRUE.
- When the S3 value is greater than the S2 value, all the values in S1[S2 – 1] starting from S1[0] are initialized to the default values.
- When the S2 value or S3 value is negative, S1[ ] remains unchanged and the function returns FALSE.

5.10.13 AryAnd

Return Value Type	Logic AND operation on each bit of each element between arrays		
BOOL			
Operand	Description	Range	Data Type
S1	Array 1 to process	-	BOOL, BYTE, INT, and DINT
S2	Array 2 to process	-	Same as that of S1
S3	Number of elements	-	BYTE, INT, and DINT
S4	Processing result array	-	Same as that of S1

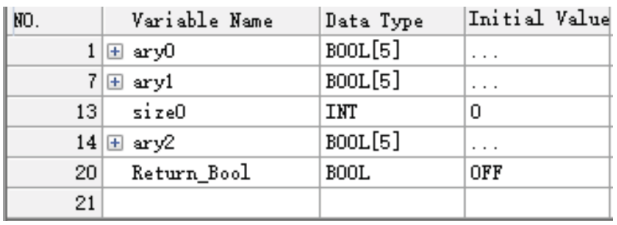
The following table lists soft elements in the instruction.

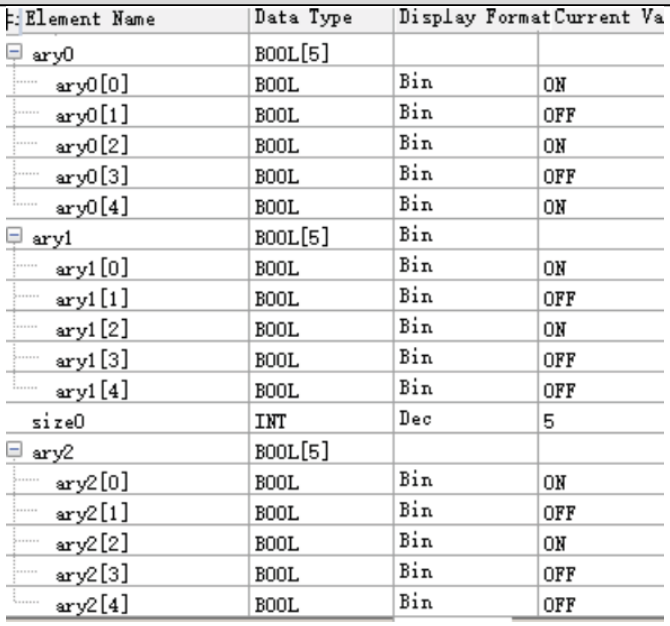
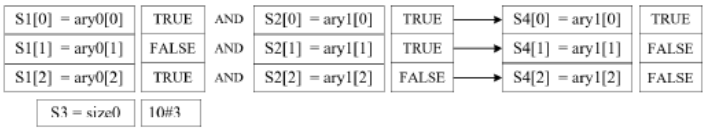
Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S4	-	-	-	-	√	-	-	-	-	-	-	-	-	-

Function and Instruction Description

- AryAnd: When both operands in the logic AND operation is TRUE, the result is TRUE. Otherwise, the result is FALSE.
- Logic operations are performed on the bits of the first elements of the number specified by S3 in the arrays S1 and S2, and the operation results are stored in the corresponding elements in S4.

Instruction Example

Item	ST
Defined variable	
Program	`Return_Bool := AryAnd(ary0[0], ary1[0], size0, ary2[0]);`
Operation result	

Item	ST
	
Principle	

Element Name	Data Type	Display Format	Current Va
ary0	BOOL[5]		
ary0[0]	BOOL	Bin	ON
ary0[1]	BOOL	Bin	OFF
ary0[2]	BOOL	Bin	ON
ary0[3]	BOOL	Bin	OFF
ary0[4]	BOOL	Bin	ON
ary1	BOOL[5]	Bin	
ary1[0]	BOOL	Bin	ON
ary1[1]	BOOL	Bin	OFF
ary1[2]	BOOL	Bin	ON
ary1[3]	BOOL	Bin	OFF
ary1[4]	BOOL	Bin	OFF
size0	INT	Dec	5
ary2	BOOL[5]		
ary2[0]	BOOL	Bin	ON
ary2[1]	BOOL	Bin	OFF
ary2[2]	BOOL	Bin	ON
ary2[3]	BOOL	Bin	OFF
ary2[4]	BOOL	Bin	OFF

NOTE

- A compilation error occurs if the data types of S1[], S2[], and S4[] are different.
- If the S3 value is beyond any array area of S1[], S2[], and S4[], the function returns FALSE and AryOut remains unchanged.
- When the S3 value is 0, the function returns TRUE and S4[] remains unchanged.
- When the S3 value is negative, the function returns FALSE and S4[] remains unchanged.

5.10.14 AryOr

Return Value Type	Logic operation on each bit of each element between arrays		
BOOL			
Operand	Description	Range	Data Type
S1	Array 1 to process	-	BOOL, BYTE, INT, and DINT
S2	Array 2 to process	-	Same as that of S1
S3	Number of elements	-	BYTE, INT, and DINT
S4	Processing result array	-	Same as that of S1

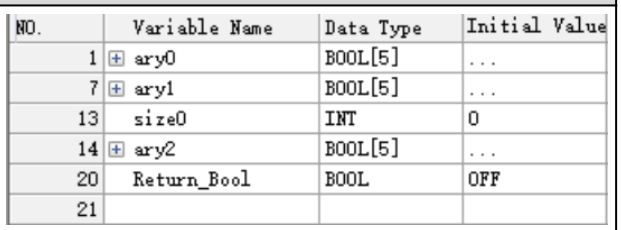
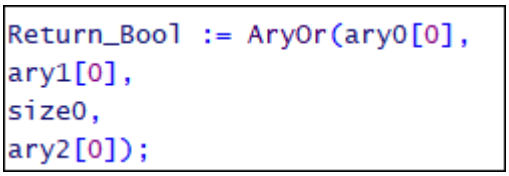
The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S4	-	-	-	-	√	-	-	-	-	-	-	-	-	-

Function and Instruction Description

- AryOr: When both operands in the logic OR operation is FALSE, the result is FALSE. Otherwise, the result is TRUE.
- Logic operations are performed on the bits of the first elements of the number specified by S3 in the arrays S1 and S2, and the operation results are stored in the corresponding elements in S4.

Instruction Example

Item	ST
Defined variable	
Program	
Operation result	


```
Return_Bool := AryOr(ary0[0],
ary1[0],
size0,
ary2[0]);
```

Item	ST
	Element Name / Data Type / Display Format / Current Value ary0 BOOL[5] ary0[0] BOOL Bin ON ary0[1] BOOL Bin OFF ary0[2] BOOL Bin ON ary0[3] BOOL Bin OFF ary0[4] BOOL Bin ON ary1 BOOL[5] ary1[0] BOOL Bin ON ary1[1] BOOL Bin OFF ary1[2] BOOL Bin ON ary1[3] BOOL Bin OFF ary1[4] BOOL Bin OFF size0 INT Bin 5 ary2 BOOL[5] ary2[0] BOOL Bin ON ary2[1] BOOL Bin OFF ary2[2] BOOL Bin ON ary2[3] BOOL Bin OFF ary2[4] BOOL Bin ON
Principle	S1[0] = ary0[0] TRUE AND S2[0] = ary1[0] TRUE → S4[0] = ary1[0] TRUE S1[1] = ary0[1] FALSE AND S2[1] = ary1[1] TRUE → S4[1] = ary1[1] TRUE S1[2] = ary0[2] TRUE AND S2[2] = ary1[2] FALSE → S4[2] = ary1[2] TRUE S3 = size0 10#3

NOTE

- A compilation error occurs if the data types of S1[], S2[], and S4[] are different.
- If the S3 value is beyond any array area of S1[], S2[], and S4[], the function returns FALSE and AryOut remains unchanged.
- When the S3 value is 0, the function returns TRUE and S4[] remains unchanged.
- When the S3 value is negative, the function returns FALSE and S4[] remains unchanged.

5.10.15 AryXor

Return Value Type	Logic XOR operation on each bit of each element between arrays		
BOOL			
Operand	Description	Range	Data Type
S1	Array 1 to process	-	BOOL, BYTE, INT, and DINT
S2	Array 2 to process	-	Same as that of S1
S3	Number of elements	-	BYTE, INT, and DINT
S4	Processing result array	-	Same as that of S1

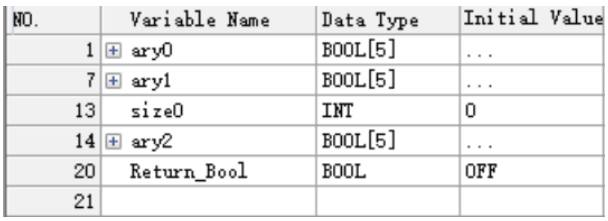
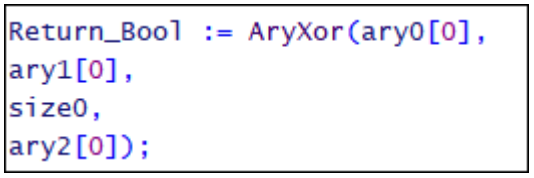
The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S4	-	-	-	-	√	-	-	-	-	-	-	-	-	-

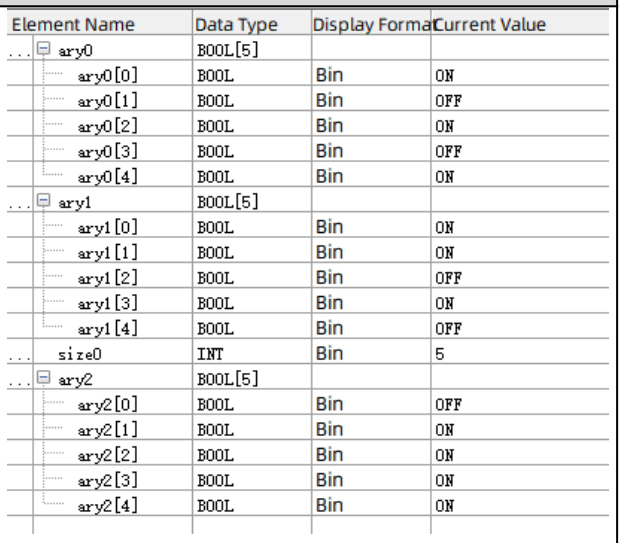
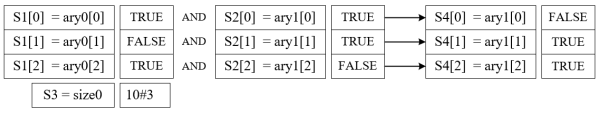
Function and Instruction Description

- AryXor: When the values of two operands in the exclusive OR operation are different, the result is TRUE. Otherwise, the result is FALSE.
- Logic operations are performed on the bits of the first elements of the number specified by S3 in the arrays S1 and S2, and the operation results are stored in the corresponding elements in S4.

Instruction Example

Item	ST
Defined variable	
Program	
Operation result	


```
Return_Bool := AryXor(ary0[0],
ary1[0],
size0,
ary2[0]);
```

Item	ST
	
Principle	

NOTE

- A compilation error occurs if the data types of S1[], S2[], and S4[] are different.
- If the S3 value is beyond any array area of S1[], S2[], and S4[], the function returns FALSE and AryOut remains unchanged.
- When the S3 value is 0, the function returns TRUE and S4[] remains unchanged.
- When the S3 value is negative, the function returns FALSE and S4[] remains unchanged.

5.10.16 AryXorN

Return Value Type	Logic XNOR operation on each bit of each element between arrays		
BOOL			
Operand	Description	Range	Data Type
S1	Array 1 to process	-	BOOL, BYTE, INT, and DINT
S2	Array 2 to process	-	Same as that of S1
S3	Number of elements	-	BYTE, INT, and DINT
S4	Processing result array	-	Same as that of S1

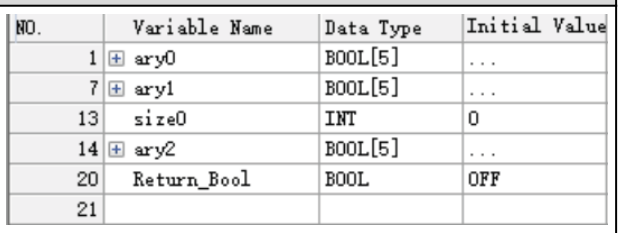
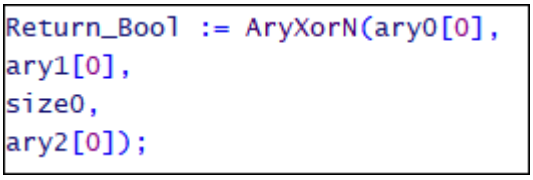
The following table lists soft elements in the instruction.

Operand	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Custom String Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S3	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S4	-	-	-	-	√	-	-	-	-	-	-	-	-	-

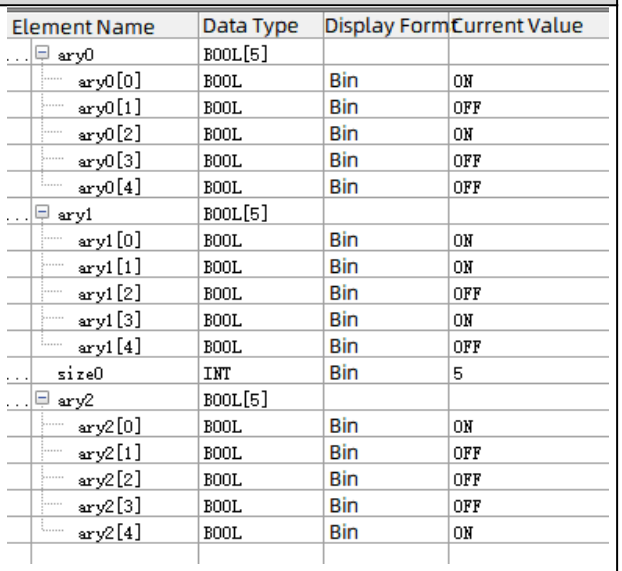
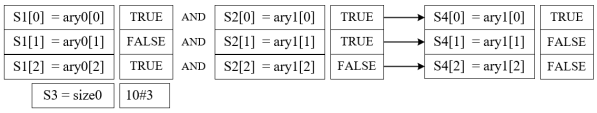
Function and Instruction Description

- AryXorN: When the values of two operands in the exclusive NOR operation are the same, the result is TRUE. Otherwise, the result is FALSE.
- Logic operations are performed on the bits of the first elements of the number specified by S3 in the arrays S1 and S2, and the operation results are stored in the corresponding elements in S4.

Instruction Example

Item	ST
Defined variable	
Program	
Operation result	


```
Return_Bool := AryXorN(ary0[0],
ary1[0],
size0,
ary2[0]);
```

Item	ST
	
Principle	

NOTE

- A compilation error occurs if the data types of S1[], S2[], and S4[] are different.
- If the S3 value is beyond any array area of S1[], S2[], and S4[], the function returns FALSE and AryOut remains unchanged.
- When the S3 value is 0, the function returns TRUE and S4[] remains unchanged.
- When the S3 value is negative, the function returns FALSE and S4[] remains unchanged.

5.10.17 ArySD

Return value type	This instruction calculates the standard deviation of elements of an array.		
BOOL			
Operands	Description	Value range	Data type
S1	Array to process	-	REAL
S2	Number of elements	-	BYTE/INT/DINT
S3	Operation result	-	REAL

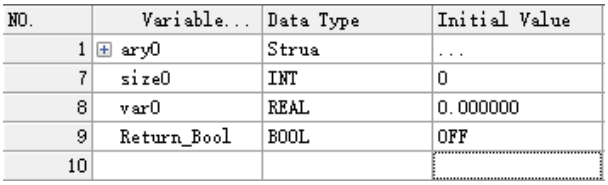
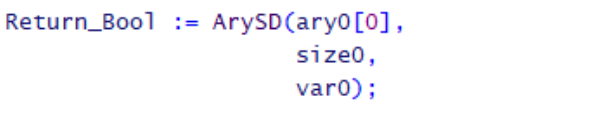
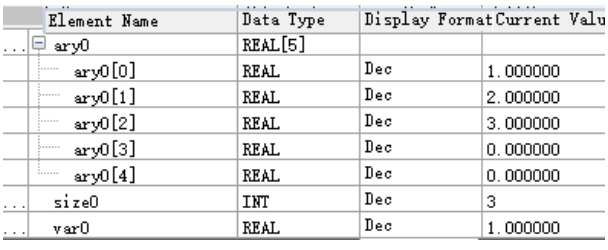
For soft elements, see the following table.

Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom string variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S3	-	-	-	-	√	-	-	-	-	-	-	-	-	-

Function

This instruction calculates the standard deviation of elements of an array (Formula: $\partial \equiv \sqrt{\frac{\sum_{i=1}^{n}\left(x_i-\bar{x}\right)^2}{n-1}}$).

Example

Items	ST
Define variables	
Program	
Running result	


```
Return_Bool := ArySD(ary0[0],
                     size0,
                     var0);
```


- When the value of S2 is 0, the value of S3 is 0.
- If the value of S2 is greater than *S1[]*, S3 does not change, and the returned value is FALSE.
- If the value of S2 is negative, S3 does not change, and the returned value is FALSE.
- If the intermediate value in the calculation process exceeds the valid range of *S1[]*, S3 does not change, and the returned value is FALSE.

5.10.18 AryMean

Return value type	This instruction calculates the average of elements of an array.
BOOL	

Operands	Description	Value range	Data type
S1	Array to process	-	BYTE/INT/DINT/REAL
S2	Number of elements	-	BYTE/INT/DINT
S3	Operation result	-	Array with the same data type as the elements of S1

For soft elements, see the following table.

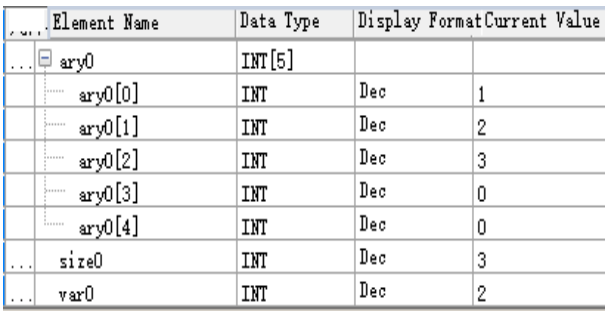
Operands	Bit			Word			Pointer	Constant						Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Custom string variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	STRING	
S1	-	-	-	-	√	-	-	-	-	-	-	-	-	-
S2	-	-	-	√	√	-	-	√	√	√	-	-	-	-
S3	-	-	-	-	√	-	-	-	-	-	-	-	-	-

Function

This instruction calculates the average of elements of an array.

Example

Items	ST
Define variables	Variable Name | Data Type | Initial Valu ary0 | INT[5] | ... size0 | INT | 0 var0 | INT | 0 Return_Bool | BOOL | OFF
Program	`Return_Bool := AryMean(ary0[0], size0, var0);`

Items	ST
Operation result	

Element Name	Data Type	Display Format	Current Value
ary0	INT[5]		
ary0[0]	INT	Dec	1
ary0[1]	INT	Dec	2
ary0[2]	INT	Dec	3
ary0[3]	INT	Dec	0
ary0[4]	INT	Dec	0
size0	INT	Dec	3
var0	INT	Dec	2

NOTE

- When S2 is 0, the value of S3 is 0 and the return value is TRUE.
- When *S1[]* or S3 is an integer, the decimal portion of the average is truncated.
- When the value of S2 exceeds the *S1[]* array, the return value is FALSE and S3 does not change.
- When the value of S2 is negative, S3 does not change, and the return value is FALSE.

5.11 Selection

5.11.1 List of Instructions

The following table lists the comparison instructions.

Category	Name	Description	Language Support
Comparison instructions	MAX	Max operation	LiteST
	MIN	Min operation	LiteST
	SEL	Select operation	LiteST

5.11.2 MAX

Return Value Type	Max operation		
BYTE, INT, DINT, and REAL			
Operand	Description	Range	Data Type
S1	Variable 1 for comparison	-	BYTE, INT, DINT, and REAL
S2		-	BYTE, INT, DINT, and REAL

	Variable 2 for comparison		

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	-	-	-	√	√	-	√	√	√	√	-	-
S2	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

The MAX instruction compares the values of two BYTE, INT, DINT, or REAL variables and returns the larger value, In this instruction:

- S1 is variable 1 for comparison.
- S2 is variable 2 for comparison.
- The return value is the larger one of two values.
- The return value type is consistent with the input type.

NOTE

- The data types of S1 and S2 must be consistent, or the type of S1 or S2 can be implicitly converted into the type of the other variable.
- When the type of S1 or S2 is implicitly converted into the type of another variable, the return value is the type after the implicit conversion.
- When the type of S1 or S2 is the result of a bit operation, the other variable must also be the result of a bit operation, or be a single variable.

Instruction Example

```
real0 := MAX(real1, real2);
```

Obtain the larger value of real1 and real2 and store the result in real0.

5.11.3 MIN

Return value type	Min operation
BYTE/INT/DINT/REAL	

Operands	Description	Value range	Data type
S1	Variable 1 for comparison	-	BYTE/INT/DINT/REAL
S2	Variable 2 for comparison	-	BYTE/INT/DINT/REAL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	BYTE	INT	DINT	REAL	BOOL	
S1	-	-	-	√	√	-	√	√	√	√	-	-
S2	-	-	-	√	√	-	√	√	√	√	-	-

Function and instruction description

This instruction is used to compare the values of two BYTE, INT, DINT, or REAL variables and return the smaller value, Specifically:

- [S1]: Variable 1 for comparison
- [S2]: Variable 2 for comparison
- The return value is the smaller one of two values.
- The return value type is consistent with the input type.

NOTE

- The data types of S1 and S2 must be consistent, or the type of one variable in S1 and S2 can be implicitly converted into the type of the other variable.
- When the type of one variable in S1 and S2 is implicitly converted into the type of another variable, the return value is the type after the implicit conversion.
- When the type of one variable in S1 or S2 is the result of a bit operation, the other variable must also be the result of a bit operation, or be a single variable.

Example

```
real0 := MIN(real1, real2);
```

Obtain the smaller value of real1 and real2 and store the result in real0.

5.11.4 SEL

Return Value Type	Binary operation		
BYTE, INT, DINT, REAL, and BOOL			
Operand	Description	Range	Data Type
S1	Conditional variable	-	BOOL
S2	Selective variable 1	-	BYTE, INT, DINT, REAL, and BOOL
S3	Selective variable 2	-	BYTE, INT, DINT, REAL, and BOOL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	√	√	√	-	-	-	-	-	-	-	√	-
S2	√	√	√	√	√	-	√	√	√	√	√	-
S3	√	√	√	√	√	-	√	√	√	√	√	-

Function and Instruction Description

The SEL instruction selects one of the two values for BYTE, INT, DINT, REAL, or BOOL variables. When S1 is TRUE, return S3. When S1 is FALSE, return S2. In this instruction:

- S1 is a conditional variable.
- S2 is selective variable 1.
- S3 is selective variable 2.
- The S2 value or S3 value is returned.
- The return value type is the same as the type of S2 or S3.

NOTE

- The data types of S2 and S3 must be consistent, or the type of S2 or S3 can be implicitly converted into the type of the other variable.
- When the type of S2 or S3 is implicitly converted into the type of another variable, the return value is the type after the implicit conversion.
- When the type of S1 or S2 is the result of a bit operation, the other variable must also be the result of a bit operation, or be a single variable.

Instruction Example

```
dint0 := SEL(bool0,dint1,dint2);
```

- When bool0 is TRUE, obtain the value of dint2 and store it in dint0.
- When bool0 is FALSE, obtain the value of dint1 and store it in dint0.

5.12 Clock Instructions

5.12.1 List of Instructions

The following table lists the clock instructions.

Category	Instruction	Function	Firmware Version	Software Version
Clock instruction	TCMP	Clock data comparison	V6.3.0.3 and later	V4.10.2.0 and later
	TZCP	Clock data zone comparison		
	TADD	Clock data addition		
	TSUB	Clock data subtraction		
	HTOS	Conversion from hour-minute-second into second (total seconds in hour-minute-second)		
	STOH	Conversion from second into hour-minute-second		
	TRD	Read clock data		
	TWR	Write clock data		
	HOUR	Hour meter		
	TZSET	Set time zone	V6.4.0.0	V4.10.2.0 and later
	TZGET	Read time zone	V6.4.0.0	V4.10.2.0 and later

Category	Instruction	Function	Firmware Version	Software Version

5.12.2 TCMP

The TCMP instruction compares the specified time (hour, minute, and second) with the time of an internal real-time clock and outputs the comparison result.

TCMP -- Compare clock data

16-bit Instruction	TCMP: Continuous execution/TCMPP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Hour	Hour of the comparison time, ranging from 0 to 23	-	INT
S2	Minute	Minute of the comparison time, ranging from 0 to 59	-	INT
S3	Second	Second of the comparison time, ranging from 0 to 59	-	INT
S	PLC time data start address	Start address of time registers that store the current time value of a real-time clock, which is usually the data read by the TRD or MOV instruction	-	INT, array*3
D	Comparison result	Start address of three consecutive variable units that store the comparison result	-	BOOL, array*3

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	-	-
S3	-	-	-	√	√	-	√	-	-
S	-	-	-	√	√	-	-	-	-
D	√[1]	√	-	-	-	-	-	-	-

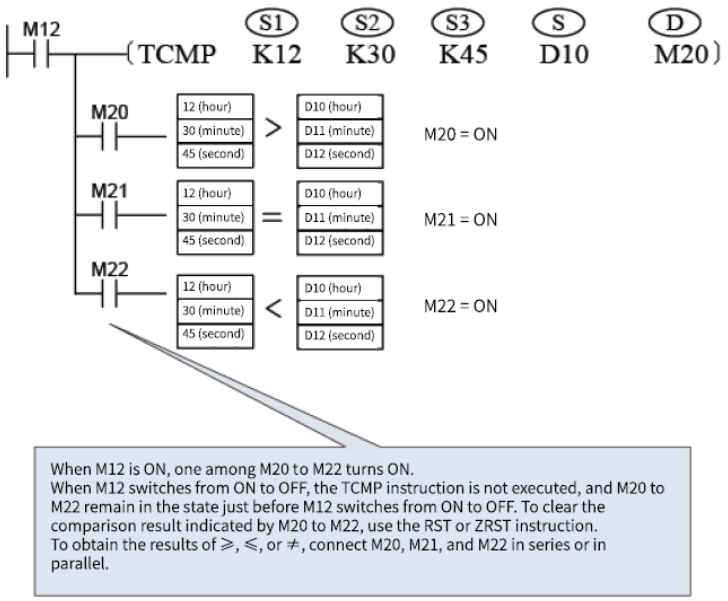
> **NOTE**
>
> [1]: The X element is not supported.

Function and Instruction Description

The TCMP instruction compares the specified time (hour, minute, and second) with the time of an internal real-time clock and outputs the comparison result. In this instruction:

- S1 is the hour of the comparison time, which ranges from 0 to 23.
- S2 is the minute of the comparison time, which ranges from 0 to 59.
- S3 is the second of the comparison time, which ranges from 0 to 59.
- S is the start address of time registers that store the current time value of a real-time clock. It is usually the data read by the TRD or MOV instruction.
- D is the start address of three consecutive variable units that store the comparison result.

Instruction Example



5.12.3 TZCP

Start address of three consecutive variable units that store the comparison result.

TZCP: Clock data range comparison

16-bit instruction	TZCP: Continuous execution TZCPP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type

S1	Lower limit	Lower limit (hour, minute, and second) of the comparison time zone, which occupies three consecutive variable units	-	INT, array*3
S2	Upper limit	Upper limit (hour, minute, and second) of the comparison time zone, which occupies three consecutive variable units	-	INT, array*3
S	PLC time data start address	Start address of time registers that store the current time value of a real-time clock, which is usually the data read by the TRD or MOV instruction	-	INT, array*3
D	Comparison result	Start address of three consecutive variable units that store the comparison result	-	BOOL, array*3

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	-	-
S	-	-	-	√	√	-	-	-	-
D	√[1]	√	-	-	-	-	-	-	-

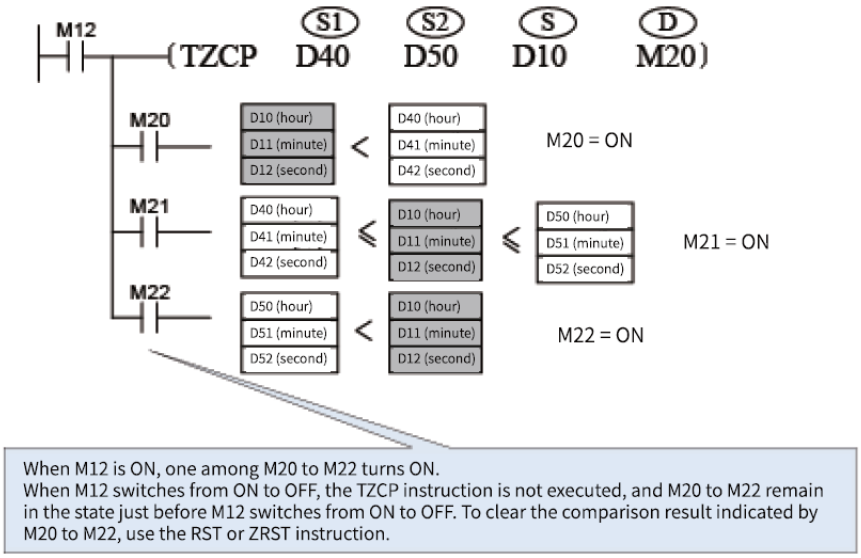
NOTE

[1] The X element is not supported.

Function and Instruction Description

The TZCP instruction compares the time data of the internal real-time clock with two specified sets of preset values, including hour (0 to 23), minute (0 to 59), and second (0 to 59), and outputs the comparison result.

Example



5.12.4 TADD

The TADD instruction adds two time values (hour:minute:second) together and stores the result in the specified variables.

TADD -- Clock data addition

16-bit Instruction	TADD: Continuous execution/TADDP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Time augend	Time augend, which occupies three consecutive variable units to store the hour, minute, and second data respectively	-	INT, array*3
S2	Time addend	Time addend, which occupies three consecutive variable units to store the hour, minute, and second data respectively	-	INT, array*3
D	Sum of two time values	Sum of two time values, which occupies three consecutive variable units to store the hour, minute, and second data respectively	-	INT, array*3

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

The TADD instruction adds two time values (hour:minute:second) together and stores the result in the specified variables. In this instruction:

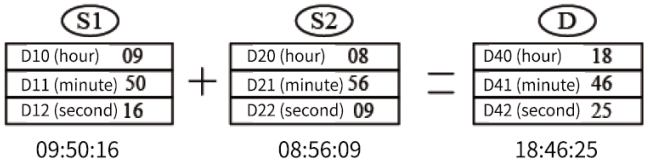
If the operation result exceeds 24 hours, the carry flag M8022 is set to 1, and the value displayed is equal to the subtraction result by 24:00:00

If the operation result is 00:00:00, the zero flag M8020 is set to 1.

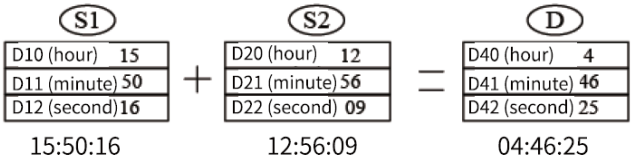
Instruction Example



The operation is performed as follows.



If the addition result is greater than 24 hours, the carry flag M8022 turns ON.



5.12.5 TSUB

The TSUB instruction subtracts one time value (hour, minute, and second) from another and stores the result in the specified variables.

TSUB: Clock data subtraction operation

16-bit instruction	TSUB: Continuous execution TSUBP: Pulse execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type

S1	Time subtrahend	Time subtrahend, which is represented by hours, minutes, and seconds respectively stored in three consecutive variable units	-	INT, array*3
S2	Time minuend	Time minuend, which is represented by hours, minutes, and seconds respectively stored in three consecutive variable units	-	INT, array*3
D	Difference between two time values	Difference between two time values, which is represented by hours, minutes, and seconds respectively stored in three consecutive variable units	-	INT, array*3

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

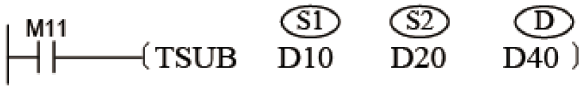
Function

The TSUB instruction subtracts one time value (hour, minute, and second) from another and stores the result in the specified variables.

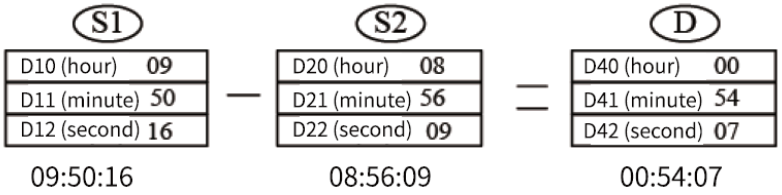
If the subtraction of two time values results is a negative value, the borrow flag M8021 is set to 1 and the actually displayed time is equal to the subtraction result plus 24:00:00.

If the subtraction result is 00:00:00, the zero flag M8020 is set to 1.

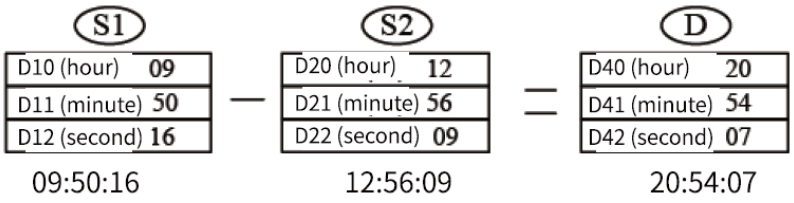
Example



Operation:



If the subtraction result is a negative value, the borrow flag M8021 is set to ON.



5.12.6 HTOS

The HTOS instruction converts the time data in the unit of hour-minute-second into data in the unit of second.

HTOS: Conversion from hour-minute-second into second

16-bit instruction	HTOS: Continuous execution HTOSP: Pulse execution			
32-bit instruction	DHTOS: Continuous execution DHTOSP: Pulse execution			
Operands	Name	Description	Value range	Data type
S	Source data	Start number of soft elements that store the time data (in the unit of hour-minute-second) to be converted	-	INT, array*3 (3 registers)
D	Result	Number of soft elements that store the time data (in the unit of second) after conversion	-	INT/DINT

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function

- 16-bit instruction
  The HTOS instruction converts the time data in the unit of hour-minute-second stored in [S, S+1, S+2] into data in the unit of second and stores the result in D.
  - Hours' range: 0 to 9
  - Minutes' range: 0 to 59
  - Seconds' range: 0 to 59
- 32-bit instruction

The HTOS instruction converts the time data in the unit of hour-minute-second stored in [S, S+1, S+2] into data in the unit of second and stores the result in [D, D+1].

- Hours' range: 0 to 32,767
- Minutes' range: 0 to 59
- Seconds' range: 0 to 59

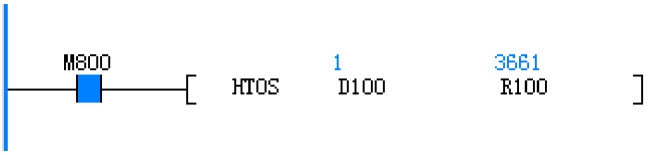
Errors

An error is returned and the instruction is not executed in the following conditions:

- The operands of the 16-bit or 32-bit instruction are out of range.
- The conversion result obtained by the 16-bit instruction is greater than 32767.
- The time data in [S, S+1, S+2] is out of range.

Example

The time data in the unit of hour-minute-second stored in D100, D101, and D102 is converted into data in the unit of second. The conversion result is stored in R100.



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Dec	1
D101	16-bit INT	Dec	1
D102	16-bit INT	Dec	1
	16-bit INT	Dec	
R100	16-bit INT	Dec	3661
	16-bit INT	Dec	

5.12.7 STOH

The STOH instruction converts the time data in the unit of second into data in the unit of hour-minute-second.

STOH -- Conversion from second into hour-minute-second

16-bit Instruction	STOH: Continuous execution/STOHP: Pulse execution			
32-bit Instruction	DSTOH: Continuous execution/DSTOHP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Number of the element that stores time data (in the unit of second) to be converted	-	INT and DINT
D	Result	Start number of elements that store the time data (in the unit of hour-minute-second) after conversion	-	INT Array*3 (fixed to three registers)

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

- 16-bit instruction
  The STOH instruction converts the time data in the unit of second stored in [S] into data in the unit of hour-minute-second and stores the result in [D, D+1, D+2].

  The value in [S] ranges from 0 to 32767.
- 32-bit instruction
  The DSTOH instruction converts the time data in the unit of second stored in [S, S+1] into data in the unit of hour-minute-second and stores the result in [D, D+1, D+2].

  The value in [S, S+1] ranges from 0 to 117964799.

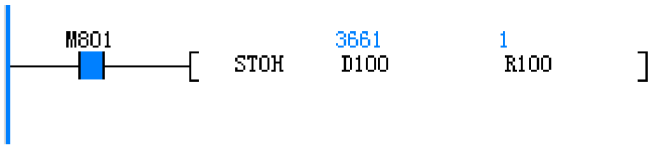
Errors

An error is reported and the instruction is not executed in the following cases:

- The operands of the 16-bit or 32-bit instruction are out of range.
- The time data (in the unit of second) to be converted in the 16-bit or 32-bit instruction is out of range.

Instruction Example

The time data in the unit of second stored in D100 is converted into data in the unit of hour-minute-second. The conversion result is stored in R100, R101, and R102.



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Dec	3661
	16-bit INT	Dec	
R100	16-bit INT	Dec	1
R101	16-bit INT	Dec	1
R102	16-bit INT	Dec	1
	16-bit INT	Dec	

5.12.8 TRD

The TRD instruction reads the data (year, month, day, hour, minute, second, day of week, and millisecond) of the internal real-time clock of the PLC and stores the read data in specified registers.

TRD -- Read clock data

16-bit Instruction	TRD: Continuous execution/TRDP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	Time storage start address	Start address of eight consecutive variable units that store the year, month, day, hour, minute, second, day of week, and millisecond in turn (from low to high)	-	INT, array*8

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	-	-	-	-

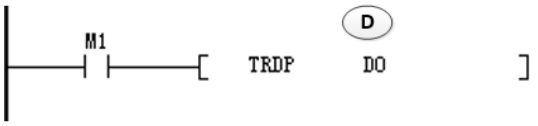
Function and Instruction Description

The TRD instruction reads the data (year, month, day, hour, minute, second, day of week, and millisecond) of the internal real-time clock of the PLC and stores the read data in specified registers.

The instruction of the pulse execution type (TRDP) is recommended.

D is the start address of the eight consecutive variable units that store the year, month, day, hour, minute, second, day of week, and millisecond in turn (from low to high).

Instruction Example



The following table describes the conversion process.

Item	→	Target Value
Year (2000 to 2038)	→	D0
Month (1 to 12)	→	D1
Day (1 to 31)	→	D2
Hour (0 to 23)	→	D3
Minute (0 to 59)	→	D4
Second (0 to 59)	→	D5

Item	→	Target Value
Day of week (0 to 6: Sunday to Saturday)	→	D6
Millisecond (0 to 999)	→	D7

NOTE

Generally, to use the clock of the PLC, you need to execute the TDR instruction to store the read the clock data in the registers specified in D.

5.12.9 TWR

The TWR instruction writes the clock data (year, month, day, hour, minute, second, and day of week) specified in S to the internal real-time clock of the PLC.

TWR -- Write clock data

16-bit Instruction	TWR: Continuous execution/TWRP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Start address for time data to be written	Start address of seven consecutive variable units that store the year, month, day, hour, minute, second, and day of week to be written in turn (from low to high)	-	INT, array*7

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-

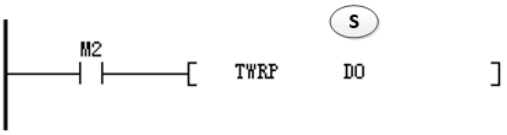
Function and Instruction Description

The TWR instruction writes the clock data (year, month, day, hour, minute, second, and day of week) specified in S to the internal real-time clock of the PLC.

The instruction of the pulse execution type (TWRP) is recommended.

S is the start address of the seven consecutive variable units that store the year, month, day, hour, minute, second, and day of week to be written in turn (from low to high).

Instruction Example



The following table describes the conversion process.

Data Source	→	Item
D0	→	Year (2000 to 2038)
D1	→	Month (1 to 12)
D2	→	Day (1 to 31)
D3	→	Hour (0 to 23)
D4	→	Minute (0 to 59)
D5	→	Second (0 to 59)
D6	→	Day of week (0 to 6: Sunday to Saturday)

NOTE

All of the seven data entries are written to the clock. All of the seven data entries must be preset. If the default values are used, the clock may be inactive. For example, if the day of week is not set, the default value 0 is used, which indicates Sunday. If the month is not set, the default value 0 is used, but the PLC considers it as a variable error. This makes the modification on the clock data invalid.

5.12.10 HOUR

When driving conditions are met, the HOUR instruction records time cumulatively. When the cumulative time reaches the preset value, a designated output becomes active.

HOUR: Hour meter

16-bit instruction	HOUR: continuous execution			
32-bit instruction	DHOUR: continuous execution			
Operands	Name	Description	Value range	Data type
S	Preset time	Preset time, measured in hours. When the cumulative time reaches the preset one, a designated output becomes active.	-	INT/DINT
D1	Cumulative time storage start unit	Start number of units that store the cumulative time	-	INT/DINT, array*2

D2	Time reached flag	Variable unit that outputs a time reached alert. When the cumulative time reaches the preset value, this unit is active.	-	BOOL

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D1	-	-	-	√	√	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-

NOTE

[1] The X element is not supported.

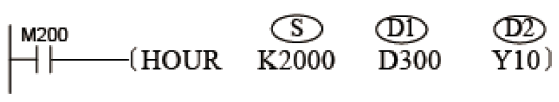
Function and Instruction Description

When driving conditions are met, the HOUR instruction records time cumulatively. When the cumulative time reaches the preset value, a designated output becomes active. Specifically:

- S is the preset time in the unit of hour. When the cumulative time reaches the preset value, the specified output becomes active.
- D1 is the start number of units that store the cumulative time.
- D2 is the variable unit that outputs a time reach alarm. When the cumulative time reaches the preset value, this unit becomes active.
- In 16-bit operation, the value in D1 ranges from K0 to K32,767, in the unit of hour. D1+1 stores the current time value less than 1 hour. The value ranges from K0 to K3599, in the unit of second. D1 occupies two units.
- In 32-bit operation, the value in D1+1 and D1 ranges from K0 to K2147483647, in the unit of second. D1+3 and D1+2 store the current time value less than 1 hour. The value ranges from K0 to K3599, in the unit of second. D1 occupies four units.

The time value in D1 cannot be negative. If D1 is specified as a register without support for retention upon power failure, the value in D1 is cleared when the PLC switches from STOP to RUN or when a power failure occurs. If you need to retain the current data in the case of a power failure, specify D1 as a register with support for retention upon power failure.

Example

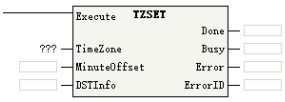


When M200 is ON, the time during which M200 remains ON is recorded cumulatively and stored in D300. If the time value is less than 1 hour, the equivalent value in the unit of second is recorded in D301. When the cumulative time in D300 reaches 2000 hours, Y10 turns ON.

After the cumulative time counted from when the timing condition is met reaches the preset value in S, the cumulative time value continues to increase. Timing stops when the current time value in D300 reaches 32767 hours or the value in D301 reaches 3599s. To restart timing, clear the values in D300 and D301.

5.12.11 TZSET

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TZSET	Set time zone		-

16-bit Instruction	TZSET: Triggered execution
32-bit Instruction	-

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	TimeZone	Integral time zone	No	-	[-15, 15][1]	INT
S2	Minute Offset	Minute offset of non-integral time zone	Yes	-	[0, 59][2]	INT
S3	DSTInfo	Daylight saving time (DST) information. It consists of the start addresses of nine consecutive variable units that store the time difference, start time (week in month, day of week, month, and hour), and end time (week in month, day of week, month, and hour) from low to high.	Yes	-	For details, see the description of elements in DSTInfo below.	INT
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D4	ErrorID	Error code	Yes	0	For details, see the	INT

					description of time zone error codes below.	

NOTE

[1]: The time zone range specified by using the TZSET instruction is greater than that configured on the configuration interface. If the time zone cannot be displayed on the configuration interface after execute the TZSET instruction, use the TZGET instruction to read the time zone information.

[2]: If the value of MinuteOffset is positive, that of TimeZone must also be positive, and vice versa. For example, if TimeZone is set to -8 and MinuteOffset is set to 30, the time zone set by this instruction is UTC-8:30. If TimeZone is set to 8 and MinuteOffset is set to 30, the time zone set by this instruction is UTC+8:30.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√[2]	-	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

NOTE

[1]: The X element is not supported.

[2]: Only custom variables of the INT[9] type are supported.

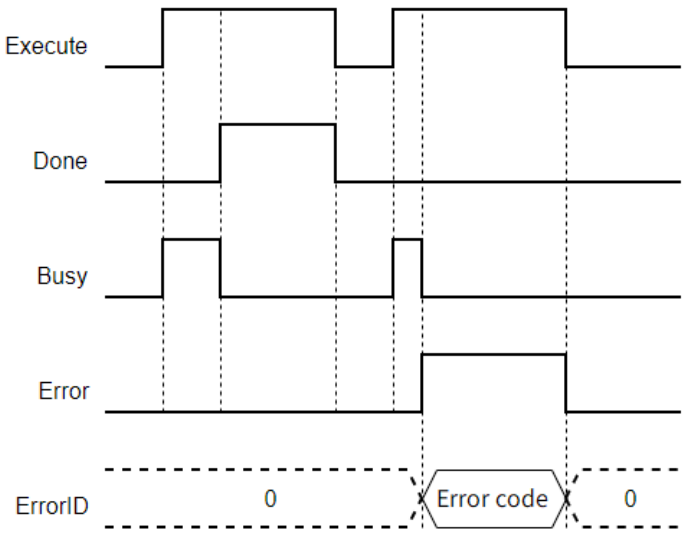
Function and Instruction Description

The TZSET instruction sets the time zone based on the input parameters when Execute receives a rising edge signal. When Execute remains at high level, you can determine the progress of time zone setting based on the output state.

Multiple TZSET instructions cannot be executed at the same time. The TZSET and TZGET instructions cannot be executed at the same time. Otherwise, other time zone instructions except the first one that is executed report error 5. It is recommended that the Done signal of the previous time zone instruction trigger Execute of the next time zone instruction. When you specify DSTInfo, the month in the start time and the end time cannot be adjacent or the same. Otherwise, error 4 is reported. For example, a DST variable of the INT[9] type

is defined and passed to DSTInfo, with the initial value of all-0s. When Execute is triggered, the function block reports error 4 because both DST[3] and DST[7] is 0, which indicates January.

Timing Diagram



The following table describes elements in DSTInfo.

Element	Description	Range	Remarks
INT[0]	Time difference in DST	[0, 120], in minutes	-
INT[1]	Start time: week in month	[0, 4]	0 indicates the first week in a month (day 1 to day 7), and 4 indicates the last week in a month.
INT[2]	Start time: day of week	[0, 6]	0 indicates Sunday and 6 indicates Saturday.
INT[3]	Start time: month	[0, 11]	0 indicates January and 11 indicates December.
INT[4]	Start time: hour	[0, 23], in hours	0 indicates 00:00:00 and 23 indicates 23:00:00.
INT[5]	End time: week in month	[0, 4]	0 indicates the first week in a month (day 1 to day 7), and 4 indicates the last week in a month.
INT[6]	End time: day of week	[0, 6]	0 indicates Sunday and 6 indicates Saturday.
INT[7]	End time: month	[0, 11]	0 indicates January and 11 indicates December.
INT[8]	End time: hour	[0, 23], in hours	0 indicates 00:00:00 and 23 indicates 23:00:00.

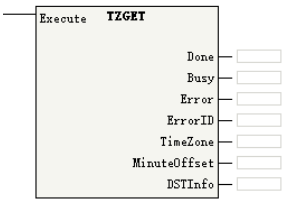
The following table describes time zone error codes.

Error Code	Description
0	No fault
1	Time zone reading failure
2	Time zone writing failure
3	Parameter out-of-range

Error Code	Description
4	DST start and end month the same or adjacent
5	Limited resource access of the same cycle

5.12.12 TZGET

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TZGET	Read time zone		-

The following table describes the instruction format.

16-bit Instruction	TZGET: Triggered execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D4	ErrorID	Error code	Yes	0	For details, see Time Zone Error Codes.	INT
D5	TimeZone	Integral time zone	Yes	0	[-15, 15]	INT
D6	Minute Offset	Minute offset of non-integral time zone	Yes	0	ON or OFF	INT
D7	DSTInfo	Daylight saving time (DST) information. It consists of the start addresses of nine consecutive variable units that store the	Yes	0	For details, see the description	INT[9]

		time difference, start time (week in month, day of week, month, and hour), and end time (week in month, day of week, month, and hour) from low to high.			of elements in DSTInfo below.	

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-
D6	-	-	-	√	√	√	-	-	-
D7	-	-	-	√	√	-	-	-	-

NOTE

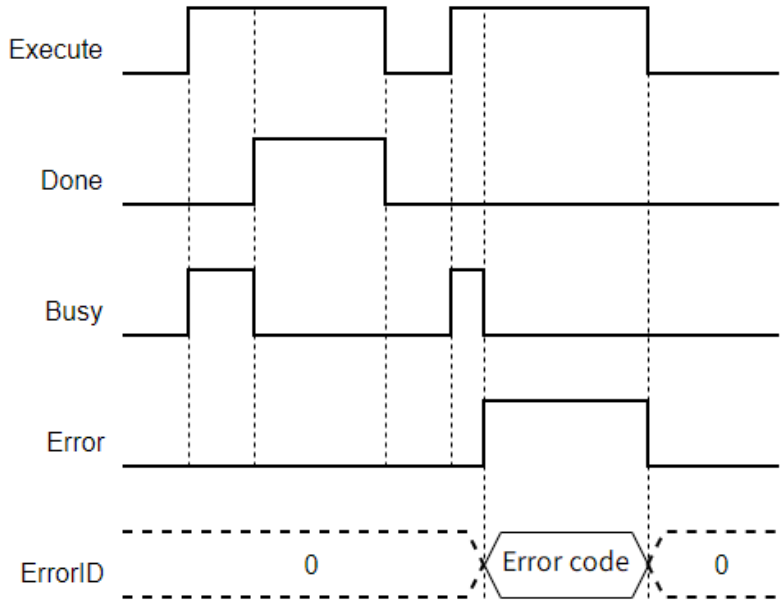
[1]: The X element is not supported.

Function and Instruction Description

The TZGET instruction reads the time zone when Execute receives a rising edge signal. When Execute keeps at high level, the output value is retained. When Execute is at low level, the output value restores the default.

Multiple TZGET instructions cannot be executed at the same time. The TZSET and TZGET instructions cannot be executed at the same time. Otherwise, other time zone instructions except the first one that is executed report error 5. It is recommended that the Done signal of the previous time zone instruction trigger Execute of the next time zone instruction.

Timing Diagram



5.13 MC Axis Control (EtherCAT and Pulse Output)

5.13.1 Basic Instructions

5.13.1.1 List of Instructions

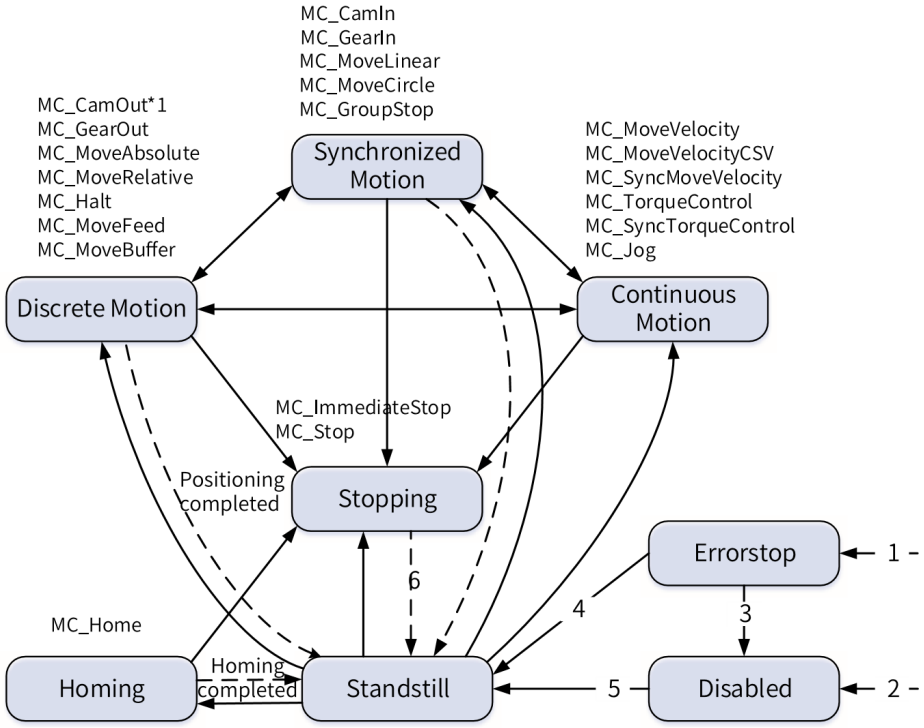
The following table lists the motion control axis instructions.

Instruction	Name	Language Support
MC_Power	Enable control	LD and LiteST
MC_Reset	Fault reset	LD and LiteST
MC_ReadStatus	Axis state read	LD and LiteST
MC_ReadAxisError	Axis error read	LD and LiteST
MC_ReadDigitalInput	Digital input read	LD and LiteST
MC_ReadActualPosition	Current position read	LD and LiteST
MC_ReadActualVelocity	Current velocity read	LD and LiteST
MC_ReadActualTorque	Current torque read	LD and LiteST
MC_SetPosition	Set current position	LD and LiteST
MC_TouchProbe	Probe Instructions	LD and LiteST
MC_MoveRelative	Relative positioning	LD and LiteST
MC_MoveAbsolute	Absolute positioning	LD and LiteST
MC_MoveVelocity	Velocity control	LD and LiteST

Instruction	Name	Language Support
MC_Jog	Jogging	LD and LiteST
MC_TorqueControl	Torque control	LD and LiteST
MC_Home	Homing	LD and LiteST
MC_Stop	Stop	LD and LiteST
MC_Halt	Halt	LD and LiteST
MC_ImmediateStop	Immediate stop	LD and LiteST
MC_MoveFeed	Interrupt positioning instruction	LD and LiteST
MC_MoveBuffer	Multi-position positioning	LD and LiteST
MC_MoveSuperImposed	Motion superimposition	LD and LiteST
MC_MoveVelocityCSV	CSV-based velocity control with adjustable pulse width	LD and LiteST
MC_SyncMoveVelocity	CSV-based synchronous velocity control with adjustable pulse width	LD and LiteST
MC_SyncTorqueControl	Synchronous torque control	LD and LiteST
MC_SetAxisConfigPara	Axis parameter configuration	LD and LiteST
MC_SetOverride	Target velocity adjusting during motion	LD and LiteST
MC_FollowVelocity	CSP-based synchronous velocity control	LD and LiteST
MC_MoveVelocityPV	Profile velocity control	LD
MC_MoveAbsRelPP	Positioning based on the profile position mode	LD
MC_JogP	Jogging based on the profile mode	LD
MC_MoveBufferPP	Multi-segment positioning in profile position mode	LD
MC_TorqueControlPT	Torque control in profile torque mode	LD
MC_ResetFollowingError	Following error reset	LD and LiteST

5.13.1.2 Description of MC Axis State Machine

The MC axis state machine manages the states and motions of axes based on the PLCOpen state machine.



The state machine is described as follows:

See the following table for state definitions.

Status value	Status	Function
0	Disabled	Disabled
1	ErrorStop	Stop upon fault state
2	Stopping	Stopping
3	StandStill	Enabled
4	DiscreteMotion	Discrete motion
5	ContinuousMotion	Continuous motion
7	Homing	Homing
8	SynchronizedMotion	Synchronized motion

The following table summarizes the status transition conditions.

Transition	Transition condition
1	The fault detection logic of the axis has detected a fault.
2	The axis has no fault and MC_Power.Enable is OFF.

Transition	Transition condition
3	MC_Reset is executed to reset the axis fault and MC_Power.Status is OFF.
4	MC_Reset is called to reset the axis fault and MC_Power.Status=ON.
5	Both MC_Power.Enable and MC_Power.Status are ON.
6	MC_Stop(MC_ImmediateStop).Done is ON and MC_Stop(MC_ImmediateStop).Execute is OFF.

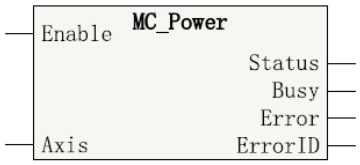
Precision of Instruction Parameters

The floating-point numbers such as the target position and target velocity in the instructions are single-precision floating-point data. Therefore, the values in the instructions must meet the requirements of the range and precision of single-precision floating-point data when being processed in the PLC program. That is, a value should fall between -3.4E38 and +3.4E38, with a maximum of 7 significant digits. If a value has more than 7 significant digits, the excess part will be automatically rounded.

5.13.1.3 MC_Power

MC_Power -- Enable control

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_Power	Enable control		MC_Power(Enable := ???, Axis := ???, Status => , Busy => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	MC_Power: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
D1	Status	Axis enable flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL

D3	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D4	ErrorID	Error code*1	Yes	0	-	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

The MC_Power instruction sets the enable state of the axis. It is active at high level. It is applicable to the EtherCAT bus axis and the local pulse axis.

- Specifying the axis
  Axis is latched at the rising edge of the Enable input.

  If you access an axis by using the axis ID and modify Axis when Enable is ON, an instruction error occurs and the previously controlled axis is disabled.

  When you access an axis by using the axis ID and modify Axis when Enable is OFF, if the axis specified by Axis is enabled and the MC_Power instruction is the last instruction executed in a PLC scan cycle, the axis is disabled and becomes non-operational.

- Function description
  When Enable is set to ON, the axis is enabled, and the Status signal of the instruction is active.

  The PLCopen state machine of the axis transitions from the Disabled state to the StandStill state.

  After the axis is enabled, it can execute motion instructions such as MC_MoveRelative.

When Enable is set to OFF, the enable state of the axis is cleared and the execution of motion instructions (such as MC_MoveAbsolute) is interrupted. You cannot control the axis because it does not acknowledge motion instructions. However, you can still execute non-motion instructions such as MC_Power, MC_Reset, and MC_SetPosition.

When the axis enters the ErrorStop state upon a fault, re-enabling MC_Power cannot switch the axis to the StandStill state. You must call the MC_Reset instruction first to restore the axis.

- Multi-execution
  When multiple MC_Power instructions are executed, the control flow of the last MC_Power instruction executed in a cycle prevails.

Errors

An error is reported when the axis ID does not exist.

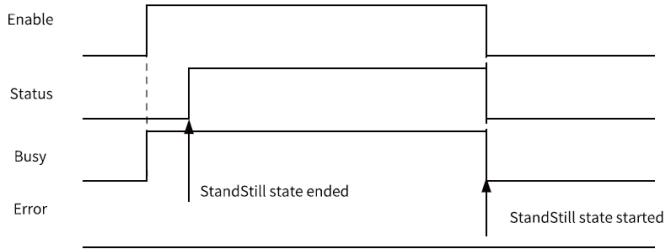
An error is reported when the axis type is incorrect.

An error is reported when axis initialization fails.

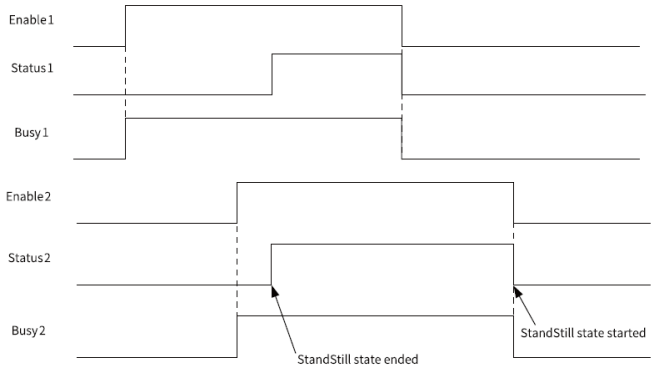
An error is reported when the PDOs of the axis do not contain control word, status word, target position, and current position.

Timing Diagram

- When an MC_Power instruction is executed, the axis can be enabled normally.



- When two MC_Power instructions are executed, the axis can be enabled normally.

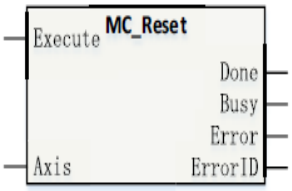


5.13.1.4 MC_Reset

MC_Reset -- Fault reset

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_Reset	Fault reset		MC_Reset(Execute := ???, Axis := ???,

Instruction	Name	LD Expression	LiteST Expression
			Done => , Busy => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	MC_Reset: Continuous execution
32-bit Instruction	-

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
D1	Done	Reset completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D4	ErrorID	Error code*1	Yes	0	-	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

The MC_Reset instruction resets faults of the axes. It is applicable to the EtherCAT bus axis and the local axis and is triggered on the rising edge.

The MC_Reset instruction attempts to reset the fault of the axis on the rising edge of the Execute signal. If the reset is successful, the Done output is active. Otherwise, the Error signal is active, and ErrorID specifies the reason for the reset failure.

After the reset is successful, the PLCopen state machine of the axis transitions into the StandStill state if the drive is enabled or the Disabled state if the drive is disabled.

Abortion

This instruction does not abort output signals and cannot be aborted during execution.

If there are two reset instructions in one scan cycle, the program starts to execute the reset logic as long as one reset instruction is active. If the reset is successful, the Done signal output of the triggered instruction becomes active.

Errors

An error is reported when the axis ID does not exist.

An error is reported when the axis type is incorrect.

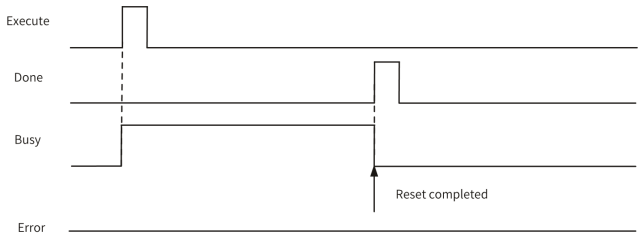
An error is reported when axis initialization fails.

An error is reported if this instruction is executed when the axis has no fault.

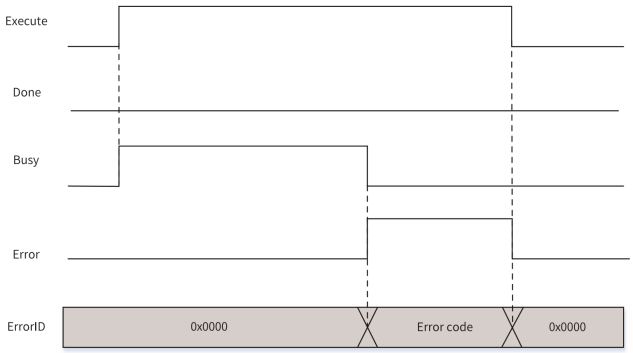
An error is reported when the axis cannot be reset.

Timing Diagram

- A fault occurs on the axis, and the MC_Reset instruction is executed to reset the axis fault successfully.



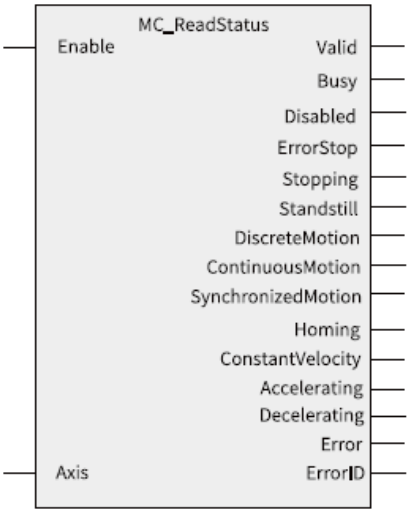
- A non-resettable fault occurs on the drive.



5.13.1.5 MC_ReadStatus

MC_ReadStatus: Axis state read

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_ReadStatus	Read the axis state		MC_ReadStatus(Enable := ???, Axis := ???, Valid => , Busy => , Disabled => , ErrorStop => , Stopping => , Standstill => , DiscreteMotion => , ContinuousMotion => , SynchronizedMotion => , Homing => , ConstantVelocity => , Accelerating => , Decelerating => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	MC_ReadStatus continuous execution
32-bit instruction	-

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/ Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
D1	Valid	Enabled	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Disabled	PLCOpen state machine, disabled	Yes	OFF	ON/OFF	BOOL
D4	ErrorStop	PLCOpen state machine, stopping upon a fault	Yes	OFF	ON/OFF	BOOL
D5	Stopping	PLCOpen state machine, stopping	Yes	OFF	ON/OFF	BOOL
D6	StandStill	PLCOpen state machine, enabled and not running	Yes	OFF	ON/OFF	BOOL
D7	DiscreteMotion	PLCOpen state machine, discrete motion	Yes	OFF	ON/OFF	BOOL
D8	Continuous Motion	PLCOpen state machine, continuous motion mode	Yes	OFF	ON/OFF	BOOL
D9	SynchronizedMotion	PLCOpen state machine, synchronized motion mode	Yes	OFF	ON/OFF	BOOL

D10	Homing	PLCOpen state machine, homing	Yes	OFF	ON/OFF	BOOL
D11	ConstantVelocity	The axis velocity is 0. The axis is moving at a constant speed. Invalid in torque mode	Yes	OFF	ON/OFF	BOOL
D12	Accelerating	The axis is accelerating (the absolute value of the velocity is increasing). Invalid in torque mode	Yes	OFF	ON/OFF	BOOL
D13	Decelerating	The axis is decelerating (the absolute value of the velocity is decreasing). Invalid in torque mode	Yes	OFF	ON/OFF	BOOL
D14	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D15	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Fault Code.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	√[1]	√	√	-	-	√	-	-	-
D6	√[1]	√	√	-	-	√	-	-	-
D7	√[1]	√	√	-	-	√	-	-	-
D8	√[1]	√	√	-	-	√	-	-	-
D9	√[1]	√	√	-	-	√	-	-	-
D10	√[1]	√	√	-	-	√	-	-	-
D11	√[1]	√	√	-	-	√	-	-	-
D12	√[1]	√	√	-	-	√	-	-	-
D13	√[1]	√	√	-	-	√	-	-	-
D14	√[1]	√	√	-	-	√	-	-	-
D15	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Function

When Enable is ON, this instruction reads the acceleration/deceleration state and state of the PLCOpen state machine of the axis.

In torque mode, ConstantVelocity, Acceleration, and Deceleration are always OFF.

The priority of a EtherCAT task is higher than that of a PLC master task. If the state of the axis exists only for one EtherCAT cycle in the EtherCAT task, the state cannot be obtained in the PLC master task.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is returned in the following conditions: The axis number does not exist.

Axis initialization fails.

The axis type is incorrect.

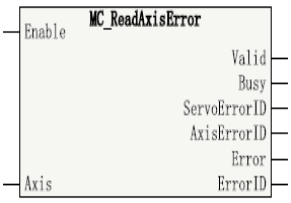
Sequence Diagram

No

5.13.1.6 MC_ReadAxisError

MC_ReadAxisError -- Read axis errors

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_ReadAxisError	Read axis errors		MC_ReadAxisError(Enable := ???, Axis := ???, Valid => , Busy => , ServoErrorID => , AxisErrorID => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	MC_ReadAxisError: Continuous execution					
32-bit Instruction	-					
Operand	**Name**	**Description**	**Empty Allowed**	**Default**	**Range**	**Data Type**
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
D1	Valid	Active	Yes	OFF	ON or OFF	BOOL

D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	ServoErrorID	Servo error code. If 0x603F is configured in the PDO, the value of 0x603F of the EtherCAT bus drive displayed. Otherwise, 0 is displayed.	Yes	0	*2	INT
D4	AxisErrorID	Axis error code	Yes	0	*1	INT
D5	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D6	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

*2: For a local pulse axis, see the list of local pulse axis error codes. For an EtherCAT bus drive, see the relevant manual of the EtherCAT bus drive.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	√[1]	√	√	-	-	√	-	-	-
D6	-	-	-	√	√	√	-	-	

NOTE

[1]: The X element is not supported.

Function and Instruction Description

The MC_ReadAxisError instruction reads the fault of the EtherCAT bus axis or the local pulse axis.

When Enable is ON, the Valid signal becomes active if the requested axis exists and no configuration failure occurs. AxisErrorID displays the error code of the axis in real time. When no fault occurs, the value is 0. When an axis fault occurs, it displays the error code. If 0x603F is configured in the PDO of the bus drive, ServoErrorID displays the value of 0x603F in real time. Otherwise, it displays 0.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is reported when the axis ID does not exist.

An error is reported when axis initialization fails.

An error is reported when the axis type is incorrect.

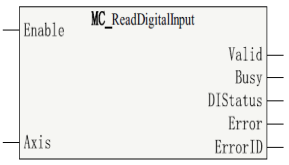
Timing Diagram

None

5.13.1.7 MC_ReadDigitalInput

MC_ReadDigitalInput -- Read digital input

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_ReadDigitalInput	Read DI state		MC_ReadDigitalInput(Enable := ???, Axis := ???, Valid => , Busy => , DIStatus => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_ReadDigitalInput: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type

S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
D1	Valid	Active	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	DIStatus	DI terminal state compliant with CiA 402. The standard format defined in the DS 402 protocol as follows: Bit 0: Reverse limit signal Bit 1: Forward limit signal Bit 2: Home signal Bit 3 to bit 31: Customized	Yes	0	-	DINT
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

> **NOTE**
>
> [1]: The X element is not supported.

Function and Instruction Description

The MC_ReadDigitalInput instruction reads the DI terminal state of the axis. It is applicable to the EtherCAT bus axis and the local pulse axis and does not support the virtual axis mode.

When Enable is ON, if 0x60fd is configured in the PDO of the requested EtherCAT bus axis or any of the left and right limit and home signals of the local pulse axis is not empty, the Valid signal is active.

For the EtherCAT bus axis, DIStatus displays the digital input 0x60fd of the EtherCAT bus drive in real time. For details, see the relevant driver manual.

For the local pulse axis, DIStatus displays the input states of the limit and home signals or 0.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

- An error is reported when the axis ID does not exist.
- An error is reported when axis initialization fails.
- An error is reported when the axis type is incorrect.
- An error is reported when 0x60fd is not configured in the PDO of the EtherCAT bus axis.

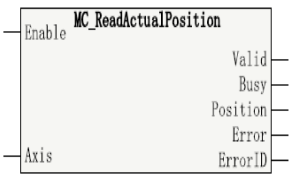
Timing Diagram

None

5.13.1.8 MC_ReadActualPosition

MC_ReadActualPosition -- Read actual position

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_ReadActualPosition	Read actual position		MC_ReadActualPosition(Enable := ???, Axis := ???, Valid => , Busy => , Position => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_ReadActualPosition: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
D1	Valid	Active	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Position	Current position	Yes	0	-	REAL
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	-	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	-	-	-	-

> **NOTE**
>
> [1]: The X element is not supported.

Function and Instruction Description

The MC_ReadActualPosition instruction reads the feedback position of the EtherCAT bus axis or the local pulse axis. It is active at high level.

When Enable is ON, if 0x6064 is configured in the PDO of the EtherCAT bus axis, the Valid signal is active, and Position displays the feedback position of the axis.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is reported when the axis ID does not exist.

An error is reported when axis initialization fails.

An error is reported when the axis type is incorrect.

An error is reported when 0x6064 is not configured in the PDO of the EtherCAT bus axis.

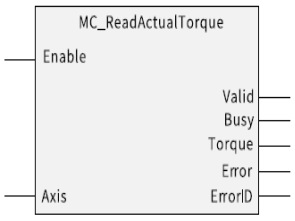
Timing Diagram

None

5.13.1.9 MC_ReadActualTorque

MC_ReadActualTorque -- Read actual torque

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_ReadActualTorque	Read actual torque		MC_ReadActualTorque(E nable := ???, Axis := ???, Valid => , Busy => , Torque => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-
32-bit Instruction	MC_ReadActualTorque: Continuous execution

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
D1	Valid	Active	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Torque	Current torque, in 1%	Yes	0	Positive number, negative number, or 0	REAL
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	-	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	-	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

The MC_ReadActualTorque instruction reads the feedback torque of the EtherCAT bus axis. It is active at high level. It does not support the virtual axis mode.

When Enable is ON, if 0x6077 is configured in the PDO of the EtherCAT bus axis, the Valid signal is active, and Torque displays the feedback torque of the axis.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is reported when the axis ID does not exist.

An error is reported when the axis type is incorrect.

An error is reported when axis initialization fails.

An error is reported when 0x6077 is not configured in the PDO of the EtherCAT bus axis.

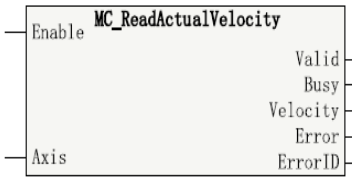
Timing Diagram

None

5.13.1.10 MC_ReadActualVelocity

MC_ReadActualVelocity -- Read actual velocity

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_ReadActualVelocity	Read actual velocity		MC_ReadActualVelocity(Enable := ???, Axis := ???, Valid => , Busy => , Velocity => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_ReadActualVelocity: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis		No	-	0 to 32,767	INT

		Axis name or axis ID				_sMCAXIS_INFO
D1	Valid	Active	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Velocity	Current velocity	Yes	0	-	REAL
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	-	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	-	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

The MC_ReadActualVelocity instruction calculates the actual velocity based on the feedback velocity of the EtherCAT bus axis or the local pulse axis. It is active at high level.

When Enable is ON, if 0x6064 is configured in the PDO of the EtherCAT bus axis, the Valid signal is active, and Velocity displays the calculated feedback velocity of the axis.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is reported when the axis ID does not exist.

An error is reported when axis initialization fails.

An error is reported when the axis type is incorrect.

An error is reported when 0x6064 is not configured in the PDO of the EtherCAT bus axis.

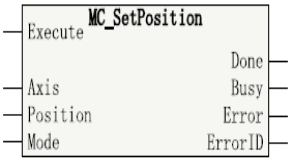
Timing Diagram

None

5.13.1.11 MC_SetPosition

MC_SetPosition -- Set the current position

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_SetPosition	Set the current position		MC_SetPosition(Execute := ???, Axis := ???, Position := ???, Mode := , Done => , Busy => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_SetPosition: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
S2		Target position	No	-		REAL

	Position				Positive number, negative number, or 0	
S3	Mode	Control mode selection 0: Absolute mode (Write the value of Position as the current position.) 1: Relative mode (Add the value of Position based on the current position.)	Yes	0	0 to 1	INT
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D4	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	-	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

The MC_SetPosition instruction sets the current position of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

This instruction can be executed only when the PLCOpen state of the axis is Disabled, StandStill, or ErrorStop. Otherwise, an error occurs.

- When Mode is set to 0 (absolute mode), this instruction writes the value of Position as the current position of the axis on the rising edge of the Execute signal.
- When Mode is set to 1 (relative mode), this instruction adds the value of Position based on the current position of the axis on the rising edge of the Execute signal.

Abortion

This instruction does not support abortion. If there are several MC_SetPosition instructions in one scan cycle at the same time, the first active instruction is executed. If other SetPosition instructions are executed during the period when the Busy signal of this instruction is active, other instructions report an error.

Errors

An error is reported when the axis ID does not exist.

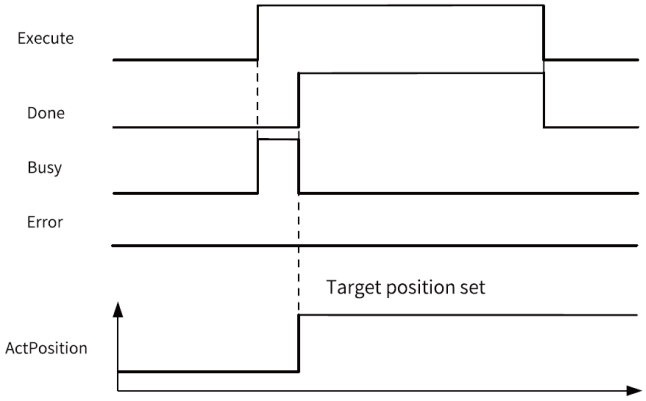
An error is reported when the axis type is incorrect.

An error is reported when axis initialization fails.

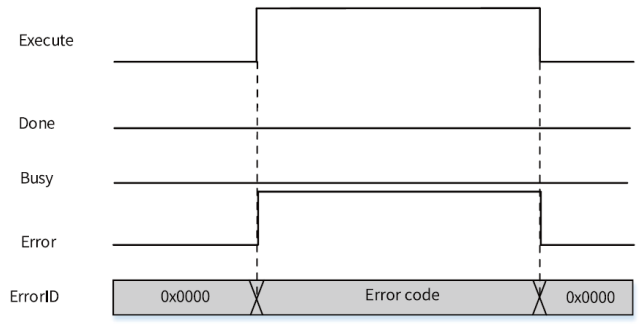
The MC_SetPosition instruction takes effect only when the axis has stopped operation, and it reports an error if the axis is in another state.

Timing Diagram

- The MC_SetPosition instruction (relative mode) is executed when the axis is in StandStill state.



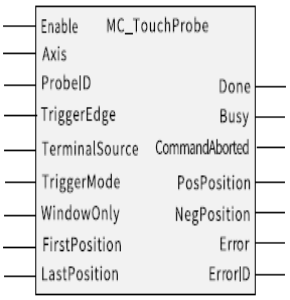
- The MC_SetPosition instruction is executed during execution of the jogging instruction.



5.13.1.12 MC_TouchProbe

MC_TouchProbe: Probe

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_TouchProbe	Probe		MC_TouchProbe(Enable := ???, Axis := ???, ProbeID := ???, TriggerEdge := ???, TerminalSource := , TriggerMode := , WindowOnly := , FirstPosition := , LastPosition := , Done => , Busy => , CommandAborted => , PosPosition => , NegPosition => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	MC_TouchProbe: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO

S2	ProbeID	Probe ID 0: Probe 1 1: Probe 2	No	-	0 to 1	INT
S3	TriggerEdge	Trigger edge 0: Only rising edge 1: Only falling edge 2: Both rising edge and falling edge	No	-	0 to 2	INT
S4	Terminal Source	Probe signal source (only for setting bus servo drive) 0: DI terminal 1: Encoder Z signal	Yes	0	0 to 1	INT
S5	TriggerMode	Trigger type 0: Single trigger 1: Continuous trigger	Yes	0	0 to 1	INT
S6	Window Only	Probe window enable 0: Disabled. Probe signals are detected at all positions. 1: Enabled. Probe signals are detected only when the current position is between FirstPosition and LastPosition (included).	Yes	OFF	ON/OFF	BOOL
S7	FirstPosition	Probe window start position	Yes	0	Positive number, negative number, or 0	REAL
S8	LastPosition	Probe window end position	Yes	0	> First position	REAL
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CmdAborted	Abortion	Yes	OFF	ON/OFF	BOOL

D4	PosPosition	Position latched on the rising edge	Yes	0	Positive number, negative number, or 0	REAL
D5	NegPosition	Position latched on the falling edge	Yes	0	Positive number, negative number, or 0	REAL
D6	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D7	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see 5.13.1.37 Axis Fault Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	–	–	–	√	√	√	√	–	–
S2	–	–	–	√	√	√	√	–	–
S3	–	–	–	√	√	√	√	–	–
S4	–	–	–	√	√	√	√	–	–
S5	–	–	–	√	√	√	√	–	–
S6	√	√	√	–	–	√	–	–	–
S7	–	–	–	√	√	√	–	√	–
S8	–	–	–	√	√	√	–	√	–
D1	√[1]	√	√	–	–	–	–	–	–
D2	√[1]	√	√	–	–	–	–	–	–
D3	√[1]	√	√	–	–	–	–	–	–

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D4	√[1]	√	√	–	–	–	–	–	–
D5	–	–	–	√	√	–	–	–	–
D6	–	–	–	√	√	–	–	–	–
D7	–	–	–	√	√	–	–	–	–

NOTE

[1] The X element is not supported.

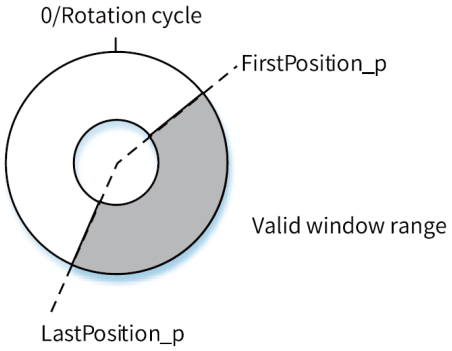
Function and Instruction Description

This instruction implements the probe function of the EtherCAT bus axis or the local pulse axis. It is active high. It does not support the imaginary axis mode.

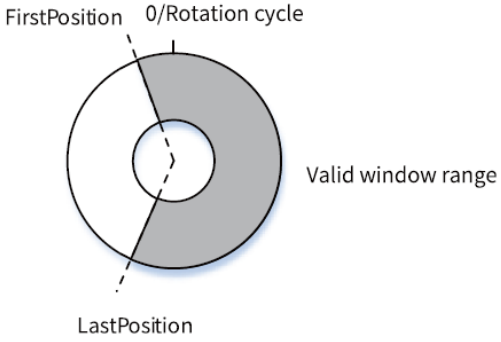
In EtherCAT bus axis mode, the probe function (0x60b8), probe state (0x60b9), and latched position (0x60ba /0x60bb/0x60bc/0x60bd) need to be configured for the drive.

In the local pulse axis mode, the probe signal source needs to be configured.

- On the rising edge, the instruction latches the input parameters on the left, such as ProbeID and TriggerEdge, and other state update parameters are invalid.
- When Enable is ON, the function block latches the current position of the axis when the instruction detects that the input of the probe specified by ProbeID is active and meets the probe detection conditions.
- When WindowOnly is OFF, the window detection function is disabled. The axis position can be latched as long as the probe input signal is active.
- When WindowOnly is ON, the window detection function is enabled.
- In linear mode, the instruction detects the probe signal only when the current position of the axis falls within the range specified by FirstPosition and LastPosition.
- In ring mode, the instruction first uses FirstPosition and LastPosition to mod the cycle to obtain FirstPosition_p and LastPosition_p in a cycle.
  - When FirstPosition_p is less than LastPosition_p, the valid window range is as follows:



◦ When FirstPosition_p is greater than LastPosition_p, the valid window range is as follows:



◦ When FirstPosition_p equals to LastPosition_p, no valid window range is available, and Error 9107 is reported.

This instruction can detect the rising edge or falling edge of the probe signal separately or both the rising edge and the falling edge at the same time.

When detecting only the rising edge (falling edge), the instruction writes the value detected on the rising edge (falling edge) into PosPosition (NegPosition). At this time, the Done signal is set to ON when a detection cycle is completed.

If the rising edge and falling edge are detected at the same time, after the Enable signal is active, the instruction immediately writes the position into PosPosition upon detecting the rising edge and writes the position into NegPosition upon detecting the falling edge. After that, the detection cycle is completed and the Done signal is output. There is no requirement on the input sequence of the rising edge and falling edge.

- For the EtherCAT bus driver, the input TerminalSource of this instruction can be used to set the terminal type to DI or Z signal of the motor (driver support required). No error is reported if the driver does not support the Z signal.
- This instruction supports the single trigger and continuous trigger modes. If the single trigger mode is used, instruction execution ends when the Done signal output is active. If the continuous trigger mode is used, the Done output active signal is reset after one PLC scan cycle, and the instruction automatically starts to detect new probe input signals.

NOTE

When the window function is enabled, probe signal loss or detection out-of-range may occur near the window area. The following is an example:

- In linear mode, the window range is 10 to 100, the EtherCAT cycle is set to 8 ms, and the velocity is 100 (user unit/s). Then, the axis moves 0.8 (user unit) per EtherCAT cycle. The user unit is set based on the customer axis settings. If the current position at the moment when an EtherCAT cycle starts is 9.9, the probe signal is not detected within this EtherCAT cycle. The current position changes to 10.7 upon start of the next EtherCAT cycle. Therefore, the probe signals between 10 and 10.7 are lost. If the current position at the moment when an EtherCAT cycle starts is 99.9, the probe signal is detected within this EtherCAT cycle. The current position changes to 100.7 upon start of the next EtherCAT cycle. Therefore, the probe signals between 100 and 100.7 are responded.
- In continuous mode, if the input frequency of the probe signal is greater than the frequency of the PLC scan cycle, some probe signals are lost.

Abortion

The MC_TouchProbe instruction supports the detection probe 1 and probe 2. If two probe instructions are defined in the program and the probe IDs of the two instructions are different, the two probe instructions will work independently. If the probe IDs are the same, the probe instruction executed later will abort the previous probe instruction.

The MC_MoveFeed instruction also uses the probe signal. The execution rules of these two instructions are as follows:

- When an MC_TouchProbe instruction and an MC_MoveFeed coexist in a program, if their probe IDs are different, the two instructions work independently.
- If their probe IDs are the same, the situation is as follows: If the MC_TouchProbe instruction is enabled first, and the Mc_MoveFeed instruction is enabled while the Busy signal of the MC_TouchProbe instruction is still active, the MC_TouchProbe instruction is aborted. If the Mc_MoveFeed instruction is enabled first, and the MC_TouchProbe instruction is triggered while the Busy signal of the Mc_MoveFeed instruction is active but the InFeed signal is inactive, the MC_TouchProbe instruction reports an error.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

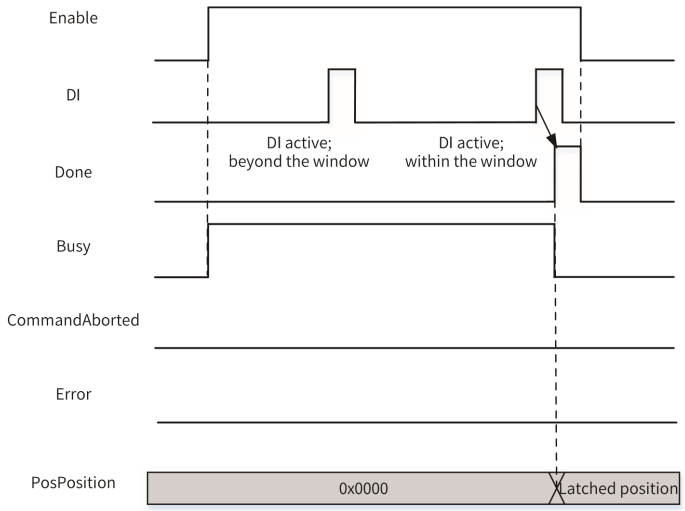
Error 9133 is reported when the imaginary axis mode is enabled.

A fault is reported when the corresponding PDO instruction is not configured.

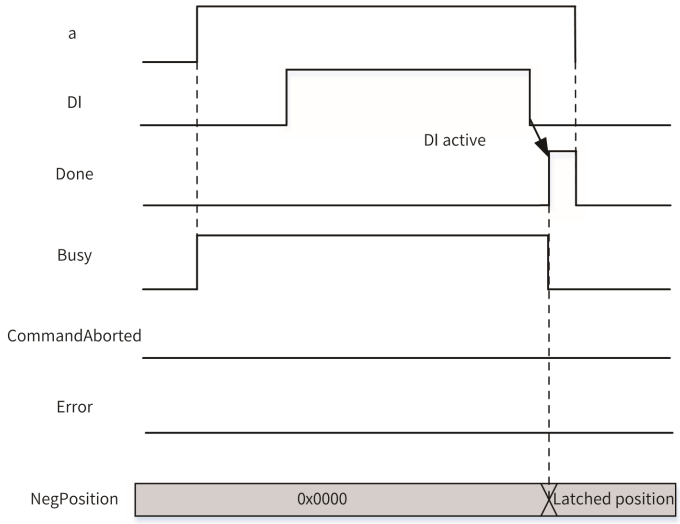
A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Enable input.

Sequence Diagram

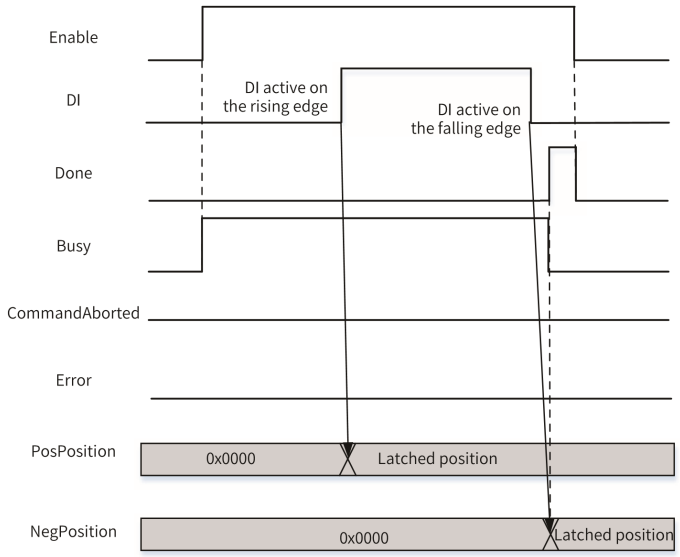
- Probe 1, active on the rising edge, DI signal trigger source, single trigger mode, window function enabled



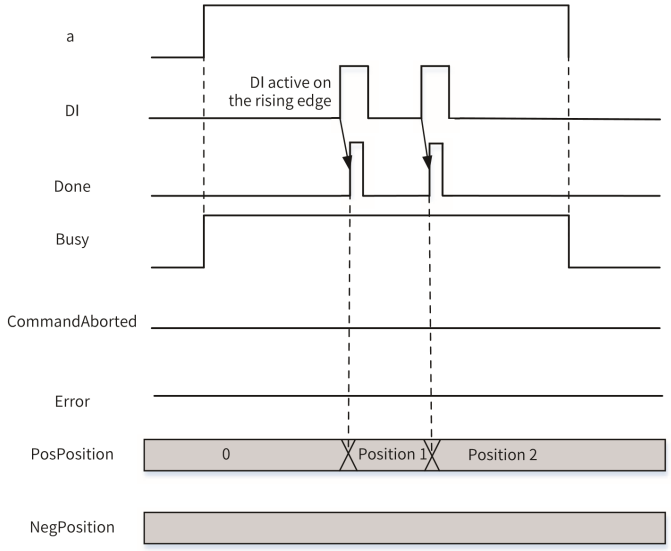
- Probe 1, active on the falling edge, DI signal trigger source, single trigger mode, window function disabled



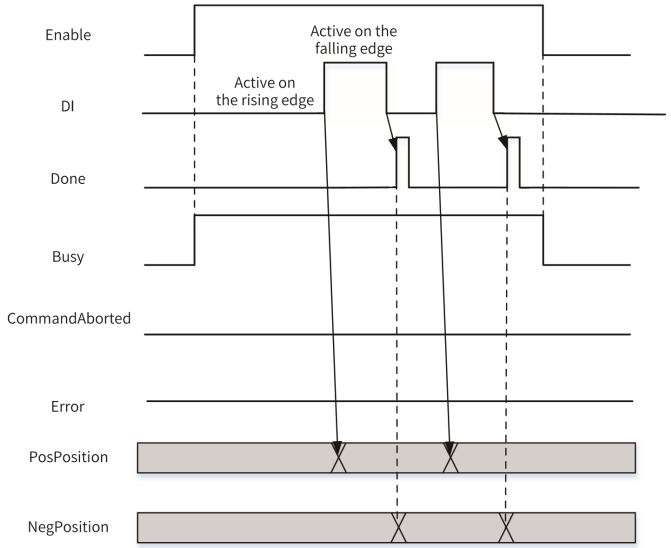
- Probe 1, active on both the rising edge and falling edge, DI signal trigger source, single trigger mode, window function disabled



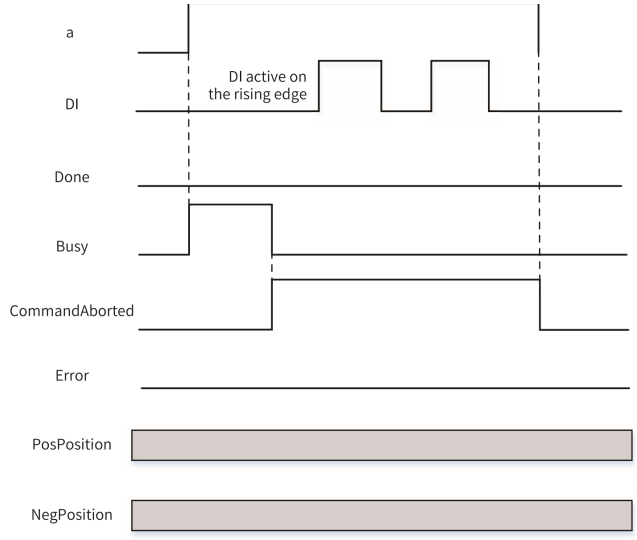
- Probe 1, active on the rising edge, DI signal trigger source, continuous trigger mode, window function disabled



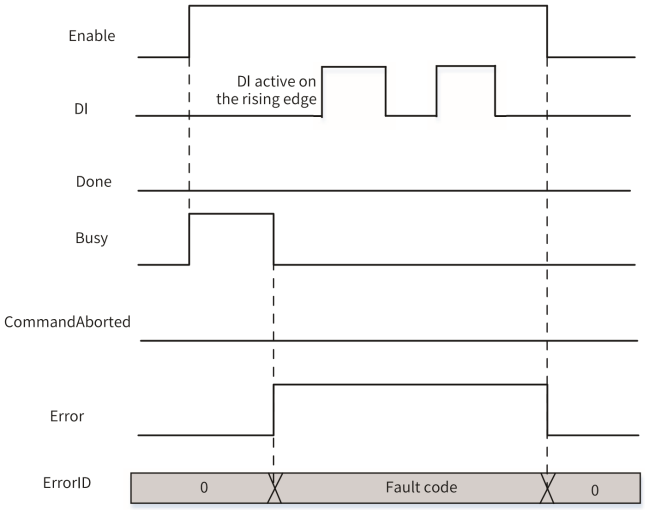
- Probe 1, active on both the rising edge and falling edge, DI signal trigger source, continuous trigger mode (the Done signal is active for a cycle after the DI signal is active on both the rising and falling edges), window function disabled



- Probe 1, aborted by another probe-related instruction, window function disabled



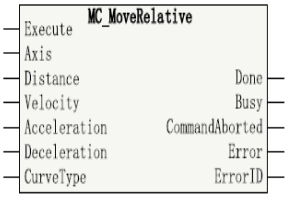
- Probe 1 instruction error



5.13.1.13 MC_MoveRelative

MC_MoveRelative -- Relative positioning

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveRelative	Relative positioning		MC_MoveRelative(Execute := ???, Axis := ???, Distance := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_MoveRelative: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO

S2	Distance	Target position	No	-	Positive number, negative number, or 0	REAL
S3	Velocity	Target velocity	No	-	Positive number, less than the maximum velocity	REAL
S4	Acceleration	Acceleration rate	No	-	Positive number, less than the maximum acceleration rate	REAL
S5	Deceleration	Deceleration rate	Yes	Acceleration rate	Positive number, less than the maximum acceleration rate	REAL
S6	CurveType	Curve type 0: T-shaped velocity curve 1: 5-segment S-shaped curve	Yes	0	0 to 1	INT
D1	Done	Target position reached	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	CmdAborted	Abortion of execution	Yes	OFF	ON or OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	-	√	-
S6	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

NOTE

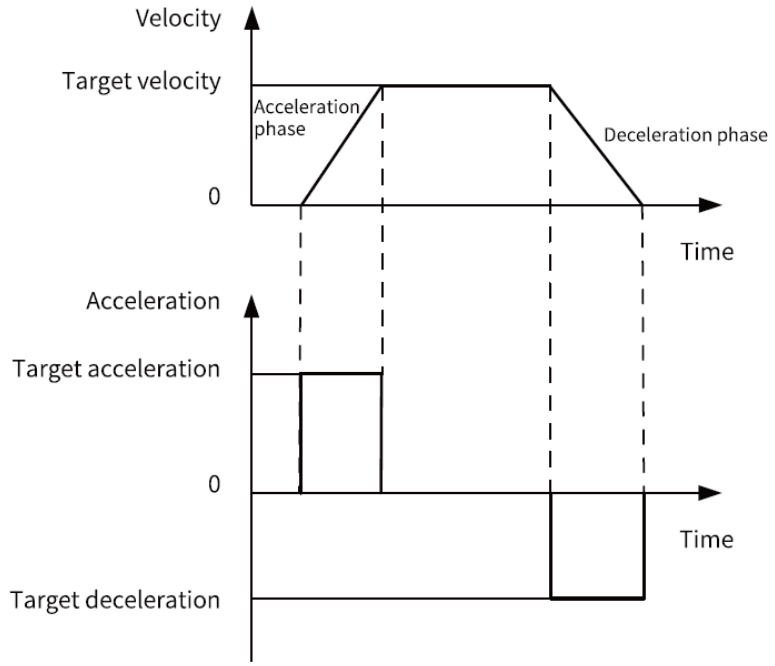
[1]: The X element is not supported.

Function and Instruction Description

The MC_MoveRelative instruction implements relative positioning of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis ID, modification on Axis is invalid when Execute is ON or OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the Execute input, the instruction latches the input parameters on the left, such as Distance and Velocity, triggers the relative positioning function, and switches the PLCOpen state machine of the axis to the DiscreteMotion state.
  - Distance specifies the distance for relative positioning. No matter in linear mode or ring mode, if Distance is a positive number, the axis travels the distance specified by Distance in the forward direction. If Distance is a negative number, the axis travels in the reverse direction by the distance specified by Distance.
  - CurveType specifies the type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration.



Target position: final position of the axis in the absolute positioning instruction, in Unit

Target velocity: maximum velocity of the axis, in Unit/s

Target acceleration: change in velocity per second during acceleration, in Unit/s2

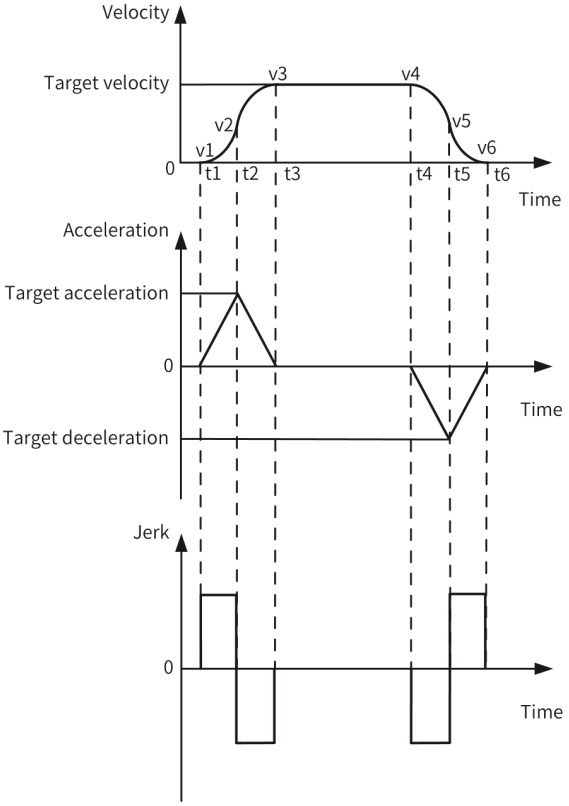
Target deceleration: change in velocity per second during deceleration in Unit/s2

During acceleration, assume that the initial velocity of the axis is Vs, the target velocity is Vt, and the target acceleration is Acc. Then, the acceleration time is calculated as follows:

Tacc = (Vt - Vs)/Acc
During deceleration, assume that the initial velocity of the axis is Vs, the target velocity is Ve, and the target deceleration is Dec. Then, the deceleration time is calculated as follows:

Tdec = (Vs - Ve)/Dec

- If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration rate and minimum deceleration rate of the axis during acceleration and deceleration.



The 5-segment S-curve is divided into five segments based on the acceleration state: increasing-acceleration, decreasing-acceleration, constant velocity, increasing-deceleration, and decreasing-deceleration. Constant acceleration or deceleration does not exist. The actual jerk during the variable-acceleration phase (such as increasing-acceleration and increasing-deceleration) is calculated internally by the PLC and cannot be set by the user.

Target position: final position of the axis in the absolute positioning instruction, in Unit

Target velocity: maximum velocity of the axis, in Unit/s

Target acceleration: maximum change in velocity per second during variable-acceleration operation, in Unit/s2. This parameter indicates the acceleration at the moment (t2) when the velocity changes from the increasing-acceleration stage to the decreasing-acceleration stage in the velocity curve.

Target deceleration: maximum change in velocity per second during variable-deceleration operation, in Unit/s2. This parameter indicates the deceleration at the moment (t5) when the velocity changes from the decreasing-acceleration stage to the decreasing-deceleration stage in the velocity curve.

During acceleration, assume that the initial velocity of the axis is V1, the target velocity is V3, and the target acceleration is Acc. Then, the acceleration time is calculated as follows:

Tacc = 2 x (V3 - V1)/Acc
During deceleration, assume that the initial velocity of the axis is V4, the target velocity is V6, and the target deceleration is Dec. Then, the deceleration time is calculated as follows:

Tdec = 2 x (V4 - V6)/Dec

Errors

Error 9101 is reported when the axis ID does not exist or the axis type does not match.

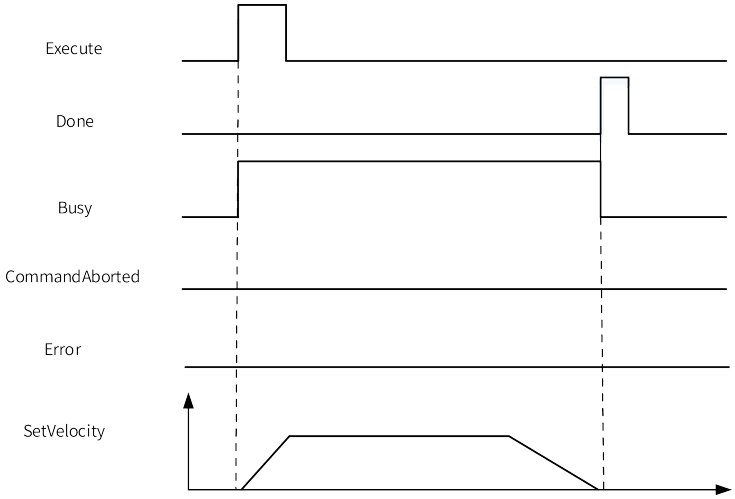
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, or ContinuousMotion.

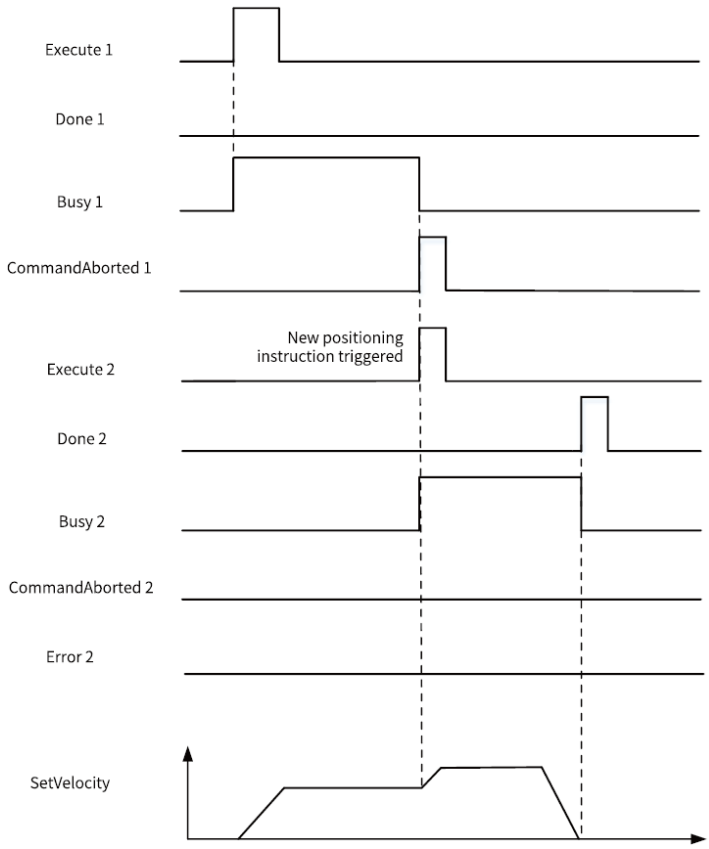
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- An error is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

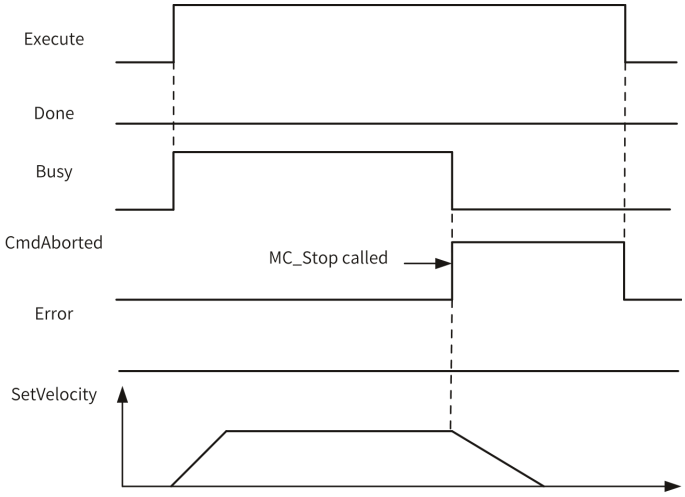
- The MC_MoveRelative instruction is executed to implement relative positioning based on the T-shaped curve when the axis is in StandStill state.



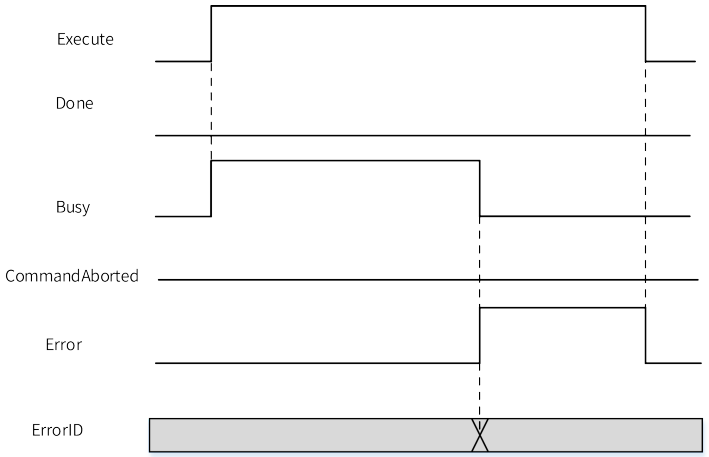
- Another relative positioning instruction is triggered during relative positioning.



- Relative positioning of the axis is aborted by the MC_Stop instruction.



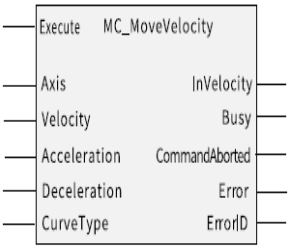
- The drive fails during motion of the axis.



5.13.1.14 MC_MoveVelocity

MC_MoveVelocity -- Velocity control

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveVelocity	Velocity control		MC_MoveVelocity(Execute := ???, Axis := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , InVelocity => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_MoveVelocity: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
S2	Velocity	Target velocity	No	-	Positive number, negative number, or 0, absolute value less than the maximum velocity	REAL
S3	Acceleration	Acceleration rate	No	-	Positive number, less than the maximum acceleration rate	REAL
S4	Deceleration	Deceleration rate	Yes	Acceleration rate	Positive number, less than the maximum acceleration rate	REAL
S5	CurveType	Velocity curve type 0: T-shaped velocity curve	Yes	0	0 to 1	INT

		1: 5-segment S-shaped curve				
D1	InVelocity	Velocity reached	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	CmdAborted	Abortion of execution	Yes	OFF	ON or OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

The MC_MoveVelocity instruction implements the velocity control function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis ID, modification on Axis is invalid when Enable is on the rising edge.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the Execute input, the instruction latches the input parameters on the left, such as Velocity and Acceleration, triggers the axis to run at the velocity specified by Velocity, and switches the PLCopen state machine of the axis to the ContinuousMotion state.

  CurveType specifies the type of the velocity curve.
  - If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration.
  - If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration rate and minimum deceleration rate of the axis during acceleration and deceleration.

During execution of this instruction, you can call MC_Stop, MC_Halt, or MC_ImmediateStop (supported by the drive) to stop the motion of the axis.

Abortion

When this instruction is active, the axis is in ContinuousMotion state in PLCopen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCopen state machine switching. CommandAborted is active when this instruction is aborted.

When Enable is ON and the Busy signal is active, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

When Enable is ON and the Busy signal is active, if deceleration to stop needs to be performed upon a limit signal, CommandAborted is active.

Errors

Error 9101 is reported when the axis ID does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

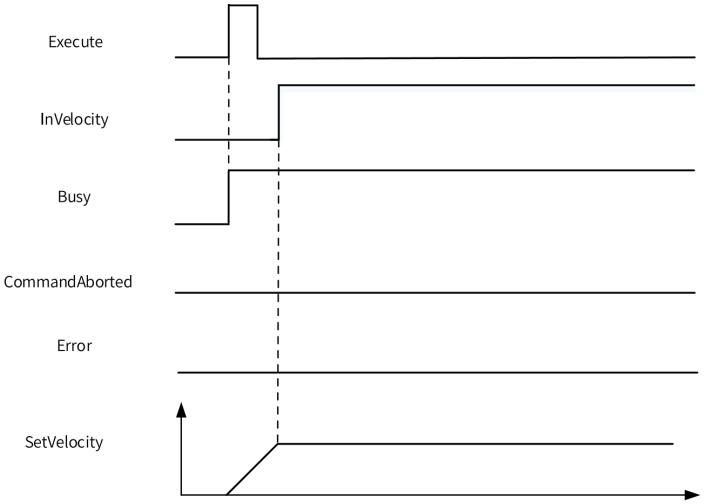
Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, or ContinuousMotion.

- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.

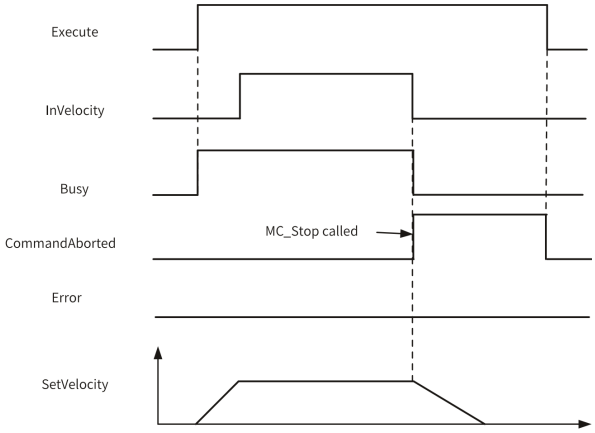
- An error is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

- The MC_MoveVelocity instruction is executed to implement continuous motion based on the T-shaped curve when the axis is in StandStill state.

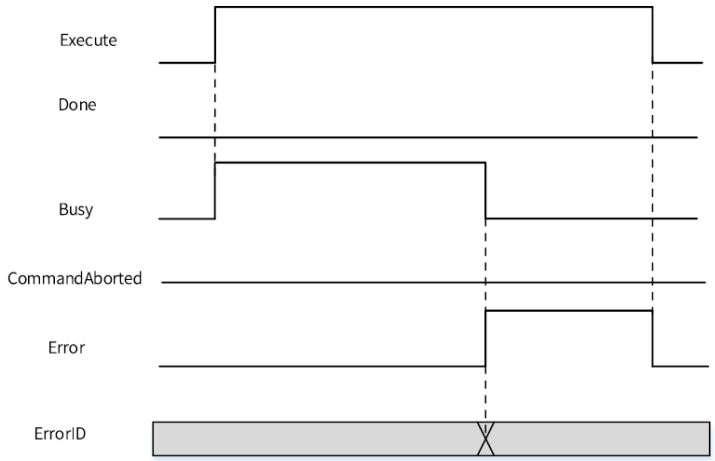


- Axis motion is aborted by the MC_Stop instruction.
  - Axis motion is aborted by the MC_Stop instruction.



  - The drive fails during acceleration of the axis.

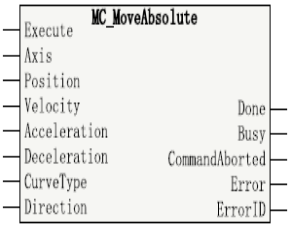
- The drive fails during acceleration of the axis.



5.13.1.15 MC_MoveAbsolute

MC_MoveAbsolute: Absolute positioning

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveAbsolute	Absolute positioning		MC_MoveAbsolute(Execute := ???, Axis := ???, Position := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , Direction := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	MC_MoveAbsolute: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT

						_sMCAXIS_INFO
S2	Position	Target position	No	-	Positive number, negative number, or 0	REAL
S3	Velocity	Target velocity	No	-	Positive number, less than the maximum velocity	REAL
S4	Acceleration	Acceleration	No	-	Positive number, less than the maximum acceleration	REAL
S5	Deceleration	Deceleration	Yes	Acceleration	Positive number, less than the maximum acceleration	REAL
S6	CurveType	Velocity curve type 0: T-shaped velocity curve 1: 5-segment S-curve	Yes	0	0 to 1	INT
S7	Direction	Direction (applicable to only the ring mode) 0: Forward (velocity > 0) 1: Reverse (velocity < 0) 2: Minimum distance 3: Current direction 4: Current direction (overtravel positioning)	Yes	0	0 to 4	INT
D1	Done	Target position reach	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CmdAborted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see 5.13.1.37 Axis Fault Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	-	√	-
S6	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

NOTE

[1] The X element is not supported.

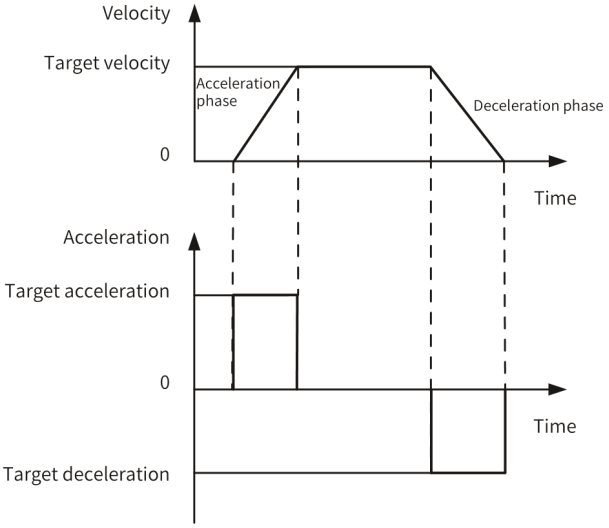
Function and Instruction Description

This instruction implements absolute positioning of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis number, modification on Axis is invalid when Execute is ON/ OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

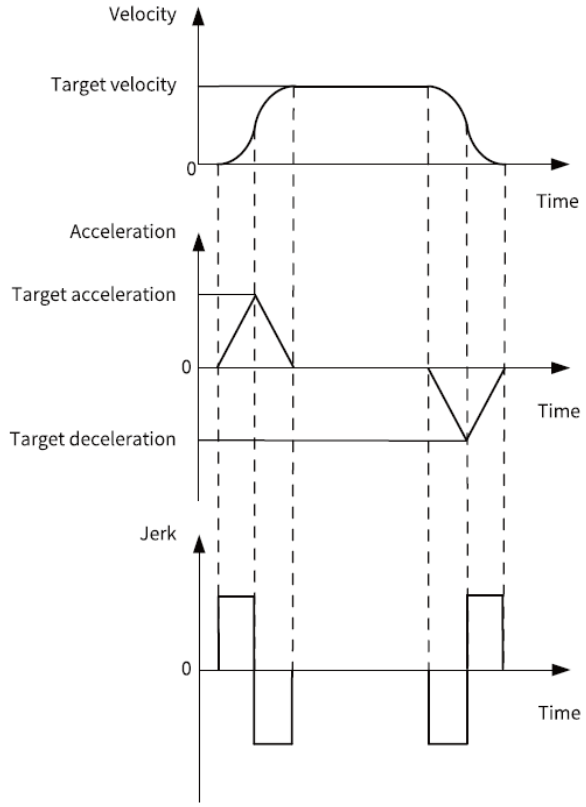
  On the rising edge of the Execute input, the instruction latches the input parameters on the left, such as Position and Velocity, triggers the absolute positioning function, and switches the PLCOpen state machine of the axis to the DiscreteMotion state.
  - In linear mode, Position specifies the target position for absolution positioning. If the current position is less than the target position, the axis moves forward to reach the position specified by Position. If the current position is greater than the target position, the axis moves in the reverse direction to reach the position specified by Position.

- CurveType specifies the type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration.



For details about the T-shaped curve, see the relative positioning instruction section.

- If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.

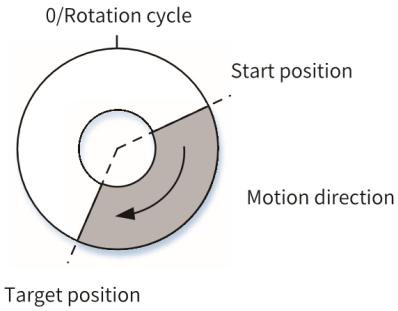


For details about the S-curve, see the relative positioning instruction section.

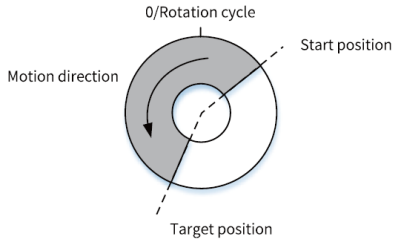
In ring mode, the instruction first uses Position to mod the revolution cycle to obtain the absolute position Position_p in a revolution cycle. The actual direction of the axis is determined based on the following four conditions:

1. Direction = 0 (Forward, target velocity > 0). If the current velocity is greater than 0, the axis continues to run in the current direction and stops at the position specified by Position_p; if the current velocity is less than 0, the axis decelerates to 0 and then starts to move at reserve velocity until it reaches the position

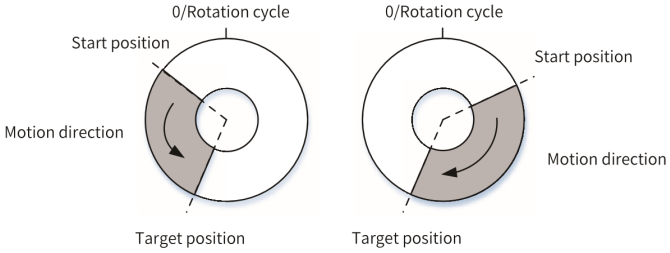
specified by Position_p; if the current position is just the position specified by Position_p*1, the axis does not move.



2. Direction = 1 (Reverse, target velocity < 0). If the current velocity is less than 0, the axis continues to run in the current direction and stops at the position specified by Position_p; if the current velocity is greater than 0, the axis decelerates to 0 and then starts to move at reserve velocity until it reaches the position specified by Position_p; if the current position is just the position specified by Position_p*1, the axis does not move.



3. Direction = 2 (Minimum distance). The current position of the axis is recorded on the rising edge of the Execute signal. Assume that the current velocity is 0. Distance indicates the distance that the axis moves forward from 0 velocity to the position specified by Position_p. If Distance is less than or equal to 0.5*revolution cycle, the axis moves forward; if it is greater than 0.5*revolution cycle, the axis moves in the reverse direction; if the current position is just the position specified by Position_p*1, the axis does not move.



4. Direction = 3 (Current direction). On the rising edge of the Execute signal, the axis moves in the direction same as that before the rising edge of the Execute signal until it reaches the position specified by Position_p. If the machine is powered on for the first time, the axis moves in the forward direction (target velocity > 0). If the current position is just the position specified by Position_p*1, the axis does not move.
5. Direction = 4 (Current direction, overtravel positioning). On the rising edge of the Execute signal, the axis moves in the direction same as that before the rising edge of the Execute signal until it reaches the specified position. If the displacement distance within a single rotation cycle does not meet the deceleration stop planning requirements specified by the instruction, the overtravel positioning will be completed. The number of overtravel cycles is calculated based on the initial velocity for positioning trigger and set deceleration.

> **NOTE**
>
> *1: In ring mode, if the target position is greater than the ring cycle, the instruction uses the target position to mod the ring period to obtain a new target position. If the absolute value of the difference between the new target position and the set position of the axis is less than 0.001, the two values are considered equal.

Abortion

When this instruction is active, the axis is in DiscreteMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Enable is ON and the Busy signal is active, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

When Enable is ON and the Busy signal is active, if deceleration to stop needs to be performed upon a limit signal, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

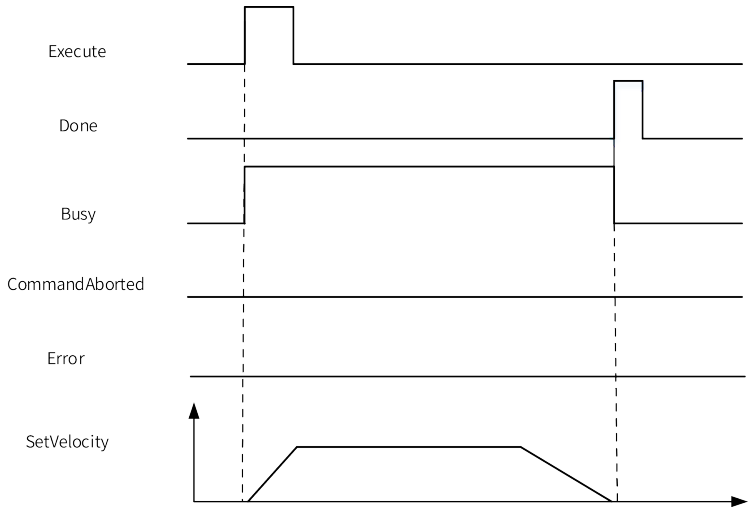
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

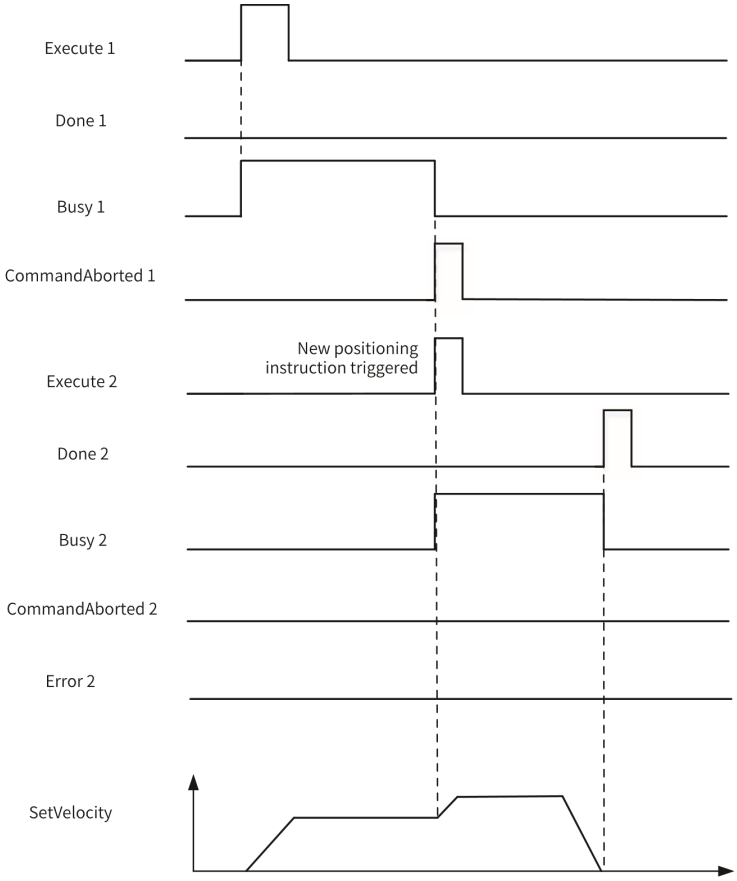
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- When Execute is ON and the Done signal is inactive, if the axis enters the commissioning state, Error 9116 is reported.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Sequence Diagram

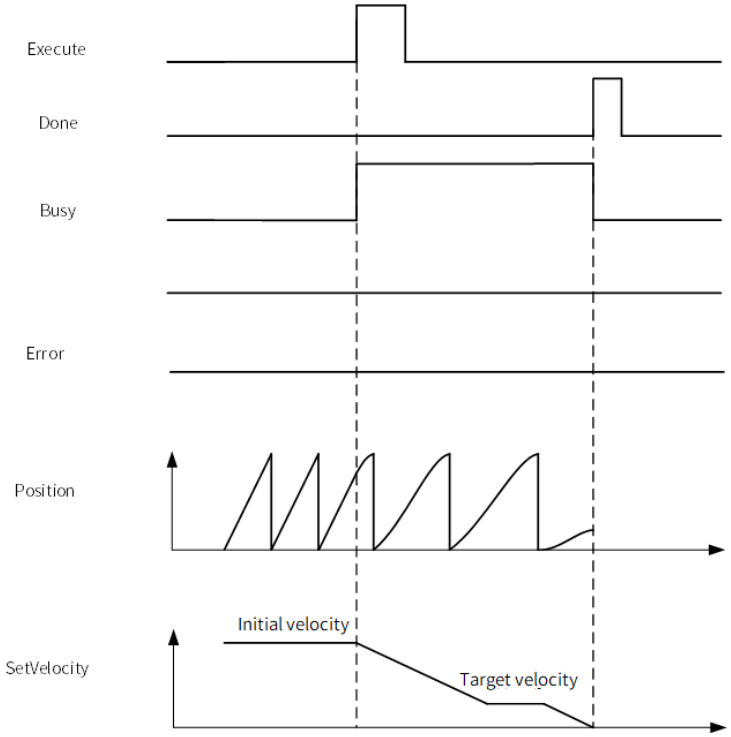
- The MC_MoveAbsolute instruction is executed to implement absolute positioning based on the T-shaped curve when the axis is in StandStill state.



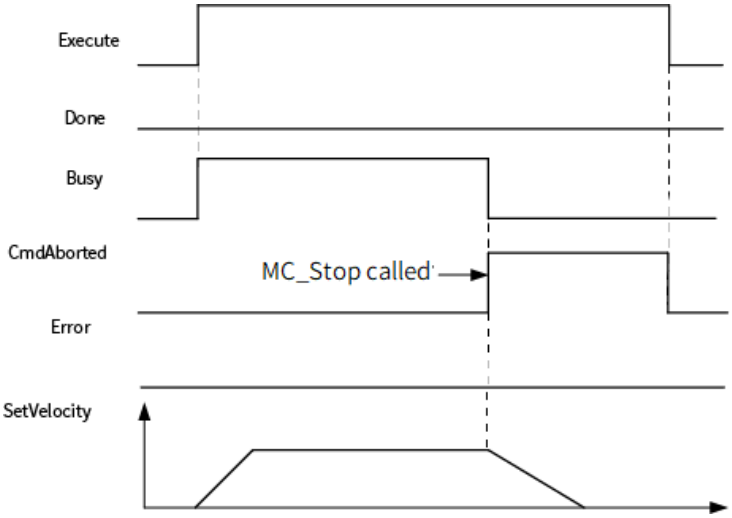
- Another absolute positioning instruction is triggered during absolute positioning.



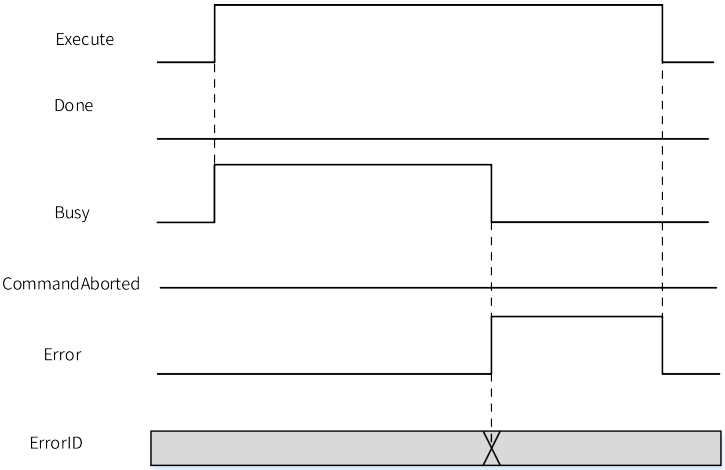
- The positioning instruction is triggered when the axis is in overtravel positioning mode, and the distance required for deceleration is greater than the set position distance.



- Absolute positioning of the axis is aborted by MC_Stop instruction.



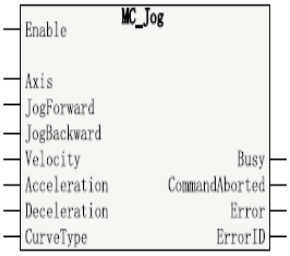
- The drive fails during motion of the axis.



5.13.1.16 MC_Jog

MC_Jog: Jogging

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_Jog	Jog		MC_Jog(Enable := ???, Axis := ???, JogForward := ???, JogBackward := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , Busy => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	MC_Jog: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	JogForward	Jogging in forward direction, triggered on the rising edge	No	-	-	BOOL
S3	JogBackward	Jogging in reverse direction, triggered on the rising edge	No	-	-	BOOL
S4	Velocity	Target speed	No	-	Positive number, less than the maximum velocity, less than the maximum jogging velocity	REAL
S5		Acceleration	No	-		REAL

	Acceleration				Positive number, less than the maximum acceleration	
S6	Deceleration	Deceleration	Yes	Acceleration	Positive number, less than the maximum acceleration	REAL
S7	CurveType	Curve shape 0: T-shaped velocity curve 1: 5-segment S-curve	Yes	0	0-1	INT
D1	Busy	Busy flag	Yes	OFF	-	BOOL
D2	CommandAborted	Abortion of execution	Yes	OFF	-	BOOL
D3	Error	Error flag	Yes	OFF	-	BOOL
D4	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see 5.13.1.37 Axis Fault Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	√	√	-	-	-	-	-	-
S3	√	√	√	-	-	-	-	-	-
S4	-	-	-	√	√	-	-	√	-
S5	-	-	-	√	√	-	-	√	-
S6	-	-	-	√	√	-	-	√	-
S7	-	-	-	√	√	-	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-

NOTE

[1] The X element is not supported.

Function

This instruction implements the jogging function of the EtherCAT bus axis or the local pulse axis. It is active high.

- Specifying axis
  - Axis is latched on the rising edge of the Enable input.
  - If Axis specifies the axis number, modification on Axis is invalid when Enable is ON/OFF.
- Description:
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the instruction, the function block latches the input parameters, such as Velocity, Acceleration, Deceleration, and CurveType, and switches the state machine of the axis to the ContinuousMotion state to start jogging.
  - When Enable is ON, if instructions such as MC_Stop and MC_MoveRelative are called, the MC_Jog will be aborted, and the CommandAborted output of MC_Jog instruction becomes active.
  - When JogForward is active, the axis moves forward at the velocity specified by Velocity; when JogBackward is active, the axis moves in reverse direction at the velocity specified by Velocity. When both JogForward and JogBackward are active, the axis stops but does not enter the ErrorStop state, and the instruction reports a fault.
  - When Enable is ON, if the axis reaches the limit when moving toward one direction, the instruction reports a fault, and the axis stops but does not enter the ErrorStop state. When the MC_Jog instruction is triggered again, the axis will move toward the opposite direction.
  - CurveType specifies the type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration. If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.

Abortion

When this instruction is active, the axis is in ContinuousMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Enable is ON and the Busy signal is active, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

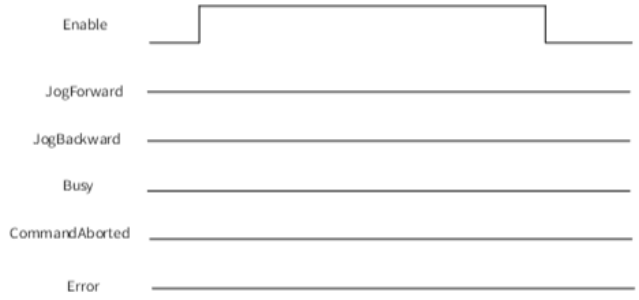
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

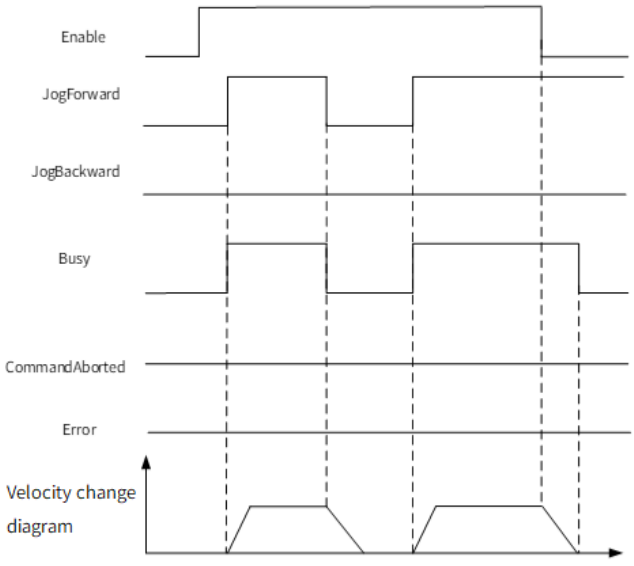
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Enable input.
- Error 9106 is reported if the axis is decelerating upon a fault on the rising edge of the Enable input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Enable input.
- Error 9116 is reported if the axis is in commissioning state when Enable is ON and Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Enable is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Sequence Diagram

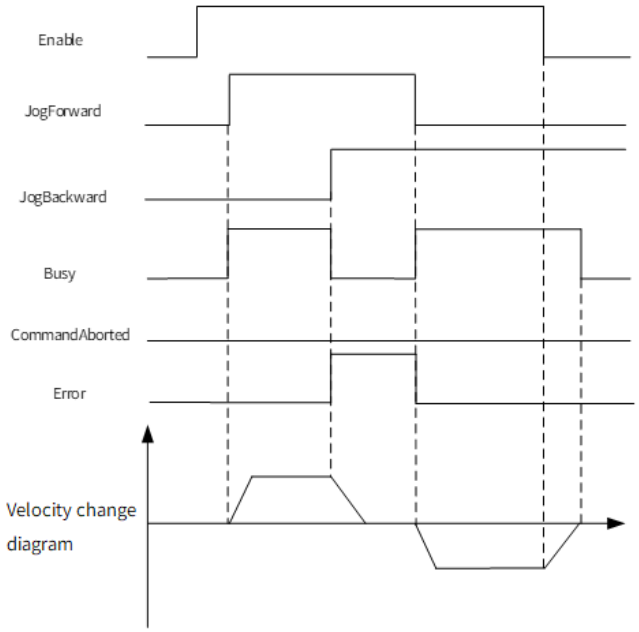
- Only the Enable input is active.



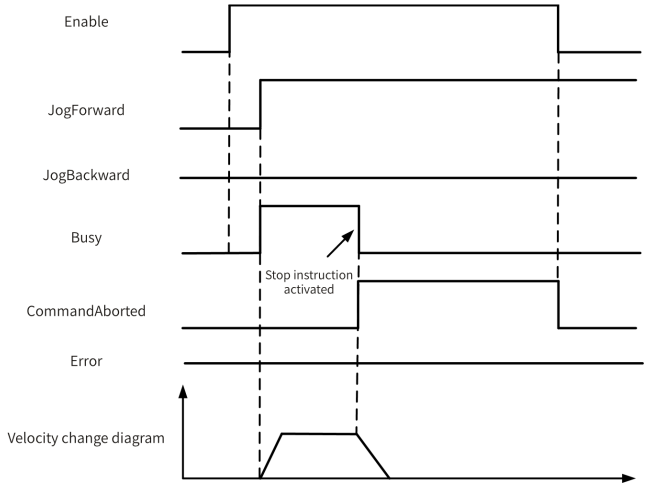
- The Enable and JogForward inputs are active.



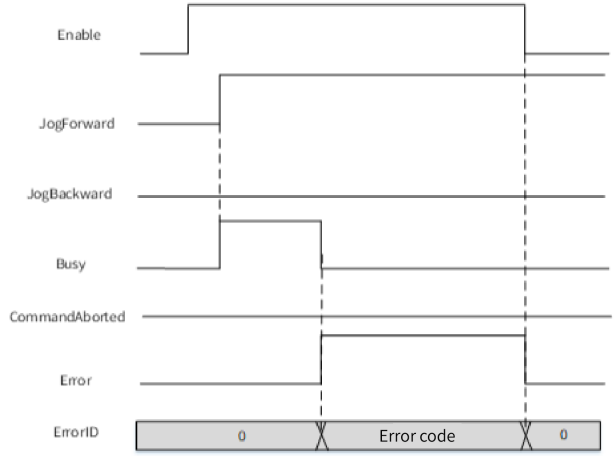
- The Enable and JogForward inputs are active, and JogBackward is set to TRUE.



- Instruction execution is aborted by the MC_Stop instruction.



- The axis reports an error.



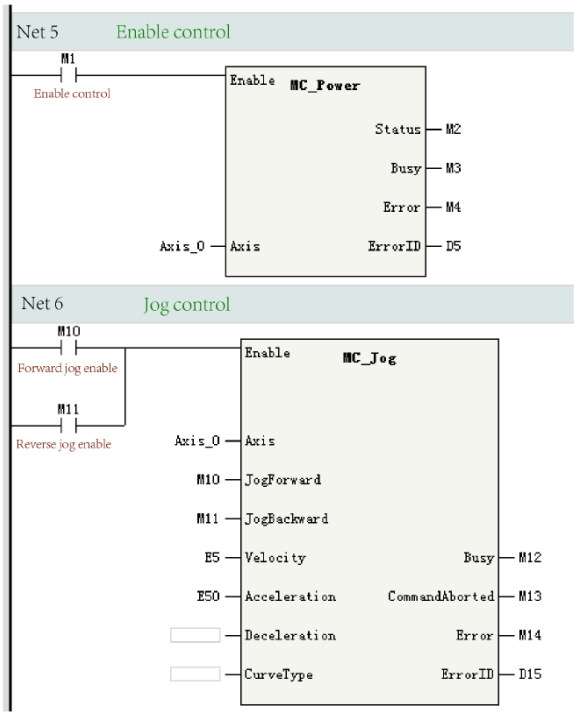
Example

The following are some examples.

1. After M1 is set to ON, Axis_0 is enabled.
2. After M10 is set to ON, Axis_0 runs in forward direction at 5 unit/s.
3. After M10 is set to OFF, Axis_0 stops running.

4. After M11 is set to ON, Axis_0 runs in reverse direction at 5 unit/s.

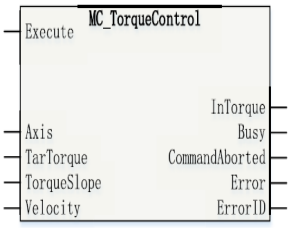
5. After M11 is set to OFF, Axis_0 stops running.



5.13.1.17 MC_TorqueControl

MC_TorqueControl -- Torque control

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_TorqueControl	Torque control		MC_TorqueControl(Execute := ???, Axis := ???, TarTorque := ???, TorqueSlope := ???, Velocity := , InTorque => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-
32-bit Instruction	MC_TorqueControl: Continuous execution

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
S2	TarTorque	Target torque, in 1%	No	-	Positive number, negative number, or 0	REAL
S3	TorqueSlope	Torque slope, in %/s This operand specifies the acceleration rate, that is, gain per second, of a torque instruction in torque mode.	No	-	Positive number	REAL
S4	Velocity	Velocity limit	No	-	Positive number or 0	REAL
D1	InTorque	Torque reached The output is active when the set torque reaches the target torque and the absolute value of the difference between the feedback torque and the target torque is less than 5%.	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON or OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

This instruction is used to implement the torque control function only for the EtherCAT bus axis. It is active on the rising edge and does not support the virtual axis mode.

- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis ID, modification on Axis is invalid when Execute is ON or OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  The torque instruction can be used only when the following PDOs are configured: 0x6040, 0x6041, 0x6060, 0x6061, 0x6071, and 0x6077. Otherwise, an error is reported.

  This instruction adopts the synchronous torque mode of the drive to implement the torque control function.

  The function block latches the input parameters TarTorque, TorqueSlope, and Velocity on the rising edge of the instruction. The axis enters the ContinuousMotion state and performs torque motion.

  - TarTorque: Target torque, in 1%. Only one decimal place after the decimal point is valid in the program, and the subsequent ones are directly discarded. The actual torque of the drive is limited by the maximum positive and negative torque specified in the configuration parameters.

  - TorqueSlope: Torque slope, in %/s. Only one decimal place after the decimal point is valid in the program, and the subsequent ones are directly discarded.

- Velocity control in torque mode
  For servo drives of Inovance, if 0x607F is mapped, this instruction limits the maximum velocity of the servo motor through 0x607F. If 0x607F is not mapped, the velocity limit is invalid.

  On the rising edge of Execute, the instruction converts the velocity limit specified by Velocity into pulse unit and writes it into 0x607F through PDO.

  If the torque instruction is aborted by another instruction, the maximum velocity of the axis can be limited by specifying **Max. Velocity** on the configuration interface.

  For third-party drives, Velocity can be used as the velocity limit only when the following conditions are met:

  - The maximum velocity of the servo motor can be limited by using 0x607F.
  - 0x607F can be configured in the PDO.
  - The unit of 0x607F is a pulse unit, not a rotation velocity unit.

- Stop control in torque mode
  In torque mode, the MC_Stop instruction can be executed to stop the drive. Upon receiving the MC_Stop instruction, the drive switches to the synchronous position mode and decelerates according to the deceleration rate specified in the MC_Stop instruction.

Abortion

When this instruction is active, the axis is in ContinuousMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Execute is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.

When Execute is ON and the Done signal is inactive, if the axis is disabled, CommandAborted is active.

Errors

Error 9101 is reported when the axis ID does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

Error 9113 is reported when the MC_TorqueControl instruction is called after the virtual axis mode is enabled.

A PDO configuration fault is reported when the required PDO is not configured.

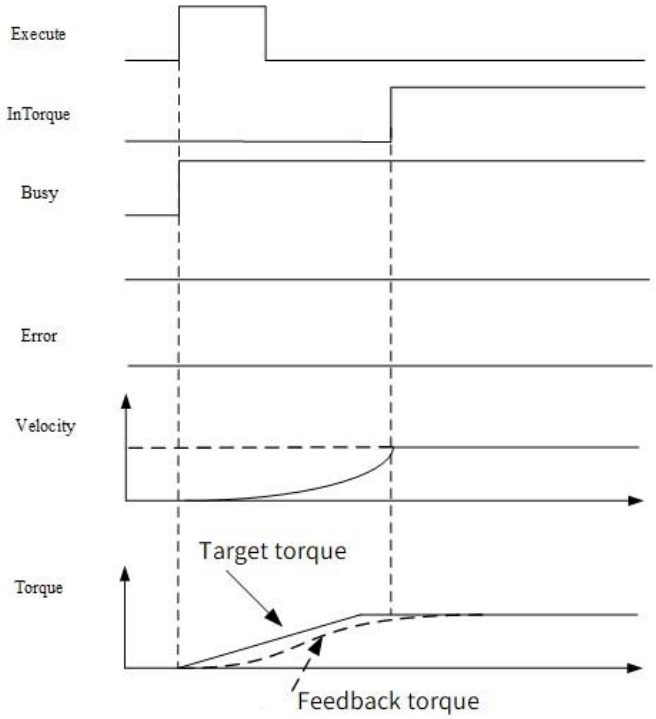
Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, or ContinuousMotion.

- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- An error is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.

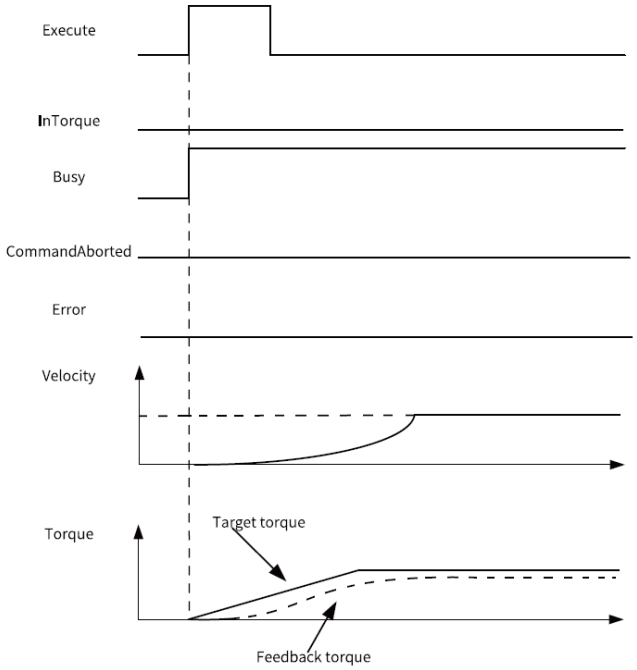
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

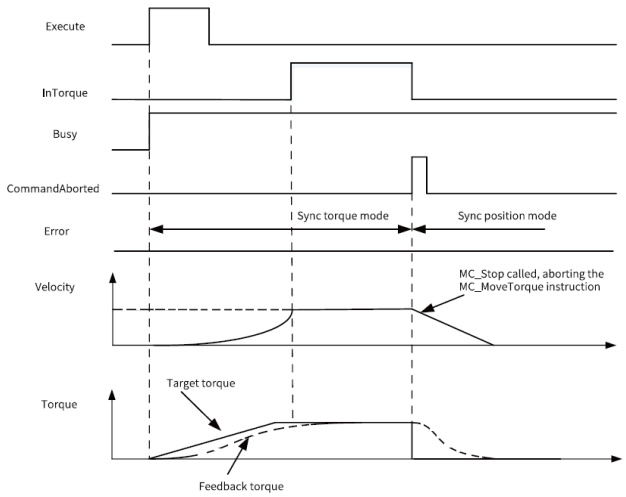
- The instruction is triggered after the target torque is specified, and the actual output torque can reach the target torque.



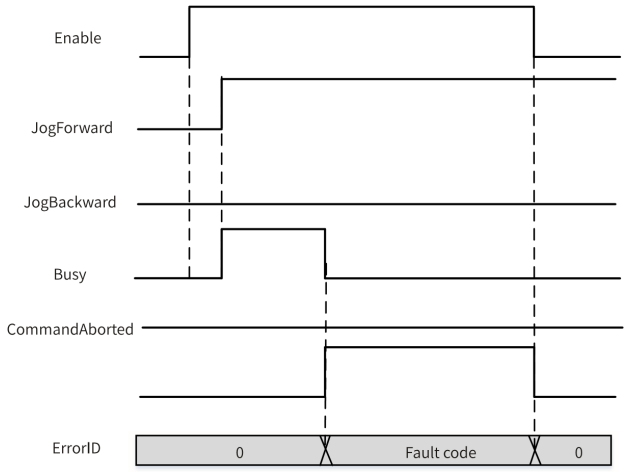
- The instruction is triggered after the target torque is specified, and the actual output torque cannot reach the target torque.



- The MC_Stop instruction aborts the instruction execution during torque operation.



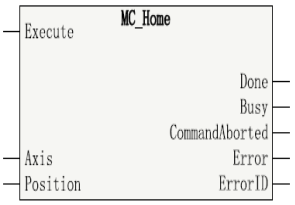
- The drive reports an error during torque operation.



5.13.1.18 MC_Home

MC_Home -- Homing

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_Home	Homing		MC_Home(Execute := ???, Axis := ???, Position := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-
32-bit Instruction	MC_Home: Continuous execution

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
S2	Position	Home offset	Yes	0	Positive number, negative number, or 0	REAL
D1	Done	Homing completed	Yes	OFF	OFF or ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF or ON	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	OFF or ON	BOOL
D4	Error	Error flag	Yes	OFF	OFF or ON	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

The MC_Home instruction implements the homing function of the EtherCAT bus axis and the local pulse axis. It is active on the rising edge.

- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis ID, modification on Axis is invalid when Execute is ON or OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  The function block latches the input parameter Position on the rising edge of the instruction. The axis enters the Homing state and performs homing.

  Position specifies the home offset.

  When this instruction is called in virtual axis mode, absolute homing is performed according to method 35 in CiA 402. The CiA 402 standard defines the protocol for motor drive and motion control, and method 35 is the absolute homing method in this standard.
- Multi-execution
  The MC_Home instruction does not support multi-execution. After an MC_Home instruction is called to change the axis to Homing state, if another MC_Home instruction is called, the instruction called later reports an error.

> **NOTE**
>
> The MC_Home instruction does not support the synchronous motion mode. When the master axis is homing, the position type of the synchronization instruction is set to the instruction position, and the slave axis does not perform synchronous motion.

Abortion

When this instruction is active, the axis is in Homing state in PLCopen. This instruction can be aborted by MC_Stop and MC_ImmediateStop, which can make the axis enter Stopping state. CommandAborted is active when this instruction is aborted.

If the MC_Power instruction is inactive when Enable is ON and the Done signal is inactive, the axis is disabled and CommandAborted is active.

Errors

Error 9101 is reported when the axis ID does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

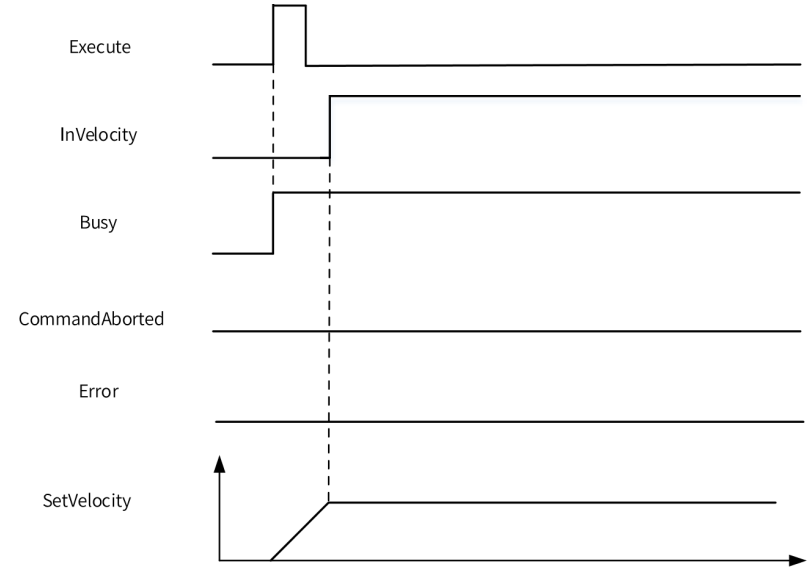
Error 9108 is reported if this instruction is executed when the axis is not in StandStill state.

- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- An error is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.

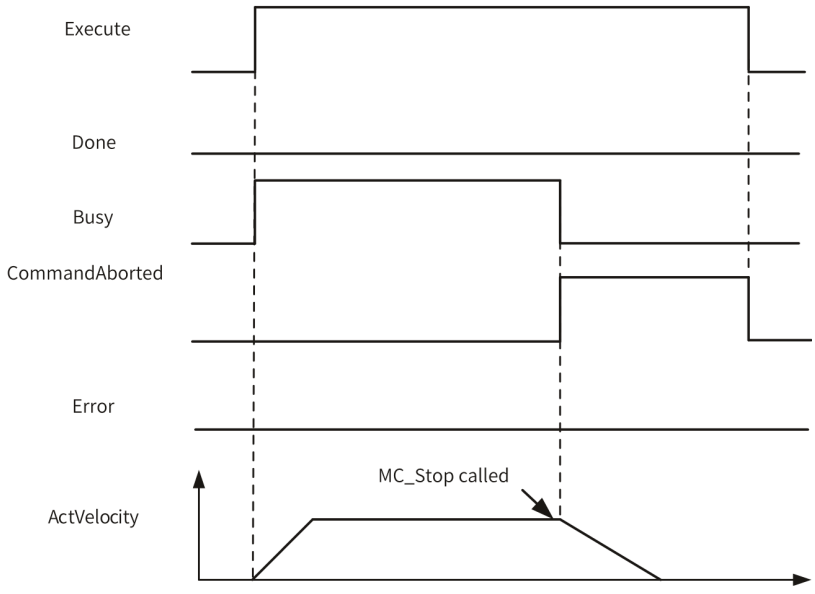
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

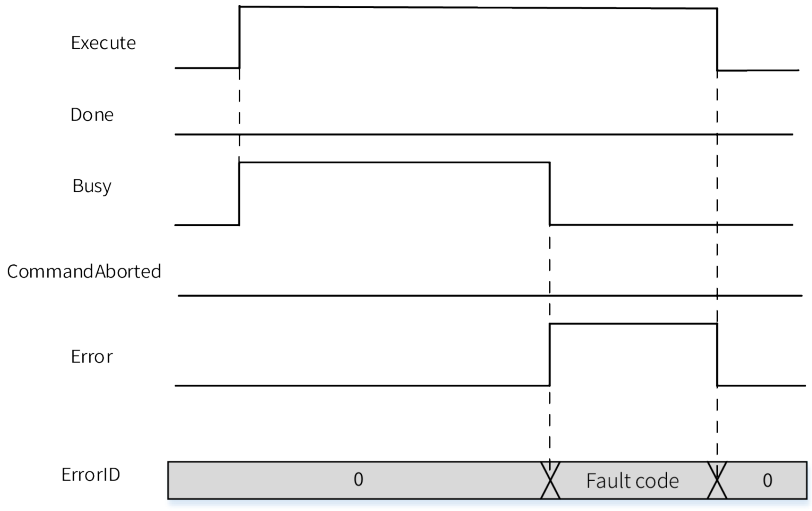
- The drive performs homing properly.



- Homing is aborted by the MC_Stop instruction.



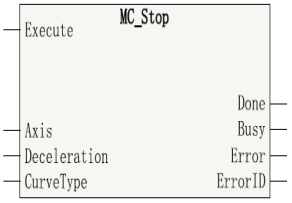
- The drive fails during the homing process.



5.13.1.19 MC_Stop

MC_Stop -- Stop

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_Stop	Stop		MC_Stop(Execute := ???, Axis := ???, Deceleration := ???, CurveType := , Done => , Busy => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_Stop: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
S2	Deceleration	Deceleration rate	No	-	Positive number, less than the maximum acceleration rate	REAL
S3	Curve Type	Curve type 0: T-shaped velocity curve 1: 5-segment S-curve	Yes	0	0 to 1	INT
D1	Done	Stop completed	Yes	OFF	OFF or ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF or ON	BOOL
D3	Error	Error flag	Yes	OFF	OFF or ON	BOOL
D4	ErrorID	Error code	Yes	0	*1	INT

> **NOTE**
>
> *1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	-	-	√	-
S3	-	-	-	√	√	-	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-

> **NOTE**
>
> [1]: The X element is not supported.

Function and Instruction Description

The MC_Stop instruction implements the stop function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis ID, modification on Axis is invalid when Execute is ON or OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  The function block latches the input parameters such as Deceleration and CurveType on the rising edge of the Execute input. The axis enters the Stopping state and performs deceleration.

  After deceleration is completed, the Done signal becomes active, and the axis remains in the Stopping state when Execute is TRUE.

  When Execute is FALSE and Done is TRUE, the axis switches from the Stopping state to the StandStill state.

  The stop mode varies according to the running state of the axis when this instruction is executed.

  1. If the axis is executing the positioning instruction or running continuously when this instruction is called, CurveType specifies the type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis decelerates based on the value of Deceleration. If CurveType is set to 1, the

5-segment S-curve is used. In this case, Deceleration specifies the maximum deceleration rate of the axis during deceleration.

2. When the axis is in Homing state, this instruction triggers the Halt flag of the control word of the drive, and the drive decelerates according to the preset parameters. In this case, CurveType and Deceleration are invalid.
3. If you call this instruction when the axis is in profile control mode, CurveType is invalid, T-shaped curve is planned for stopping the axis, and the axis decelerates based on the value of Deceleration.

Re-execution

The same stop instruction can be executed repeatedly. If the same stop instruction is re-triggered during deceleration, the drive decelerates to stop according to the deceleration specified by the instruction triggered last.

Multi-execution

The MC_Stop instruction does not support multi-execution. If an MC_Stop instruction is called while another MC_Stop instruction is still active, an error is reported.

Abortion

When this instruction is active, the axis is in Stopping state, and the instruction cannot be aborted by other motion instructions. When this instruction becomes inactive, the axis switches from the Stopping state to the StandStill state, and other motion control instructions can run.

This instruction can be aborted by the MC_ImmediateStop instruction. If the MC_ImmediateStop instruction is called while MC_Stop is active, the MC_Stop instruction reports an error.

Errors

Error 9101 is reported when the axis ID does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

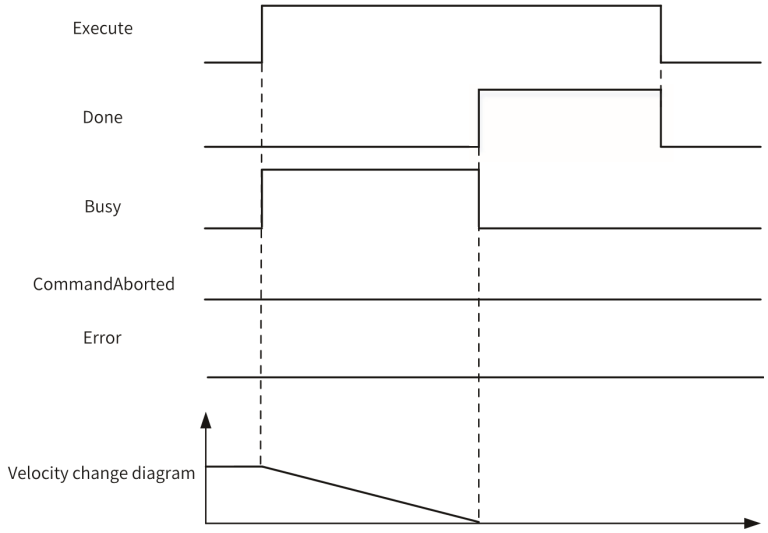
Error 9108 is reported if this instruction is executed when the axis is in Disabled or ErrorStop state.

Error 9142 is reported if this instruction is executed after the MC_ImmediateStop instruction is executed to put the axis in Stopping state.

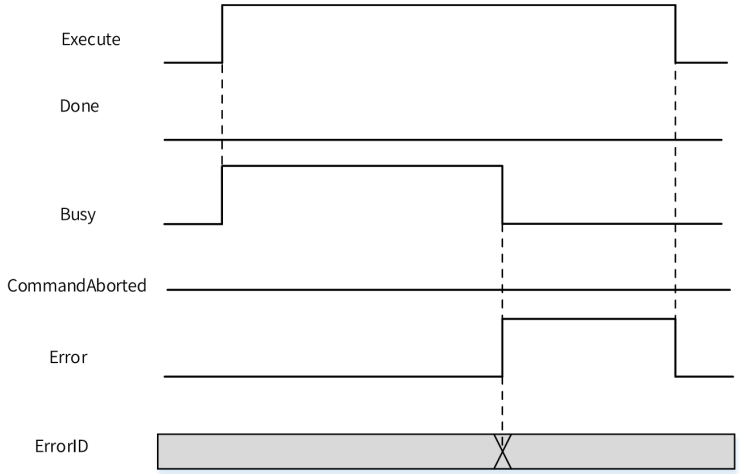
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- An error is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

- The MC_Stop instruction is executed after the MC_MoveVelocity instruction.



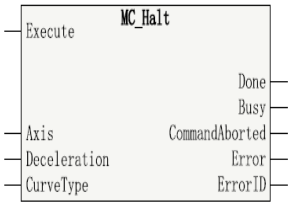
- The drive fails during instruction execution.



5.13.1.20 MC_Halt

MC_Halt: Halt

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_Halt	Halt (Not recoverable)		MC_Halt(Execute := ???, Axis := ???, Deceleration := ???, CurveType := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-

32-bit instruction	MC_Halt: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Name	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	Deceleration	Deceleration	No	-	Positive number, less than the maximum acceleration	REAL
S3	CurveType	Curve Shape 0: T-shaped velocity curve 1: 5-segment S-curve acceleration and deceleration	Yes	0	0 to 1	INT
D1	Done	Stop completed	Yes	OFF	OFF/ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF/ON	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	OFF/ON	BOOL
D4	Error	Error flag	Yes	OFF	OFF ON	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see 5.13.1.37 Axis Fault Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

NOTE

[1] The X element is not supported.

Function

This instruction implements the halt function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis number, modification on Axis is invalid when Execute is ON/ OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the instruction, the function block latches input parameters such as Deceleration and CurveType, and the axis performs deceleration. This instruction can be aborted by other instructions.

  The halt mode varies according to the running state of the axis when this instruction is executed.

  - If the axis is executing the positioning instruction of the cyclic synchronous mode or running continuously when this instruction is called, CurveType specifies the type of the velocity curve.
    1. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis decelerates based on the value of Deceleration.
    2. If CurveType is set to 1, the 5-segment S-curve is used. In this case, Deceleration indicates the minimum deceleration of the axis during deceleration.
  - If the instruction is executed when the axis is in profile control mode, CurveType is invalid. The axis performs a controlled stop using T-shaped curve. In this case, the axis decelerates based on the value of Deceleration.

Abortion

When this instruction is active, the axis is in DiscreteMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion or ContinuousMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

- When Execute is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.
- When Execute is ON and the Done signal is inactive, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in Disabled, ErrorStop, or Homing state.

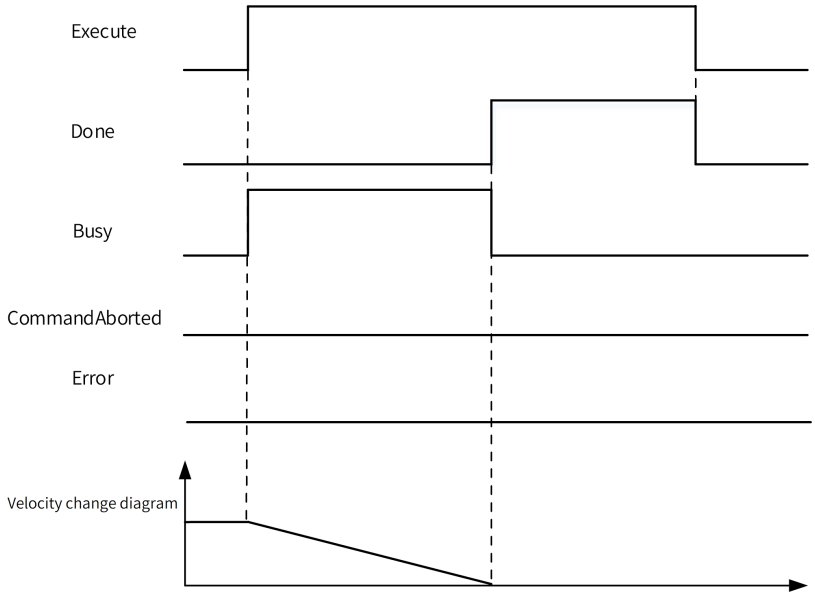
Error 9115 is reported if this instruction is executed after the MC_Stop instruction is executed to put the axis in Stopping state.

Error 9142 is reported if this instruction is executed after the MC_ImmediateStop instruction is executed to put the axis in Stopping state.

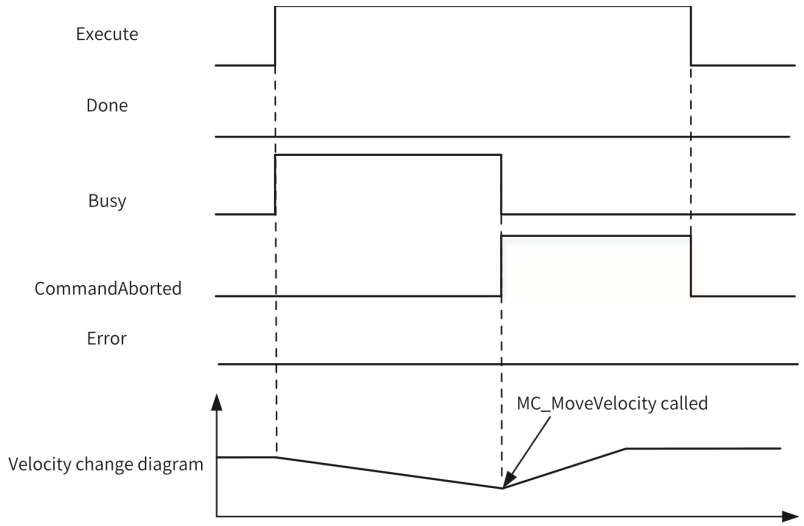
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Sequence Diagram

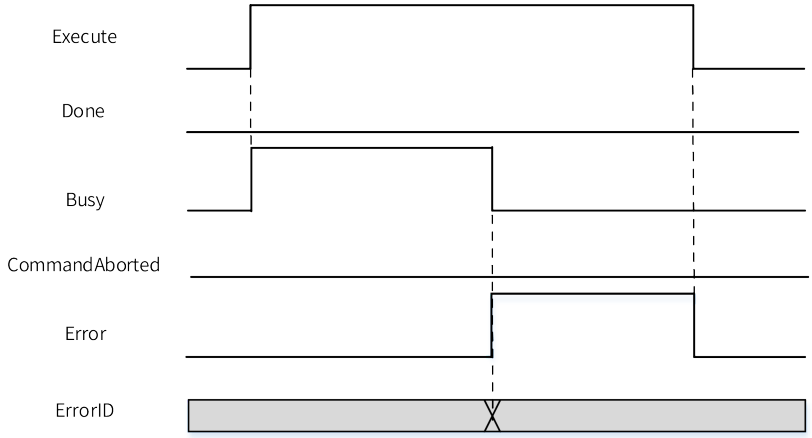
- After the positioning instruction is called, the MC_Halt instruction is triggered.



- After the MC_Halt instruction is triggered, the velocity instruction is called to abort the execution of the MC_Halt instruction.



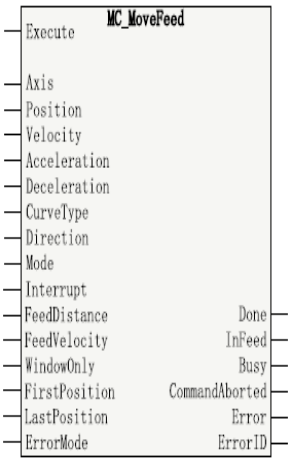
- The drive stops upon a fault during execution of the MC_Halt instruction.



5.13.1.21 MC_MoveFeed

MC_MoveFeed: Interrupt positioning

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveFeed	Positioning interruption		MC_MoveFeed(Execute := ???, Axis := ???, Position := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , Direction := , Mode := , Interrupt := , FeedDistance := ???, FeedVelocity := , WindowOnly := , FirstPosition := , LastPosition := ,

Instruction	Name	LD Expression	LiteST Expression
			ErrorMode := , Done => , InFeed => , Busy => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	MC_MoveFeed: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	Position	Target position	No	-	Positive number, negative number, or 0	REAL
S3	Velocity	Target velocity	No	-	0, positive number, less than the maximum velocity	REAL
S4	Acceleration	Acceleration	No	-	0, positive number, less than the maximum acceleration	REAL
S5	Deceleration	Deceleration	Yes	Same as acceleration	0, positive number, less than the minimum acceleration	REAL
S6	CurveType	Velocity curve type 0: T-shaped acceleration 1: 5-segment S-curve acceleration	Yes	0	0 to 3	INT

		2: T-shape acceleration, no speed change mode 3: 5-segment S-shape acceleration, no speed change mode Others: T-shaped velocity curve				
S7	Direction	Motion direction of absolute positioning in ring mode 0: Forward (Target velocity > 0) 1: Reverse (Target velocity < 0) 2: Minimum distance 3: Current direction	Yes	0	0 to 3	INT
S8	Mode	Mode 0: Absolute positioning mode 1: Relative positioning mode 2: Speed control mode	Yes	0	0 to 2	INT
S9	Interrupt	Interrupt source 0: Probe 1 1: Probe 2	Yes	0	0 to 1	INT
S10	FeedDistance	Travel distance after the interrupt feed input Positive: Feed in the same direction as the axis was moving before the interrupt input for the distance specified by FeedDistance. Negative: Feed in the opposite direction as the axis was moving before the interrupt input for the distance specified by FeedDistance.	No	-	Positive number, negative number, or 0	REAL
S11	FeedVelocity	Target velocity after the interrupt feed input	Yes	Same as Velocity	Positive number, less than the maximum velocity	REAL

S12	WindowOnly	Interrupt source window enable 0: Disabled 1: Enabled	Yes	OFF	OFF/ON	BOOL
S13	FirstPosition	Start position of the interrupt source window	Yes	0	Positive number, negative number, or 0	REAL
S14	LastPosition	End position of the interrupt source window	Yes	0	> FirstPosition	REAL
S15	ErrorMode	Fault mode OFF: After the position specified by Position is reached, if no interrupt signal is detected, the Done signal is set to ON, and the instruction does not report a fault. ON: After the position specified by Position is reached, if no interrupt signal is detected, the Error signal is set to ON, and the instruction reports a fault.	Yes	OFF	OFF/ON	BOOL
D1	Done	The target position is reached.	Yes	OFF	-	BOOL
D2	InFeed	Interrupt signal active	Yes	OFF	-	BOOL
D3	Busy	Busy flag	Yes	OFF		BOOL
D4	CmdAborted	Abortion of execution	Yes	OFF	-	BOOL
D5	Error	Error flag	Yes	OFF	-	BOOL
D6	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see 5.13.1.37 Axis Fault Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	-	-	√	-
S3	-	-	-	√	√	-	-	√	-
S4	-	-	-	√	√	-	-	√	-
S5	-	-	-	√	√	-	-	√	-
S6	-	-	-	√	√	-	√	-	-
S7	-	-	-	√	√	-	√	-	-
S8	-	-	-	√	√	-	√	-	-
S9	-	-	-	√	√	-	√	-	-
S10	-	-	-	√	√	-	-	√	-
S11	-	-	-	√	√	-	-	√	-
S12	√	√	√	-	-	-	-	-	-
S13	-	-	-	√	√	-	√	-	-
S14	-	-	-	√	√	-	√	-	-
S15	√	√	√	-	-	-	-	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	√[1]	√	√	-	-	-	-	-	-
D6	-	-	-	√	√	-	-	-	-

NOTE

[1] The X element is not supported.

Function and Instruction Description

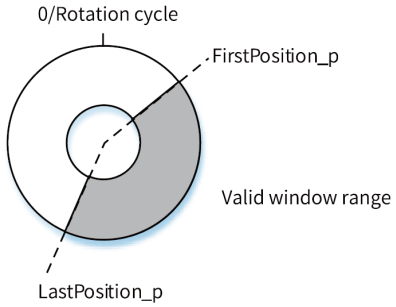
This instruction implements the interrupt positioning function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge, and it does not support the imaginary axis mode.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis number, modification on Axis is invalid when Execute is ON/ OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

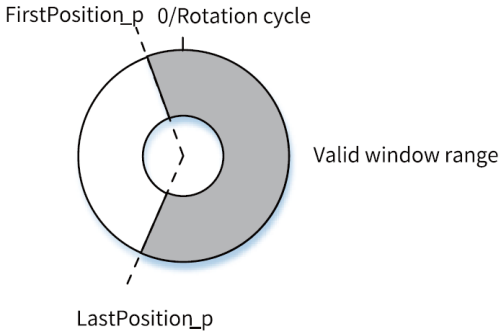
  On the rising edge of the instruction, the function block latches input parameters such as Position, Velocity, Direction, Number, Acceleration, and Deceleration.

  Before the interrupt arrives, the axis performs absolute positioning (Mode = 0), relative positioning (Mode = 1), or continuous motion (Mode = 2) based on parameters including Position, Velocity, Direction, and Mode. After an interrupt signal is generated by the interrupt source specified by Interrupt, the axis performs relative movement at the position where the interrupt arrives based on FeedDistance and FeedVelocity.
  - Position: Target position of the axis before the interrupt arrives when Mode is set to 0 (absolute positioning) or 1 (relative positioning).
  - Velocity: Target velocity of the axis before the interrupt arrives.
  - CurveType: Type of the velocity curve. If CurveType is set to 0 or 2, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration. If CurveType is set to 1 or 3, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.
    When CurveType is set to 2 or 3, if the Acceleration/Deceleration is set to 0, there is no acceleration/deceleration changes during acceleration/deceleration. If the set position after interrupt response exceeds the interrupt target position, the rotation stops.
  - Direction: Direction of rotation, which is the same as Direction of the MC_MoveAbsolute instruction. It specifies the axis rotation direction in ring mode when Mode is set to 0 (absolute positioning). The value 0 indicates forward, 1 indicates reverse, 2 indicates the shortest path, and 3 indicates the current direction.
  - Mode: Movement mode before the interrupt arrives. When Mode is set to 0, the axis performs absolute positioning before the interrupt feed input; when Mode is set to 1, the axis performs relative positioning before the interrupt feed input; when Mode is set to 2, the axis performs continuous motion before the interrupt feed input.
  - Interrupt: Interrupt source. When it is set to 0, the interrupt source is probe 1, and the interrupt is active on the rising edge of probe 1. When it is set to 1, the interrupt source is probe 2, and the interrupt is active on the rising edge of probe 2.
  - FeedDistance: Target travel distance after the interrupt feed input. If the value is positive, the axis runs in the current direction for the distance specified by FeedDistance when the interrupt signal arrives. If the value is negative, the axis runs in the opposite direction for the distance specified by FeedDistance when the interrupt signal arrives.
  - FeedVelocity: Target velocity after the interrupt feed input.

- When WindowOnly is OFF, the window detection function is disabled. The axis position can be latched as long as the probe input signal is active.
- When WindowOnly is ON, the window detection function is enabled.
- In linear mode, the instruction detects the probe signal only when the current position of the axis falls within the range specified by FirstPosition and LastPosition.
  In ring mode, the valid window range is as follows:
  - When FirstPosition is less than LastPosition, the valid window range is as follows:



  - When FirstPosition is greater than LastPosition, the valid window range is as follows:



  - When FirstPosition equals to LastPosition, the valid window range is as follows:

- ErrorMode: Fault handling mode when there is no interrupt. In absolute or relative positioning mode, when no interrupt signal is detected after the position (travel distance) specified by Position is reached, the instruction reports a fault if ErrorMode is set to ON and does not report a fault if ErrorMode is set to OFF.
- InFeed: The InFeed output becomes active after the interrupt signal arrives.

Abortion

When this instruction is active, the axis is in DiscreteMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Execute is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.

When Execute is ON and the Done signal is inactive, if the axis is disabled, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

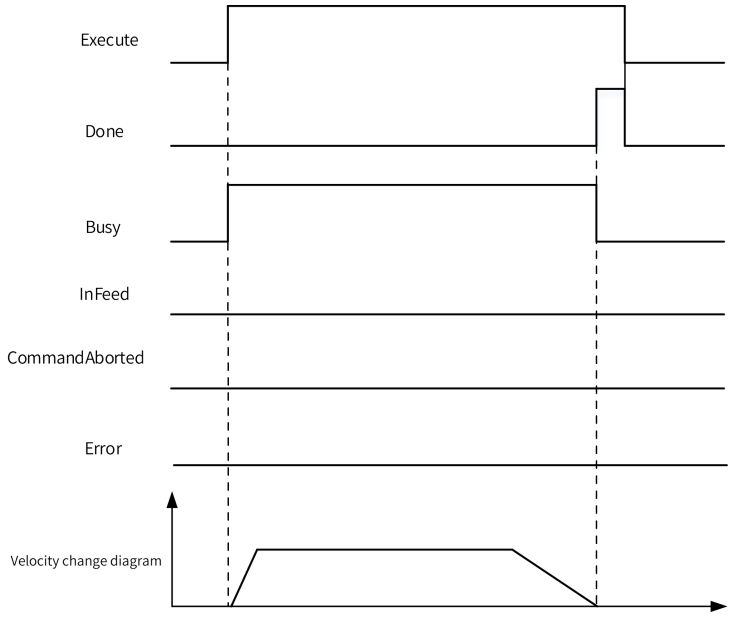
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

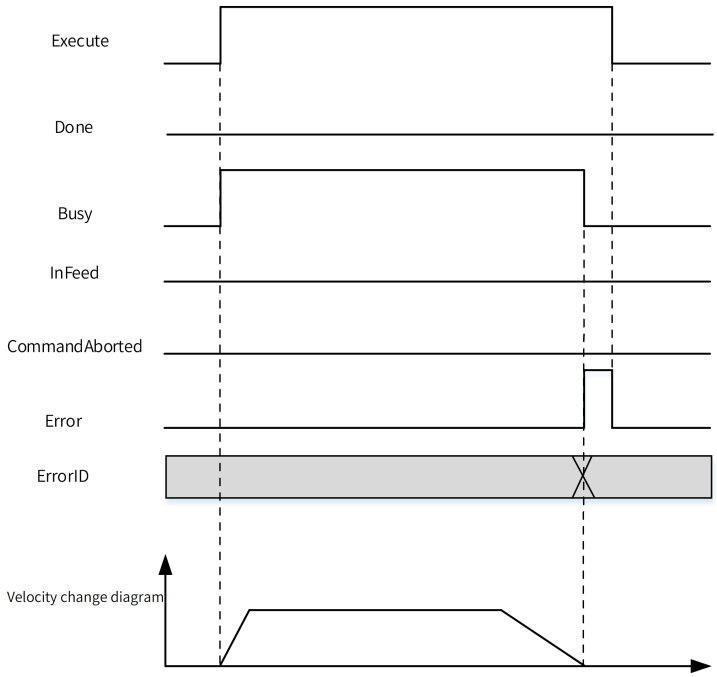
- Error 9116 is reported when the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9133 is reported when the imaginary axis mode is enabled on the rising edge of the Execute input.
- Error 9106 is reported when the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- When Execute is ON and the Done signal is inactive, if the axis enters the commissioning state, Error 9116 is reported.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Sequence Diagram

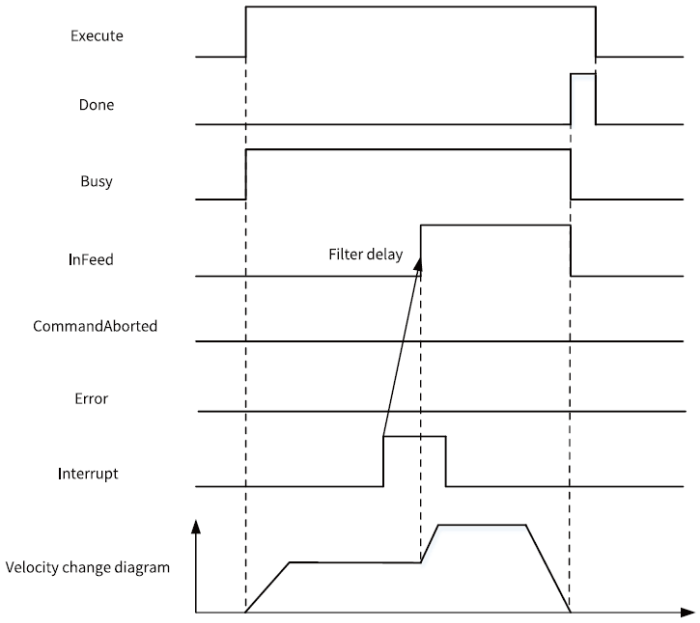
- The relative or absolute positioning mode is selected, the interrupt signal is not triggered, and ErrorMode is set to OFF.



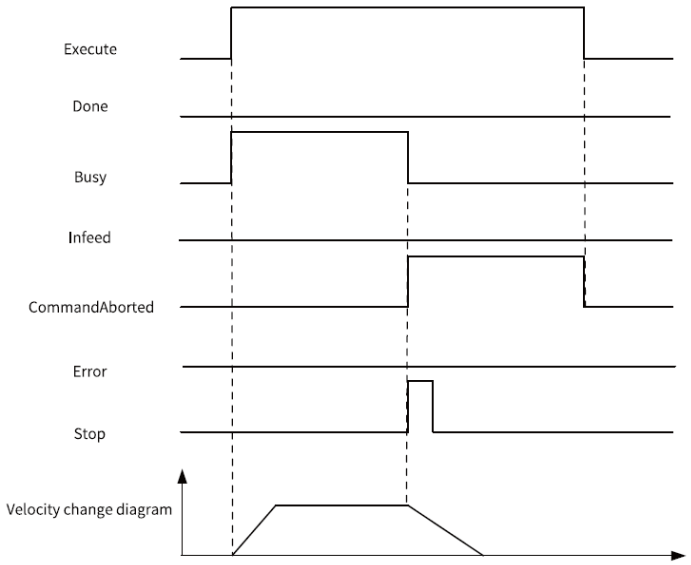
- The relative or absolute positioning mode is selected, the interrupt signal is not triggered, and ErrorMode is set to ON.



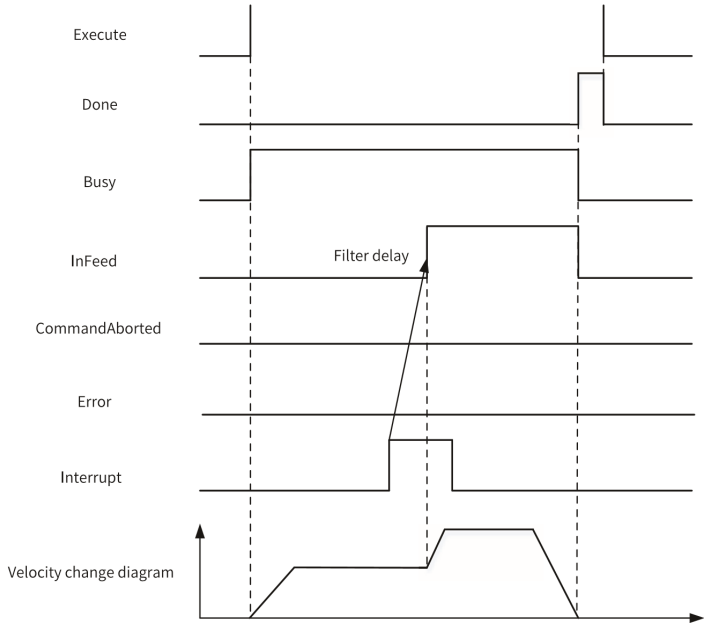
- The relative or absolute positioning mode is selected and the interrupt signal is triggered.



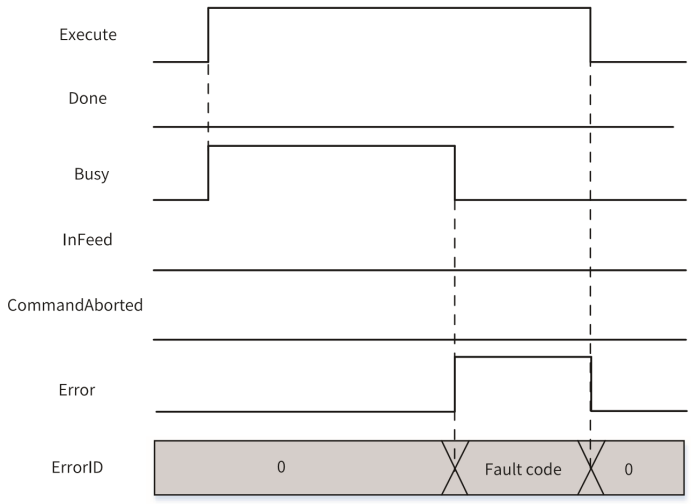
- The velocity mode is selected, the interrupt is not triggered, and the MC_Stop instruction is called to abort this instruction after it is executed for a period of time.



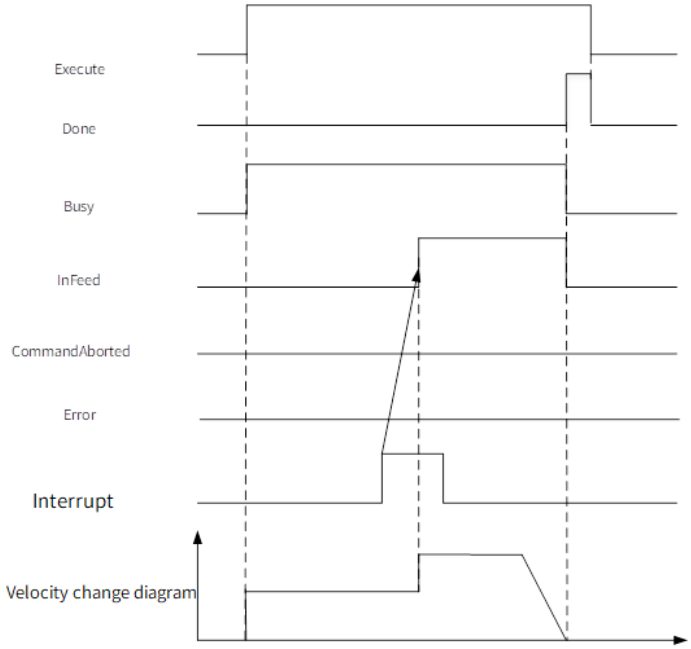
- The velocity mode is selected and the interrupt is triggered.



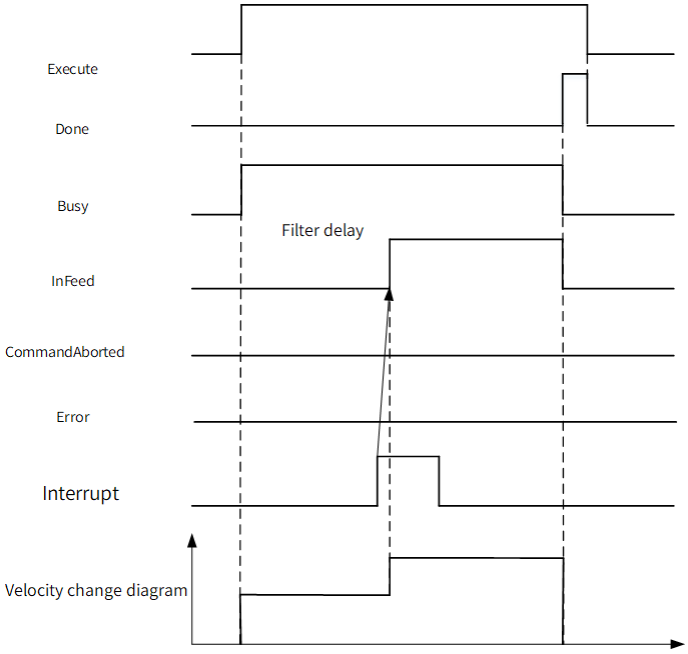
- A fault occurs during instruction execution.



- When CurveType is set to 2 and Acceleration is set to 0:



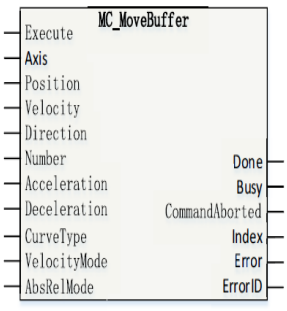
- When CurveType is set to 2 and Acceleration and Deceleration are both set to 0:



5.13.1.22 MC_MoveBuffer

MC_MoveBuffer: Multi-position positioning

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveBuffer	Multi-position		MC_MoveBuffer(Execute := ???, Axis := ???, Position := ???, Velocity := ???, Direction := , Number := ???, Acceleration := ???,

Instruction	Name	LD Expression	LiteST Expression
			Deceleration := , CurveType := , VelocityMode := , AbsRelMode := , Done => , Busy => , CommandAborted => , Index => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	MC_MoveBuffer: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	Position	Start address of the target position	No	-	Positive number Negative number 0	REAL
S3	Velocity	Start address of the target velocity	No	-	Positive number	REAL
S4	Direction	Start address of absolute positioning direction in ring mode 0: Forward (Target velocity > 0) 1: Reverse (Target velocity < 0) 2: Minimum distance 3: Current direction	Yes	0	0 to 3	INT, array*16
S5	Number	Number of buffer pairs	No	-	1 to 16	INT

S6	Acceleration	Acceleration	No	-	-	FLT32
S7	Deceleration	Deceleration	Yes	Acceleration	-	FLT32
S8	CurveType	Velocity curve type 0: T-curve acceleration 1: 5-segment S-curve acceleration Others: T-shaped velocity curve	Yes	0	-	INT
S9	Velocity Mode	Velocity switching mode 0: Decelerate to 0 and start the next segment 1: Keep the current velocity and start the next segment	Yes	0	0 to 1	INT
S10	AbsRelMode	Positioning mode 0: Absolute positioning 1: Relative positioning	Yes	0	0 to 1	INT
D1	Done	Stop completed	Yes	OFF	-	BOOL
D2	Busy	Busy flag	Yes	OFF	-	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	-	BOOL
D4	Index	Current segment	Yes	0	0 to 15	INT
D5	Error	Error flag	Yes	OFF	-	BOOL
D6	ErrorID	Fault code	Yes	0	*1	INT



*1: For details, see 5.13.1.37 Axis Fault Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	✓	√	√	-	-	-
S4	-	-	-	√	√	√	-	-	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	√	-	√	-
S7	-	-	-	√	√	√	-	√	-
S8	-	-	-	√	√	√	-	✓	-
S9	-	-	-	√	√	√	√	-	-
S10	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	√[1]	√	√			√	-	-	-
D6	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Function

This instruction implements the multi-position positioning function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis number, modification on Axis is invalid when Execute is ON/ OFF.
- Function description

This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

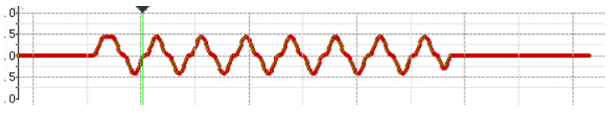
On the rising edge of the Execute input, the function block latches input parameters such as Position, Velocity, Direction, Number, Acceleration, and Deceleration.

The axis performs absolute positioning (AbsRelMode = 0) or relative positioning (AbsRelMode = 1) in buffer mode based on the value of AbsRelMode. This instruction supports up to 16-segment positions.

- Position: Target position, array type, up to 16 segments. It specifies the target absolute position of the axis in absolute positioning mode or target travel distance of the axis in relative positioning mode.
- Velocity: Target velocity, array type, up to 16 segments.
- Direction: Target direction of absolute positioning in ring mode, same as Direction of the MC_MoveAbsolute instruction.
- Number: Number of groups of target position, target velocity, and direction to be buffered. It ranges from 1 to 16. A fault is reported if the value is out of range.
- CurveType: Type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration. If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.
- VelocityMode: Velocity switching mode. When it is set to 0, the axis decelerates to 0 before reaching a target position and then starts to run from 0 velocity to the next target position; when it is set to 1, the axis runs to a target position at the current target velocity and then switches to a new velocity based on the acceleration (deceleration) to move to the next target position.

NOTE

A special situation may arise during absolute positioning when the velocity is retained. Assume that 3 position segments are set. The target position of segment 1 is 10, that of segment 2 is 10.1, and that of segment 3 is 10.2. The target velocity is 100, and the feedback velocity is also 100. The EtherCAT task cycle is 8 ms, and the increment of the target travel distance per EtherCAT cycle is 0.8. During segment 1, if the current position is 9.9 when an EtherCAT cycle starts, the current position changes to 10.7 when the next EtherCAT cycle starts, which has exceeded the target position of segment 2. In this case, the axis needs to decelerate and run in the reverse direction, and this may produce the velocity curve as shown in the following figure:



Abortion

When this instruction is active, the axis is in DiscreteMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Execute is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.

When Execute is ON and the Done signal is inactive, if the axis is disabled, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

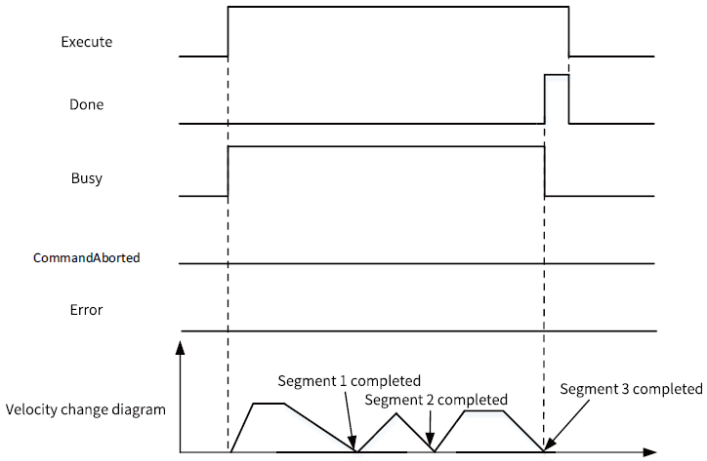
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

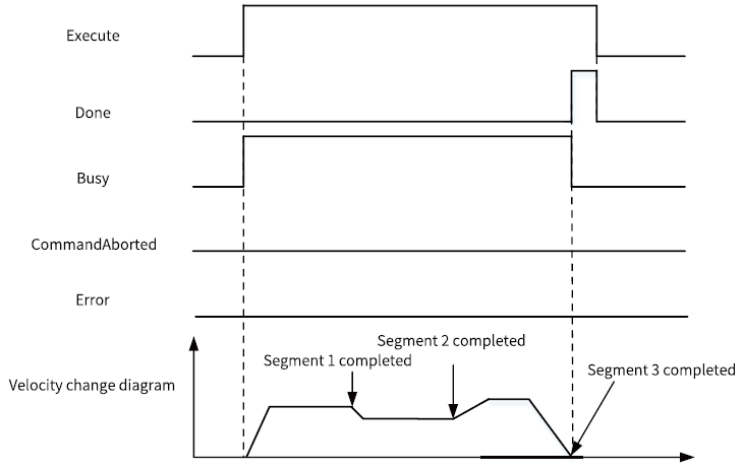
- Error 9116 is reported when the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported when the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- When Execute is ON and the Done signal is inactive, if the axis enters the commissioning state, Error 9116 is reported.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Sequence Diagram

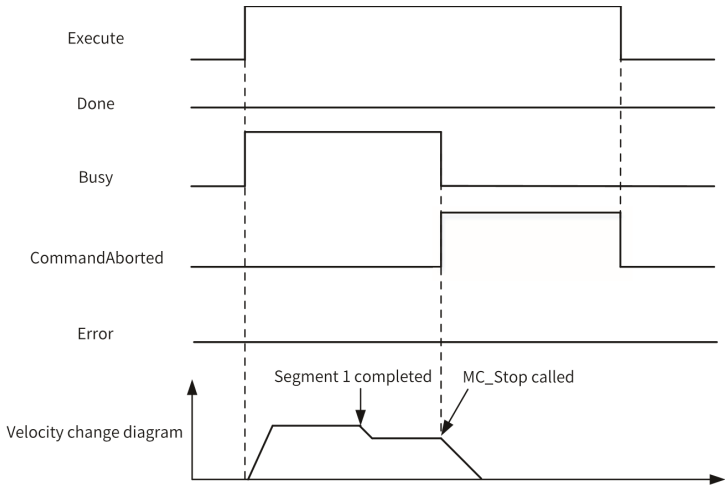
- In 3-segment buffer mode, VelocityMode is set to 0.



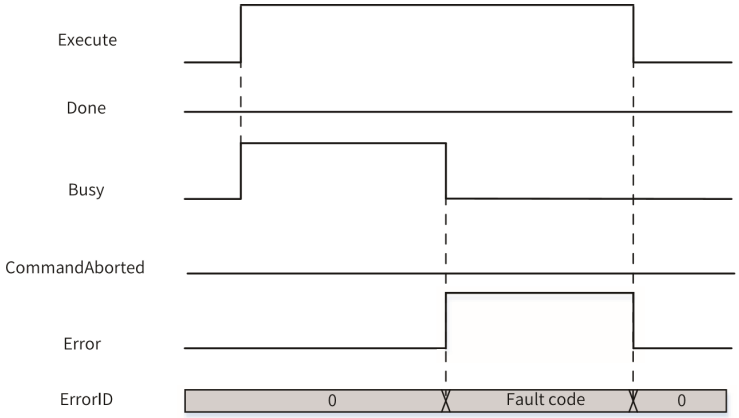
- In 3-segment buffer mode, VelocityMode is set to 1.



- In 3-segment buffer mode, instruction execution is aborted by the MC_Stop instruction.



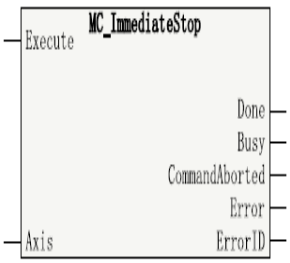
- In 3-segment buffer mode, an error occurs during instruction execution.



5.13.1.23 MC_ImmediateStop

MC_ImmediateStop -- Immediate stop

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_ImmediateStop	Emergency stop		MC_ImmediateStop(Execute := ???, Axis := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_ImmediateStop: Continuous execution					
Operand	Name	Description		Default	Range	Data Type

			Empty Allowed			
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
D1	Done	Stop completed	Yes	OFF	OFF or ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF or ON	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	OFF or ON	BOOL
D4	Error	Error flag	Yes	OFF	OFF or ON	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√		-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

The MC_ImmediateStop instruction implements the immediate stop function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis ID, modification on Axis is invalid when Execute is ON or OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the instruction, the function block switches the PLCopen state machine of the axis to the Stopping state, and switches the CANopen DS402 state machine of the drive to quick stop state. The drive stops running according to the stop mode specified by 0x605A. For the Inovance servo IS620N, the stop mode is described as follows:

  - When the servo is in CSP mode:

Setpoint	Stop Mode
0	Coast to stop, keeping de-energized status If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.
1	Stop at the emergency stop torque specified by 2007-10h, keeping the free-run state If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.
2	
3	
4	N/A
5	Stop at the emergency stop torque specified by 2007-10h, maintaining the locked position
6	
7	

  - When the servo is in HM mode:

Setpoint	Stop Mode
0	Coast to stop, keeping de-energized status If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.
1	Stop according to the ramp specified by 6084h (HM: 609Ah), keeping the free-run state If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.
2	Stop according to the ramp specified by 6085h, keeping the free-run state If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.

Setpoint	Stop Mode
3	Stop at the emergency stop torque, keeping the free-run state If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.
4	N/A
5	Stop according to the ramp specified by 6084h (HM: 609Ah), maintaining the locked position
6	Stop according to the ramp specified by 6085h, maintaining the locked position
7	Stop at the emergency stop torque specified by 2007-10h, maintaining the locked position

◦ When the servo is in CST mode:

Setpoint	Stop Mode
0	Coast to stop, keeping de-energized status If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.
1	Stop according to the ramp specified by 6087h, keeping the free-run state If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.
2	
3	Coast to stop, keeping de-energized status If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.
4	N/A
5	Stop according to the ramp specified by 6087h, maintaining the locked position
6	
7	Coast to stop, maintaining the locked position

- Re-execution
  The same MC_ImmediateStop instruction can be executed repeatedly. If the same MC_ImmediateStop instruction is re-triggered during deceleration, the drive stops according to the stop mode specified by the instruction triggered last.
- Multi-execution
  When multiple MC_ImmediateStop instructions are called, the instruction that triggers the rising edge first shall prevail. Other instructions report error 9143 (Immediate stop instruction called repeatedly).

When the axis is in the Stopping state, this instruction cannot be aborted by other motion instructions.

When the axis switches from the Stopping state to the StandStill state on the falling edge of the instruction flow, other motion control instructions can run.

This instruction takes priority over the MC_Stop instruction. If this instruction is called while the MC_Stop instruction is still active, the MC_Stop instruction reports an error.

Abortion

The axis disable interrupt signal output is active when Execute is ON.

Errors

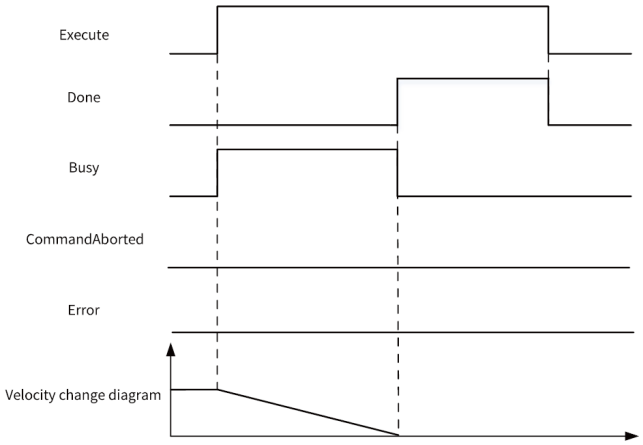
Error 9101 is reported when the axis ID does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

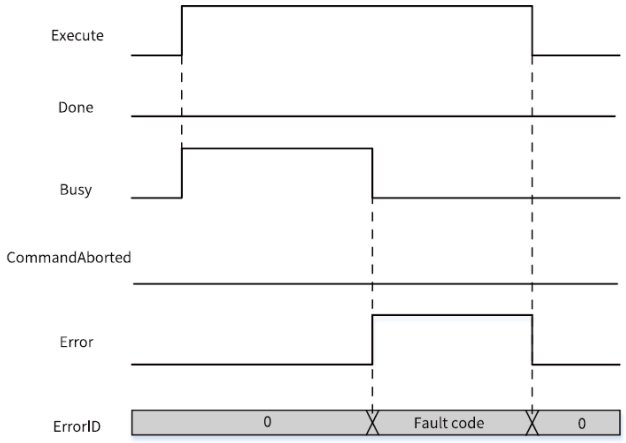
- Error 9108 is reported if the axis is in Disabled or ErrorStop state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

- This instruction is executed after the MC_MoveVelocity instruction.



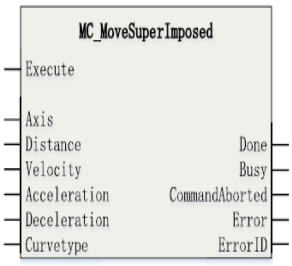
- The drive fails during instruction execution.



5.13.1.24 MC_MoveSuperImposed

MC_MoveSuperImposed -- Motion superimposition

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveSuperImposed	Motion superimposition		MC_MoveSuperImposed (Execute := ???, Axis := ???, Distance := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_MoveSuperImposed: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-		_sMCAXIS_INFO
S2	Distance	Phase compensation	No	-	Positive number 0 Negative number	REAL
S3	Velocity	Target velocity	No	-	Positive number	REAL
S4	Acceleration	Acceleration rate	No	-	Positive number	REAL
S5	Deceleration	Deceleration rate	Yes	Acc	Positive number	REAL
S6	Curvetype	Curve type 0: T-shaped velocity curve	Yes	0	0	INT
D1	Done		Yes	OFF	ON	BOOL

		Completion flag			OFF	
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	ON OFF	INT

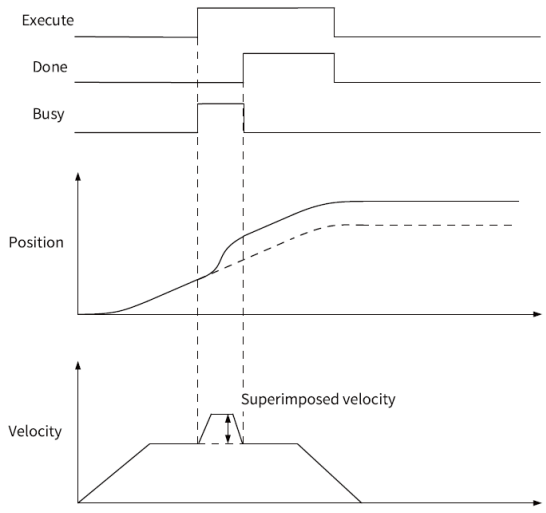
Function Description

On the rising edge of the Execute input, the MC_MoveSuperImposed instruction superimposes a phase positioning based on the original control mode of the axis according to the travel distance specified by Distance and the velocity specified by Velocity.

Instruction execution under different control modes

- Single-axis positioning instructions
  When called separately, this instruction controls the axis to perform relative positioning. The PLCopen state machine switches from the StandStill state to the DiscreteMotion state.

  When called during execution of an instruction that can make the servo axis work in CSP mode, this instruction implements motion superimposition.

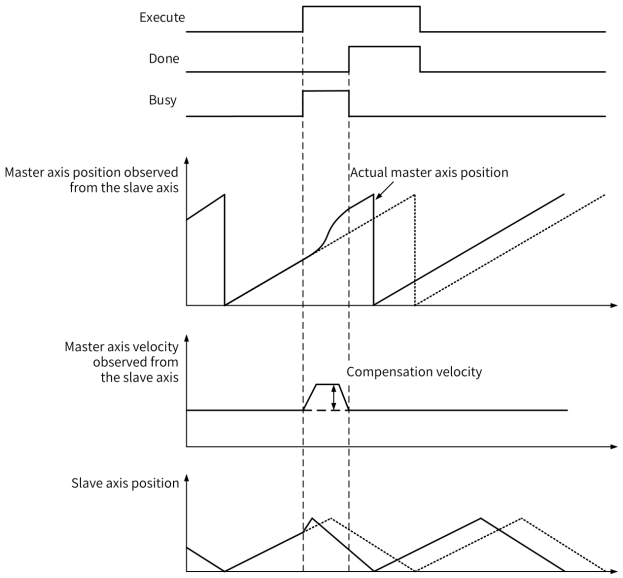


  If another instruction, such as MC_MoveAbsolute, is triggered during execution of this instruction, this instruction is aborted.

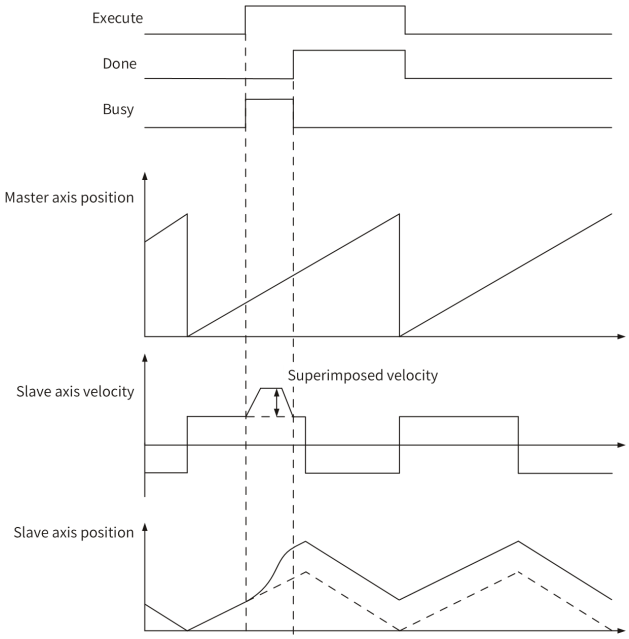
  This instruction can be aborted by MC_Stop, MC_Halt, and MC_ImmediateStop.

  The motion superimposition instruction cannot be executed while the MC_Halt instruction is active.

  If a non-CSP motion instruction such as MC_TorqueControl and MC_Home is executed during execution of this instruction, it reports an error.

- Axis group instructions
  If this instruction is called during execution of an axis group instruction, it reports an error and does not implement motion superimposition.
- Cam and gear instructions
  For details about operations on the master axis, see the rules described in the single-axis and axis group instructions.
- 

  When this instruction is called by the slave axis during gear and cam following, it performs motion superimposition.



  This instruction is aborted if the MC_GearIn instruction is re-triggered during execution of this instruction.

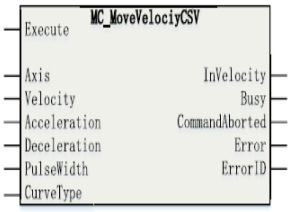
  When the MC_CamIn instruction is re-triggered during execution of this instruction, if the buffer mode is **Switch immediately**, this instruction is aborted. If the buffer mode is **Wait until the previous one is completed**, execution of this instruction continues.

  MC_GearOut and MC_CamOut can abort execution of this instruction.

5.13.1.25 MC_MoveVelocityCSV

MC_MoveVelocityCSV -- CSV-based velocity control with adjustable pulse width

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveVelocityCSV	CSV-based velocity control with adjustable pulse width		MC_MoveVelocityCSV(Execute := ???, Axis := ???, Velocity := ???, Acceleration := ???, Deceleration := , PulseWidth := , CurveType := , InVelocity => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_MoveVelocityCSV: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	-	_sMCAXIS_INFO
S2	Velocity	Target velocity	No	-	Positive number, negative number, or 0, absolute value less than the maximum velocity	REAL
S3	Acceleration	Acceleration rate	No	-	Positive number, less than the maximum acceleration rate	REAL
S4	Deceleration	Deceleration rate	Yes	Acceleration rate	Positive number, less than the maximum acceleration rate	REAL
S5	PulseWidth	Pulse width, in 0.01%	Yes	5000	1 to 9999	INT
S6	CurveType	Velocity curve type	Yes	0	0 to 1	INT

		0: T-shaped velocity curve 1: 5-segment S-shaped curve Others: T-shaped velocity curve				
D1	InVelocity	Velocity reached	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Command Aborted	Abortion of execution	Yes	OFF	ON or OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

Function Description

The MC_MoveVelocityCSV instruction uses the Cyclic Synchronous Velocity (CSV) mode to control the bus servo axis or local pulse axis to keep the PLCopen state machine of the axis in ContinuousMotion state. It has similar functions as MC_MoveVelocity.

When the bus servo axis is used, three object dictionaries need to be added to the PDO: 0x6060, 0x6061, and 0x60ff.

This instruction first writes 9 into 0x6060 to switch the drive to CSV mode, and then converts the target velocity into INT32 data and writes it into 0x60FF. The target velocity increases or decreases based on the specified acceleration rate or deceleration rate.

When the local pulse axis is used, no additional mapping parameters need to be configured. This instruction implements PWM waveform output with acceleration and deceleration. PulseWidth specifies the pulse width of the local pulse axis.

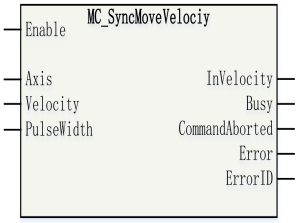
During execution of this instruction, you can call MC_Stop, MC_Halt, or MC_ImmediateStop (supported by the drive) to stop the motion of the axis.

CAUTION

The MC_MoveSuperImposed instruction cannot be called to perform motion superimposition during execution of this instruction.

5.13.1.26 MC_SyncMoveVelocity

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_SyncMoveVelocity	CSV-based synchronous velocity control of the PWM waveform		MC_SyncMoveVelocity(Enable := ???, Axis := ???, Velocity := ???, PulseWidth := , InVelocity => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_SyncMoveVelocity: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	-	_sMCAXIS_INFO
S2	Velocity	Target velocity	No	-	Positive number, negative number, or 0, absolute value less than the maximum velocity	REAL
S3	PulseWidth	Pulse width, in 0.01%	Yes	5000	1 to 9999	INT
D1	InVelocity	Velocity reached	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL

D3	CmdAborted	Abortion of execution	Yes	OFF	ON or OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

Function Description

The MC_SyncMoveVelocity instruction uses the CSV mode to control the bus servo axis or local pulse axis to keep the axis in ContinuousMotion state.

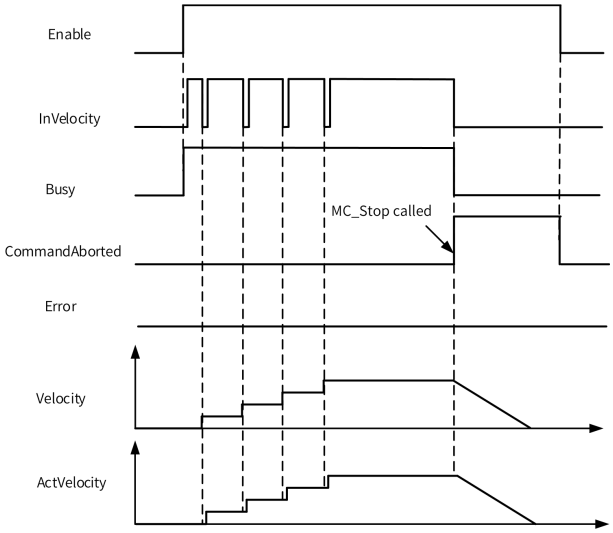
When the bus servo axis is used, three object dictionaries need to be added to the PDO: 0x6060, 0x6061, and 0x60ff.

This instruction first writes 9 into 0x6060 to switch the drive to CSV mode, and then converts the target velocity into INT32 data and writes it into 0x60FF.

When the local pulse axis is used, no additional mapping parameters need to be configured. This instruction can implement the PWM waveform output function. PulseWidth specifies the pulse width of the local pulse axis.

This instruction can modify the axis velocity and PWM duty cycle in real time in the program without re-triggering. The velocity after modification does not involve acceleration and deceleration and is directly converted into pulse unit and then written into 0x60FF.

During execution of this instruction, you can call MC_Stop, MC_Halt, or MC_ImmediateStop (supported by the drive) to stop the motion of the axis.



CAUTION

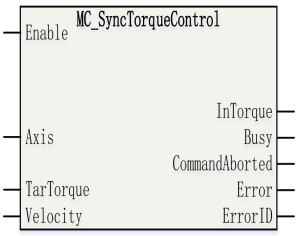
The MC_MoveSuperImposed instruction cannot be called to perform motion superimposition during execution of this instruction.

The InVelocity pin in the instruction must be set to ON in the following case:

The difference between the actual velocity specified by ActVelocity and the target velocity specified by Velocity is smaller than the velocity reach threshold configured in the mode/parameter setting of the axis at the background.

5.13.1.27 MC_SyncTorqueControl

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_SyncTorqueControl	Synchronous torque control		MC_SyncTorqueControl(Enable := ???, Axis := ???, TarTorque := ???, Velocity := , InTorque => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_SyncTorqueControl: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	Axis ID	No	-	0 to 32,767	INT
S2	TarTorque	Target torque, in 1%	No	-	Positive number, negative number, or 0	REAL
S3	Velocity	Velocity limit, in Unit This parameter is valid when 0x607F is mapped. Otherwise, it is invalid.	Yes	0	Positive number or 0	REAL
D1	InTorque	Torque reached The output is active when the set torque reaches the target torque and the	Yes	OFF	-	BOOL

		absolute value of the difference between the feedback torque and the target torque is less than 5%.				
D2	Busy	Busy flag	Yes	OFF	-	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	-	BOOL
D4	Error	Error flag	Yes	OFF	-	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

Function Description

The MC_SyncTorqueControl instruction implements synchronous torque control only for the bus servo axis. It is active on the rising edge and at high level, and does not support the virtual axis mode.

When Enable is ON, the function block converts the values specified by TarTorque and Velocity from user unit to pulse unit and transfers the data to the servo drive in real time. The axis remains in the ContinuousMotion state and performs synchronous torque motion. On the falling edge of the Enable input, the function block performs synchronous torque motion based on the values of TarTorque and Velocity latched during the high level period. Real-time modification on TarTorque and Velocity is invalid.

This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

The torque instruction can be used only when the following PDOs are configured: 0x6040, 0x6041, 0x6060, 0x6061, 0x6071, and 0x6077. Otherwise, an error is reported.

This instruction adopts the synchronous torque mode of the drive to implement the torque control function. When Enable is ON, the function block converts the values specified by TarTorque and Velocity from user unit to pulse unit and transfers the data to the servo drive in real time. The axis remains in the ContinuousMotion state and performs synchronous torque motion.

TarTorque: Target torque, in 1%. Only one decimal place after the decimal point is valid in the program, and the subsequent ones are directly discarded. The actual torque of the drive is limited by the maximum positive and negative torque specified in the configuration parameters.

Specifying the Axis

Axis is latched on the rising edge of the Enable input.

Modification on Axis is invalid when Enable is ON or OFF.

Velocity Control in Torque Mode

For servo drives of Inovance, if 0x607F is mapped, this instruction limits the maximum velocity of the servo motor through 0x607F. If 0x607F is not mapped, the velocity limit is invalid.

On the rising edge of Execute, the instruction converts the velocity limit specified by Velocity into pulse unit and writes it into 0x607F through PDO.

If the torque instruction is aborted by another instruction, the maximum velocity of the axis can be limited by specifying **Max. Velocity** on the configuration interface.

For third-party drives, Velocity can be used as the velocity limit only when the following conditions are met:

1. The maximum velocity of the servo motor can be limited by using 0x607F.
2. 0x607F can be configured in the PDO.
3. The unit of 0x607F is a pulse unit, not a rotation velocity unit.

Stop Control in Torque Mode

In torque mode, the MC_Stop instruction can be executed to stop the drive. Upon receiving the MC_Stop instruction, the drive switches to the synchronous position mode and decelerates according to the deceleration rate specified in the MC_Stop instruction.

Abortion

When this instruction is active, the axis is in ContinuousMotion state in PLCopen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCopen state machine switching. CommandAborted is active when this instruction is aborted.

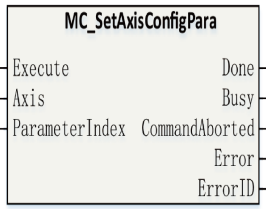
When Enable is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.

When Enable is ON and the Done signal is inactive, if the axis is disabled, CommandAborted is active.

5.13.1.28 MC_SetAxisConfigPara

MC_SetAxisConfigPara: Axis configuration parameters

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_SetAxisConfigPara	Axis configuration parameter		MC_SetAxisConfigPara(E xecute := ???, Axis := ???, ParameterIndex := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	MC_SetAxisConfigPara: Consecutive execution
32-bit instruction	-

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Name	No	-	-	_sMCAXIS_INFO
S2	ParameterIndex	Parameter index -1: All valid 0: All invalid 100: Modify only the gear ratio 200: Modify only the positive and negative software limits 300: Modify only the linearity or rotation mode 400: Modify only the encoder mode 500: Modify only the homing mode 600: Modify only the hard limit and home signal 700: Modify only the pulse output mode 800: Modify only the reverse settings 900: Modify only the virtual axis mode 1000: Modify only the probe signal 1100: Modify only the software limit variable	Yes	-1	-	INT
D1	Done	Execution completed	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT16

NOTE

*1: For details, see Axis Fault Code.

Function

This instruction is used to check and modify the configuration parameters of the axis. The axis parameters are reconfigured if they meet requirements. After configuration is completed, the Done signal output becomes active. If the configuration parameters do not meet requirements, the instruction reports an error.

ParameterIndex specifies the range of parameters to be modified. The values are described as follows:

- Parameter index -1: All parameters are updated. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.
- Parameter index 0: No parameter is updated.
- Parameter index 100: Only the gear ratio is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variable	Unit	Function
dPlusePreCycle	DINT	Pulses per revolution of the motor/encoder
fDistancePreCycle	REAL	Distance per revolution of the rotary table
dNumerator	DINT	Gear ratio numerator
dDenominator	DINT	Gear ratio denominator

- Parameter index 200: Only the positive and negative software limits are modified. Modification is allowed when the axis is in Disabled or StandStill state.

Variables	Unit	Function
bSoftLimitEnable	BOOL	Software limit enable OFF: Disabled ON: Enabled
fPLimit	REAL	Positive limit in linear mode
fNLimit	REAL	Negative limit in linear mode

- Parameter index 300: Only the linear/rotary mode is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variables	Unit	Function
iLineRotateMode	INT	Linear/Rotary mode 0: Linear mode 1: Rotary mode
fRotation	REAL	Rotation period in rotary mode

- Parameter index 400: Only the encoder mode is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variables	Unit	Function
iEncodeMode	INT	Encoder mode (valid for the bus servo axis) 0: Absolute mode 1: Incremental mode

- Parameter index 500: Only the homing mode is modified. This index is valid for the local pulse axis. Modification is allowed when the axis is in Disabled or StandStill state.

Variables	Unit	Function
iHomeMethod	INT	Homing mode
fHomeVelocity	REAL	Homing velocity
fHomeApproachVelocity	REAL	Homing approaching speed
fHomeAcceleration	REAL	Homing acceleration
dHomeTimeOut	DINT	Time Limit
iHomePositionMode	INT	Homing position mode

- Parameter index 600: Only the hardware limit and home signal are modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variables	Unit	Function
bPLimitTerminalPolarity	BOOL	Positive limit polarity OFF: Positive logic ON: Negative logic
bNLimitTerminalPolarity	BOOL	Negative limit polarity OFF: Positive logic ON: Negative logic
bHomeTerminaPolarity	BOOL	Home signal polarity OFF: Positive logic ON: Negative logic
dPLimitTerminalID	DINT	ID of the positive limit signal (Modbus address)
dNLimitTerminalID	DINT	ID of the negative limit signal (Modbus address)
dHomeTerminalID	DINT	ID of the home signal (Modbus address)

- Parameter index 700: Only the pulse output mode is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variables	Unit	Function
iPluseMethod	INT	Pulse output mode (valid for the local pulse axis)

- Parameter index 800: Only the reverse direction is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variables	Unit	Function
bDirection	BOOL	Direction OFF: Positive ON: Negative

- Parameter index 900: Only the imaginary axis mode is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variables	Unit	Function
bVirtualMode	BOOL	Imaginary axis mode OFF: Disabled ON: Enabled

- Parameter index 1000: Only the probe signal is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

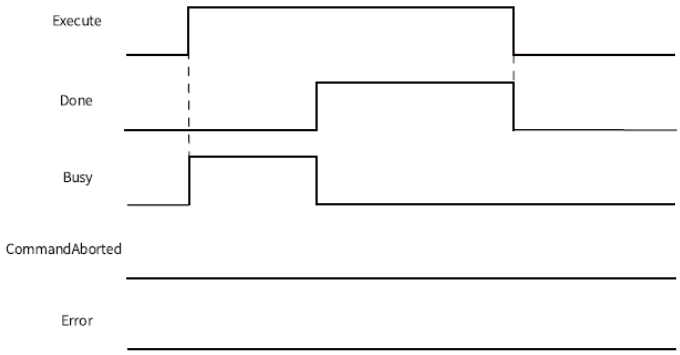
Variables	Unit	Function
dTouchProbeID1	DINT	ID of probe terminal 1
dTouchProbeID2	DINT	ID of probe terminal 2

- Parameter index 1100: Only the software limit variables are modified. Modification is allowed when the axis is in Disabled or StandStill state.

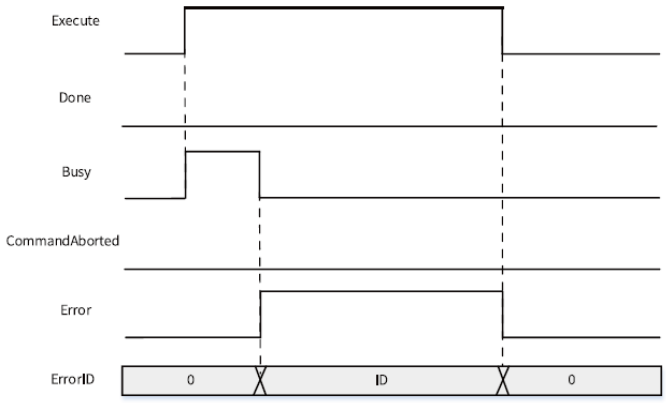
Variables	Unit	Function
fLimitDeceleraion	REAL	Limit deceleration
fErrorStopDeceleration	REAL	Deceleration upon axis fault
fFollowErrorWindow	REAL	Following error window
fInVelocityWindow	REAL	Velocity window
fMaxVelocity	REAL	Maximum velocity

Variables	Unit	Function
fMaxJogVelocity	REAL	Maximum jogging velocity
fMaxAcc	REAL	Maximum acceleration
fMaxPTorque	REAL	Maximum positive torque
fMaxNTorque	REAL	Maximum negative torque
bEnterErrorStop	BOOL	Not entering ErrorStop state upon an axis fault OFF: Disabled ON: Enabled

Sequence Diagram

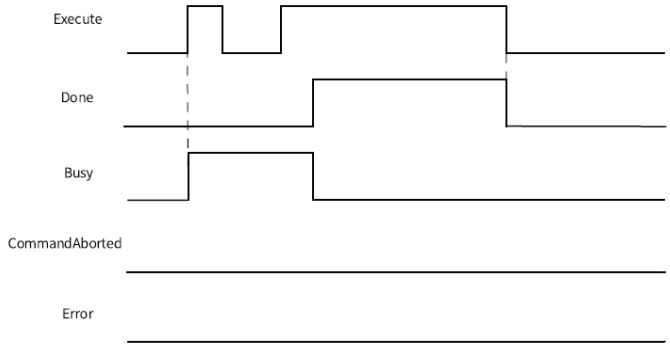


This instruction reports a fault when the parameters are configured improperly.



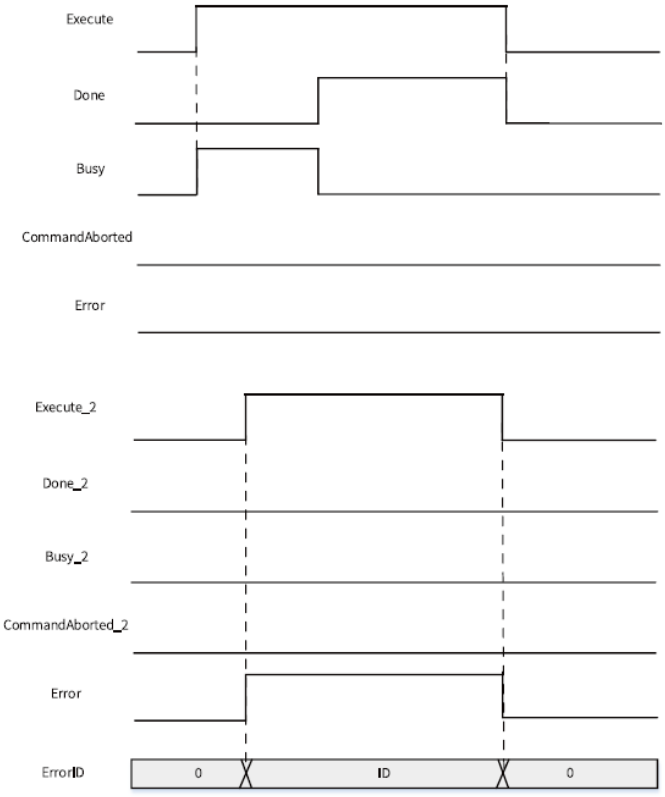
Re-triggering

This instruction can be re-triggered while the Busy output is still active.



Multi-execution

It is not allowed to call a second MC_MetAxisConfigPara instruction while the Busy output of this instruction is still active; otherwise, the second instruction reports an error.



5.13.1.29 MC_SetOverride

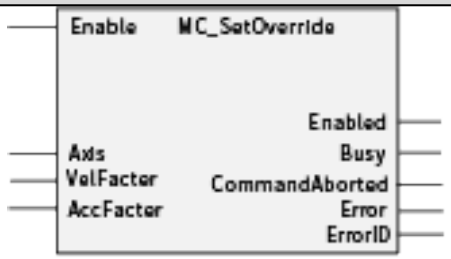
Introduction

The MC_SetOverride instruction adjusts the target velocity during motion.

MC_SetOverride Adjust velocity factors during motion

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_SetOverride	Adjust velocity factors during motion		-

Instruction	Name	LD Expression	LiteST Expression
			

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_SetOverride: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	-	_sMCAXIS_INFO
S2	VelFacter	Velocity overshoot	Yes	100	0 to 500	REAL32
S3	AccFacter	Acceleration and deceleration overshoot	Yes	100	0 to 500	REAL32
D1	Enabled	Enable flag	Yes	FALSE	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	FALSE	ON or OFF	BOOL
D3	CommandAborted	Execution abortion flag	Yes	FALSE	ON or OFF	BOOL
D4	Error	Error flag	Yes	FALSE	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT16

NOTE

*1: For details, see Axis Error Codes.

Function and Instruction Description

The MC_SetOverride instruction adjusts the target velocity and preset acceleration/deceleration rate during the motion of the controlled axis by setting the velocity and acceleration overshoot. This instruction applies

only to the master axis. It cannot be used to set the motion parameters of a slave axis, and it does not affect the status of axes in synchronous motion.

Motion parameters after adjustment:

- Target velocity = Current target velocity of the executing instruction x Velocity overshoot
- Preset acceleration/deceleration rate = Current acceleration/deceleration rate of the executing instruction x Acceleration overshoot

The unit of an overshoot value is [%], and 100 indicates 100%. The overshoot value ranges from 0 to 500, with a default value of 100. The velocity overshoot value cannot be less than 0. The acceleration overshoot value must be greater than 0. If the velocity overshoot value is set to 0, the axis decelerates and moves at a velocity of 0. The axis maintains its original motion state. If the target velocity after adjustment exceeds the maximum velocity in system settings, the target velocity is limited to the maximum velocity in system settings.

When the Enable signal is reset from TRUE to FALSE, 100 is returned for the velocity and acceleration overshoot.

By executing the start from stop instruction, re-executing a motion instruction, or executing multiple motion instructions, you can set the new target velocity based on the overshoot value.

The following table lists the instructions for which the overshoot can be adjusted.

The following table lists instructions with adjustable overshoot.

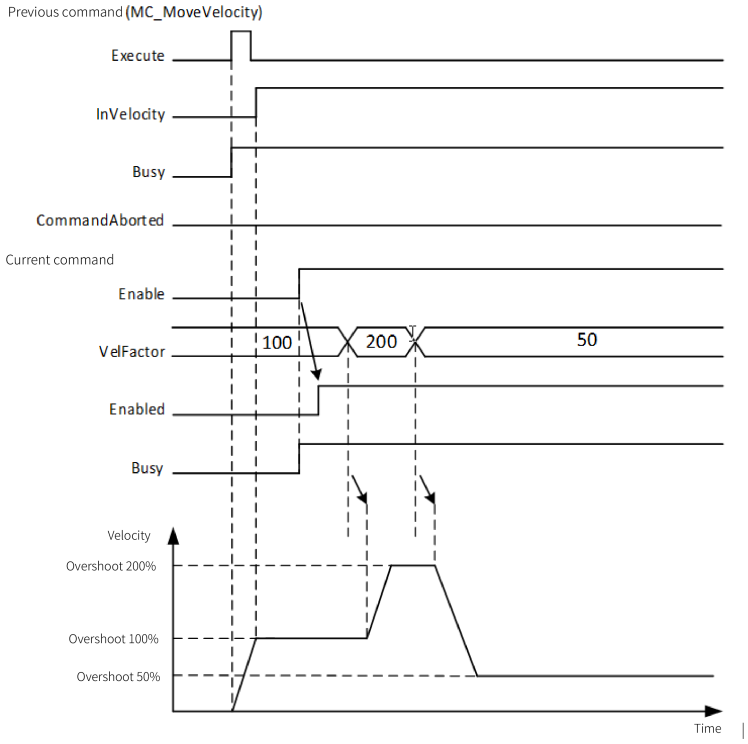
Instruction with Adjustable Overshoot	
MC_MoveAbsolute (Absolute positioning)	MC_MoveRelative (Relative positioning)
MC_MoveVelocity (Velocity control)	MC_Jog (Jogging)
MC_MoveVelocityCSV (Velocity control with adjustable pulse width)	MC_SyncMoveVelocity (Synchronous velocity control that supports PWM waveform)
MC_FollowVelocity (CSP-based synchronous velocity control)	MC_MoveVelocityPV (Profile velocity control)
MC_MoveAbsRelPP (Profile positioning in aborting mode)	MC_JogP (Profile jogging)
MC_MoveBuffer (Multi-segment positioning)	-



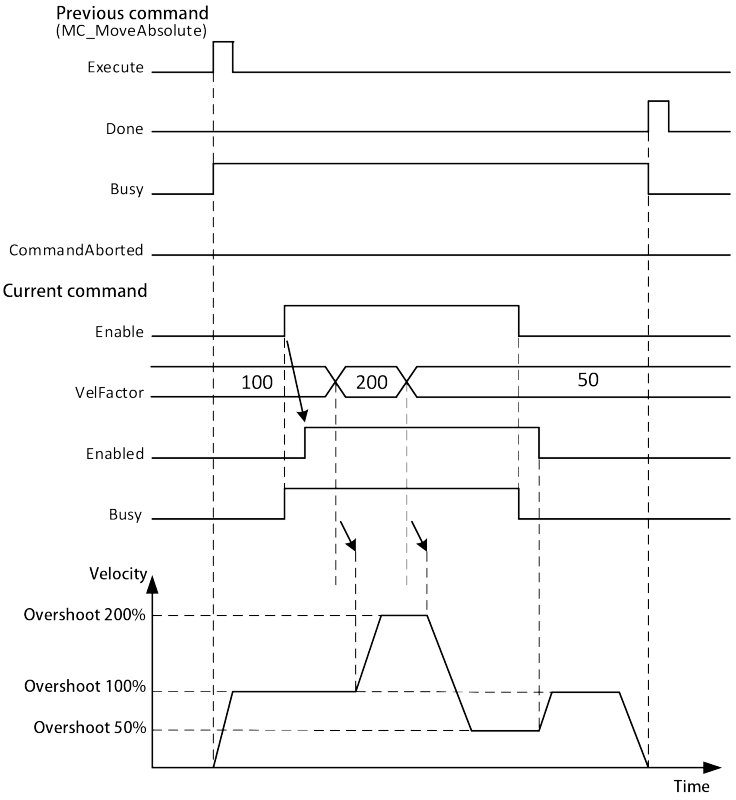
The MC_SetOverRide instruction has no effect on MC_MoveSuperImposed.

Timing Diagram

- Overshoot of the MC_MoveAbsolute (absolute positioning) instruction
  The MC_SetOverride instruction is used to implement overshoot of the MC_MoveAbsolute (absolute positioning) instruction.



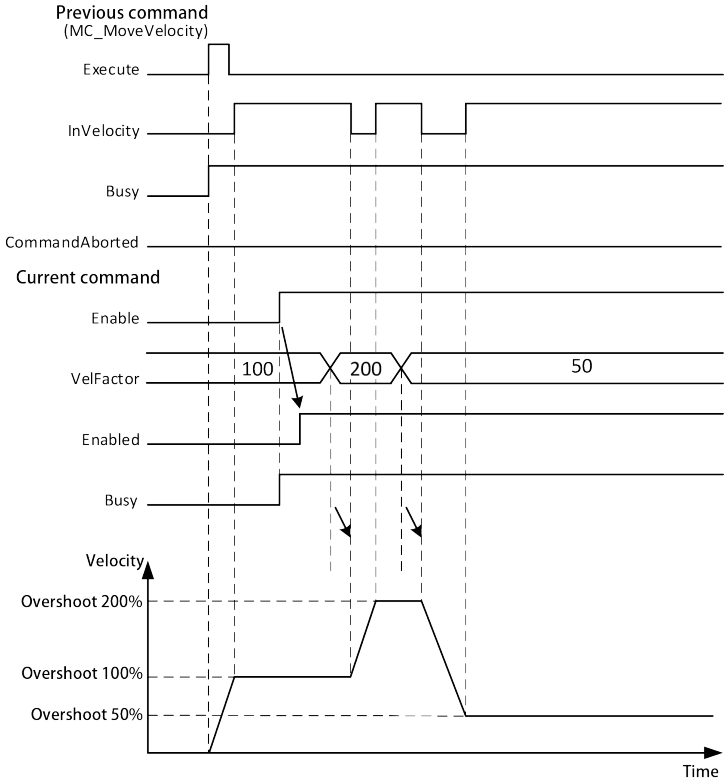
- The MC_SetOverride instruction is used to implement acceleration/deceleration overshoot of the MC_MoveAbsolute (absolute positioning) instruction.



When the MC_SetOverride instruction is inactive, the preset acceleration/deceleration rate with the overshoot value of 100% is returned.

- Overshoot of the MC_MoveVelocity (velocity control) instruction

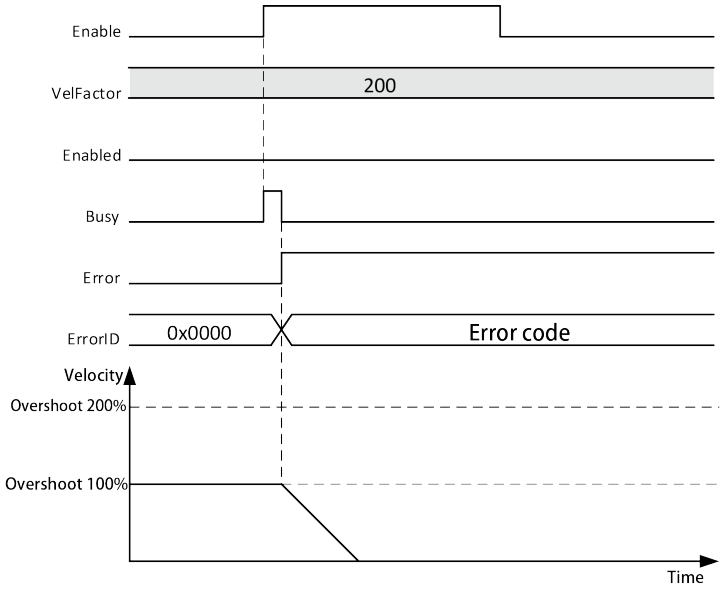
After InVelocity becomes TRUE (indicating that the target velocity has been reached), it changes to FALSE when the velocity is changed, and it changes back to TRUE when the target velocity is reached again.



- Timing diagram illustrating an exception
  When an exception occurs during execution of this instruction, the Error flag is set to TRUE, indicating that a minor fault occurs and the axis stops its motion.

  You can check the output of ErrorID to find the cause of the exception.

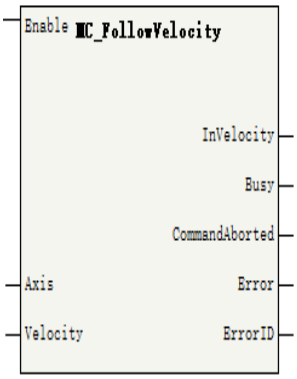
  After the exception is resolved, the Error flag becomes FALSE.



5.13.1.30 MC_FollowVelocity

MC_FollowVelocity -- CSP-based synchronous velocity control

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_FollowVelocity	CSP-based synchronous velocity control		MC_FollowVelocity(Enable := ???, Axis := ???, Velocity := ???, InVelocity => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_FollowVelocity: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	-	_sMCAXIS_INFO
S2	Velocity	Target velocity	No	-	Positive number, negative number, or 0, absolute value less than the maximum velocity	REAL32
D1	InVelocity	Velocity reached	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON or OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT16

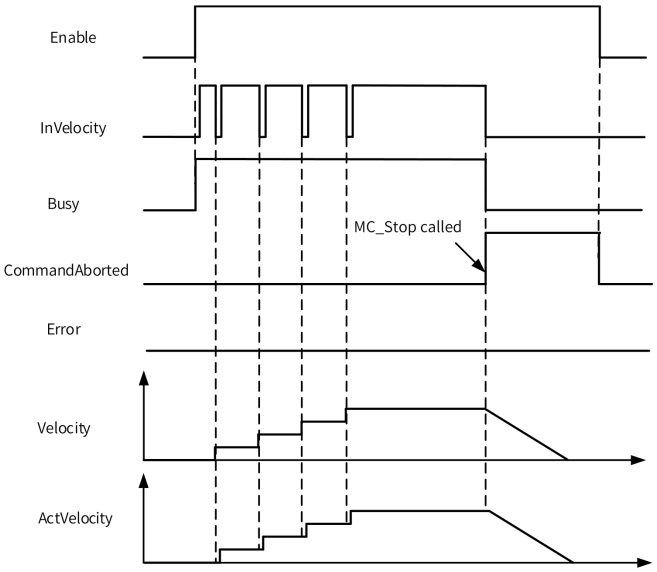
NOTE

*1: For details, see Axis Error Codes.

Function Description

The MC_FollowVelocity instruction works with the MC_MoveSuperImposed instruction to implement the motion superimposition function. This instruction uses the CSP mode to control the bus servo or local pulse axis to keep the axis in SynchronizedMotion state. It works with MC_MoveSuperImposed to implement motion superimposition.

When Enable is ON, the velocity specified by Velocity takes effect immediately after modification, without the need to re-trigger the instruction. The velocity after modification does not involve acceleration and deceleration. The way that the InVelocity signal is triggered is affected by the velocity window in axis configuration. To stop the axis, you need to call the MC_Stop instruction.

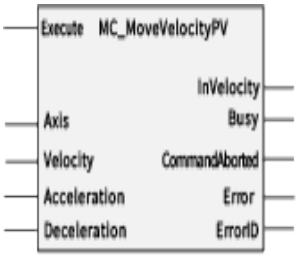


This instruction can work with the MC_MoveSuperImposed instruction to implement the motion superimposition function.

5.13.1.31 MC_MoveVelocityPV

MC_MoveVelocityPV -- Profile velocity control

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveVelocityPV	Profile velocity control		Not supported

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_MoveVelocityPV: Profile velocity control					
Operand	Name	Description		Default	Range	Data Type

			Empty Allowed			
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_ INFO
S2	Velocity	Target velocity	No	-	Positive number, negative number, or 0, absolute value less than the maximum velocity	REAL
S3	Acceleration	Profile acceleration	No	-	Positive number, less than the maximum acceleration rate	REAL
S4	Deceleration	Profile deceleration	Yes	Acceleration rate	Positive number, less than the maximum acceleration rate	REAL
D1	InVelocity	Velocity reached	Yes	FALSE	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	FALSE	ON or OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	FALSE	ON or OFF	BOOL
D4	Error	Error flag	Yes	FALSE	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT



*1: For details, see Axis Error Codes.

Function Description

The MC_MoveVelocityPV instruction implements the continuous motion function in profile velocity (PV) mode for the EtherCAT bus profile axis. This instruction can be called only when the profile mode option is selected in the basic axis configuration. It is active on the rising edge.

This is a velocity simulation instruction in PV mode.

- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - Modification on Axis is invalid when Execute is ON or OFF.
- Functions
  This instruction can be executed only after the MC_Power instruction is executed to change the axis to the StandStill state.

When Execute is TRUE, the instruction latches the input parameters on the left, such as Velocity and Acceleration, triggers the axis to run continuously at the target velocity specified by Velocity, and switches the PLCopen state machine of the axis to the ContinuousMotion state.

During execution of this instruction, you can call MC_Stop, MC_Halt, or MC_ImmediateStop (supported by the drive) to stop the motion of the axis.

The following table describes object dictionaries to be configured in the profile velocity control instruction.

RPDO	TPDO
6040h: control word	6041h: status word
60FFh: Target velocity	-
-	6064h: position actual value
-	606Ch: velocity actual value
6083h: profile acceleration	-
6084h: profile deceleration	-
6060h: modes of operation	6061h: mode of operation display

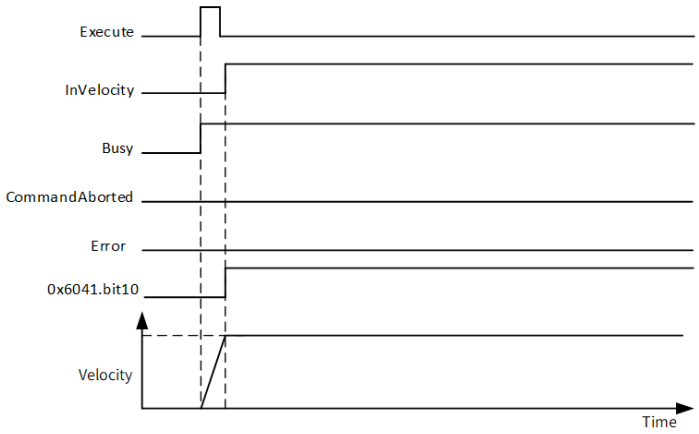
Abortion

- When this instruction is active, the axis is in ContinuousMotion state in PLCopen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCopen state machine switching in profile mode. CommandAborted is active when this instruction is aborted.
- When Execute is TRUE and the Busy signal is active, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

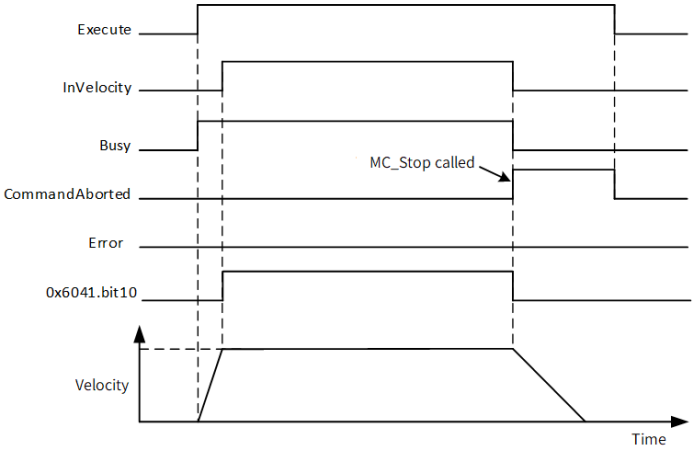
1. The state of the InVelocity pin in this instruction is related to the value of bit 10 in the status word 6041h and the velocity reach threshold specified in the background. If bit 10 in the status word 6041h is ON and the difference between the actual velocity and the target velocity is less than the velocity reach threshold specified in the background, InVelocity must be set to ON. In PV mode, bit 10 in the status word 6041h has a strong relationship with the object dictionaries 606Dh (velocity reach threshold) and 606Eh (velocity window time). For details about how to set the object dictionaries, see the servo commissioning manual.
2. In PV mode, if Unit is set to position (fSetPosition), the value is the same as the actual position specified by fActPosition. If Unit is set to velocity (fSetVelocity), the value is the same as the target velocity set in the instruction. If Unit is set to acceleration and deceleration (fSet_Acc_Dec), the value is the same as the acceleration rate set in the instruction. Monitor the PDOs dProfileAeccleration (6083h) and dProfileDeccleration (6084h) to monitor delivered acceleration rate or deceleration rate.

Timing Diagram

- If the instruction is called when the axis is in StandStill state, the axis switches to PV mode and performs continuous motion.



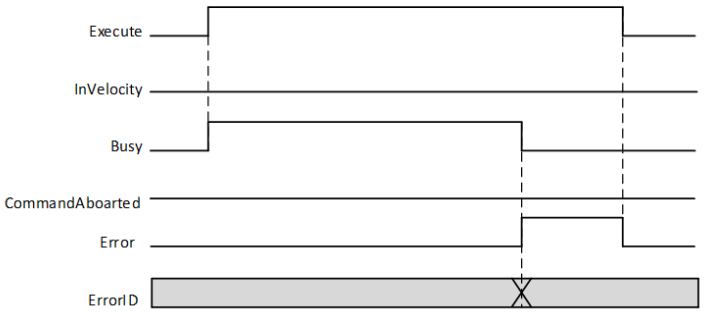
- Axis motion is aborted by the MC_Stop instruction.



NOTE

When the MC_Stop instruction is called to stop the axis motion in PV mode, the axis stops by the deceleration rate specified by profile deceleration (0x6084h) or the deceleration rate of the servo specified by H0A.72. The shorter time required to stop the axis prevails.

- The drive fault occurs.



NOTE

When the servo works in profile mode, if servo stop is triggered by a fault or hard limit, bit 8 in the control word 0x6040 is set to 1. At this time, the deceleration rate in ramp stop is determined by the following factors:

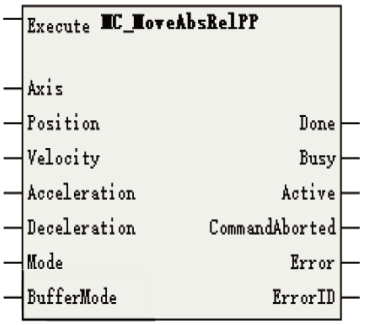
1. Setpoint of 0x6084
2. Maximum downtime in ramp stop specified by H0A.72

The smaller one of the preceding two factors prevails for servo stop deceleration.

5.13.1.32 MC_MoveAbsRelPP

MC_MoveAbsRelPP -- Positioning based on the profile position mode

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveAbsRelPP	Positioning based on the profile position mode		Not supported

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_MoveAbsRelPP: Positioning based on the profile position mode					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_ INFO
S2	Position	Target position or distance	No	-	Positive number Negative number 0	REAL
S3	Velocity	Profile velocity	No	-		REAL

					Positive number, less than the maximum velocity	
S4	Acceleration	Profile acceleration	No	-	Positive number, less than the maximum acceleration rate	REAL
S5	Deceleration	Profile deceleration	Yes	Acceleration rate	Positive number, less than the maximum acceleration rate	REAL
S6	Mode	Positioning mode 0: Absolute positioning 1: Relative positioning	Yes	0	0 to 1	INT
S7	Buffer Mode	Buffer mode 0: Aborting and no transition 1: Buffered and no transition 2: Buffered and transition with the current velocity (Not supported)	Yes	0	0 to 2	INT
D1	Done	Target reached	Yes	FALSE	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	FALSE	ON or OFF	BOOL
D3	Active	Controlling	Yes	FALSE	ON or OFF	BOOL
D4	CommandAborted	Abortion of execution	Yes	FALSE	ON or OFF	BOOL
D5	Error	Error flag	Yes	FALSE	ON or OFF	BOOL
D6	ErrorID	Error code	Yes	0	*1	INT

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit				Word			Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	Bits of Custom Word Variable	D, R, W	Pt	Custom Word Variable	K, H	E	
S1	-	-	-	-	√	√	√	√	-	-
S2	-	-	-	-	√	√	√	-	√	-
S3	-	-	-	-	√	√	√	-	√	-
S4	-	-	-	-	√	√	√	-	√	-
S5	-	-	-	-	√	√	√	-	√	-
S6	-	-	-	-	√	√	√	√	-	-
S7	-	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	√	-	-	-	-	-	-
D2	√[1]	√	√	√	-	-	-	-	-	-
D3	√[1]	√	√	√	-	-	-	-	-	-
D4	√[1]	√	√	√	-	-	-	-	-	-
D5	-	-	-	-	√	-	√	-	-	-

NOTE

[1]: The X element is not supported.

Function Description

The MC_MoveAbsRelPP instruction implements the absolute or relative positioning function in profile position mode for the EtherCAT bus profile axis. This instruction can be called only when the profile mode option is selected in the basic axis configuration. It is active on the rising edge.

- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis ID, modification on Axis is invalid when Execute is ON or OFF.
- Functions
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

On the rising edge of the Execute input, the instruction latches the input parameters on the left, such as Position and Velocity, triggers the absolute or relative positioning function, and switches the PLCOpen state machine of the axis to the DiscreteMotion state.

- When Mode is set to 0, absolute positioning mode is used. Position specifies the target position in absolute positioning mode. If the axis mode is set to rotation mode, the target position is determined based on the shortest distance.
- When Mode is set to 1, relative positioning mode is used. Position specifies the target distance in relative positioning mode.
- When BufferMode is set to 0, current motion and new motion during positioning are updated in the following way: current motion aborted without velocity transition.
- When BufferMode is set to 1, current motion and new motion during positioning are updated in the following way: new motion buffered and executed after current motion execution is completed, without velocity transition.

During execution of this instruction, you can call MC_Stop, MC_Halt, or MC_ImmediateStop to stop the motion of the axis.

The following table describes object dictionaries to be configured in the profile position control instruction.

RPDO	TPDO
6040h: control word	6041h: status word
607Ah: target position	6064h: position actual value
6081h: profile velocity	-
6083h: profile acceleration	-
6084h: profile deceleration	-
6060h: modes of operation	6061h: mode of operation display

NOTE

1. In ring and absolute positioning mode, if the target position is greater than the ring cycle, the instruction uses the target position to mod the ring cycle to obtain a new target position. If the absolute value of the difference between the new target position and the set position of the axis is less than 0.001, the two values are considered equal.
2. When BufferMode is set to 2 (Buffered and transition with the current velocity), which is not supported, the instruction functions in the same way as BufferMode is set to 1.
3. In profile position mode, fSetPosition, fSetVelocity, and fSet_Acc_Dec specify the target position, profile velocity, and profile acceleration (profile deceleration not displayed). The axis system variables dProfileVelocity (0x6081), dProfileAcceleration (0x6083), and dProfileDeceleration (0x6084) are used to monitor PDO parameters related to the profile mode.
4. In profile position mode, the state of the Done pin in the instruction is related to bits 10 and 12 of the status word 6041h. When bit 10 in 6041h is 1 and bit12 is 0, the target position is reached and Done is set to ON. In profile position mode, the setting of bit 10 in the status word 6041h has strong relationships with object dictionaries 6067h (position window) and 6068h (position window time). For details about how to configure object dictionary parameters, see the corresponding servo commissioning manual.

Abortion

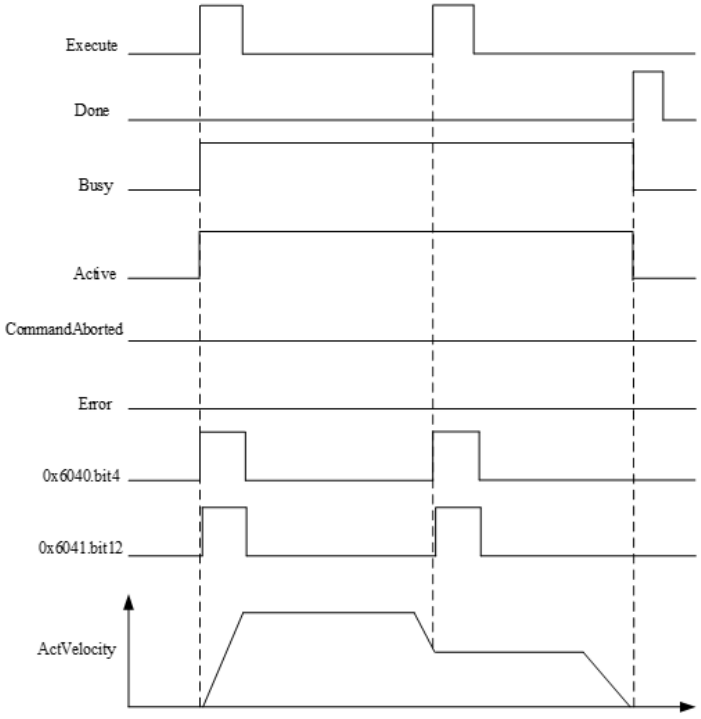
- When this instruction is active, the axis is in DiscreteMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted. Instructions in cyclic synchronous position mode cannot be called. Otherwise, an error is reported.
- When Execute is TRUE and the Busy signal is active, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

Errors

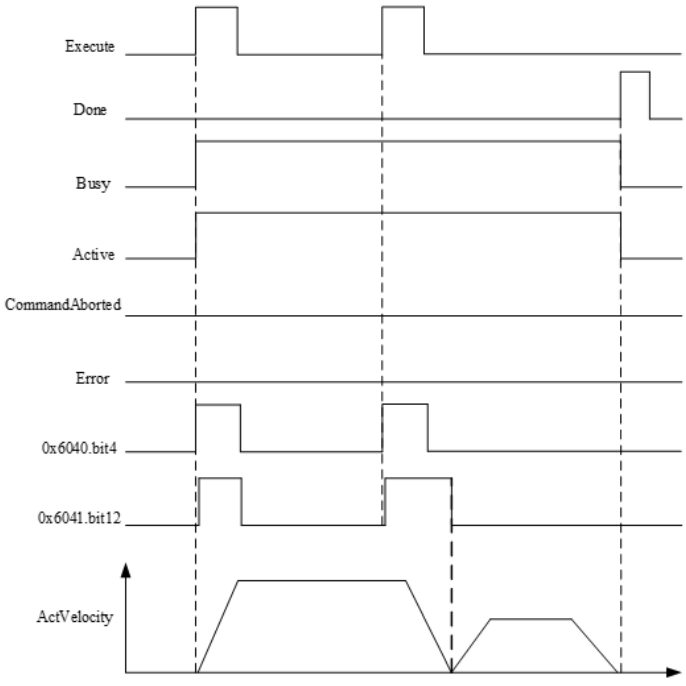
- Error 9101 is reported when the axis ID does not exist or the axis type does not match.
- Error 9102 is reported when axis initialization fails.
- Error 9108 is reported if this instruction is executed when the axis is not in StandStill, DiscreteMotion, or ContinuousMotion state.
  - Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
  - Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
  - An error is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
  - Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
  - If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

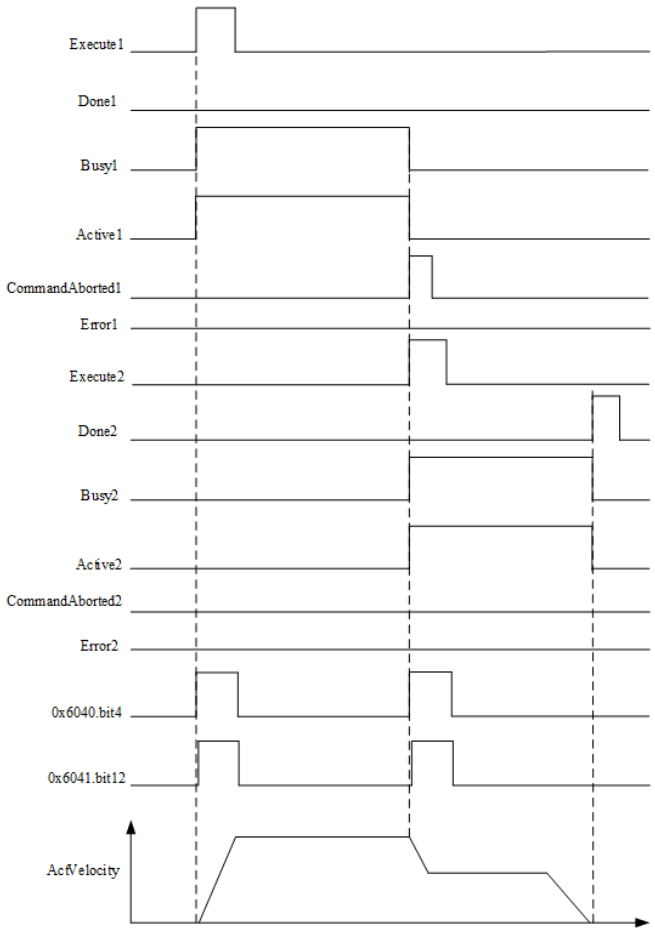
- If the instruction is called when the axis is in StandStill state, the axis performs positioning in profile position mode.
  When BufferMode is set to 0:



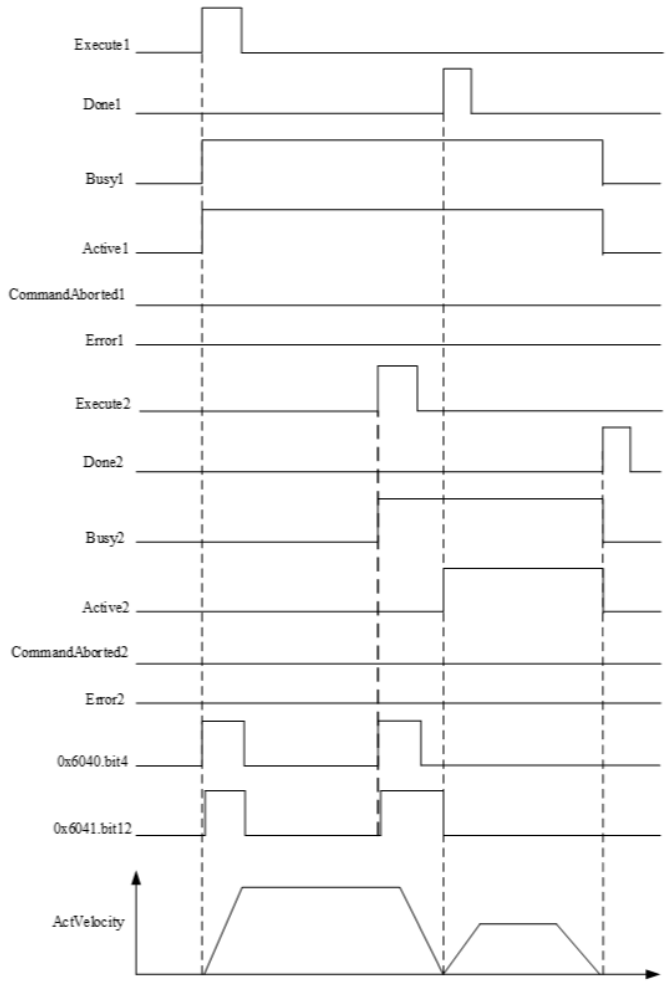
  When BufferMode is set to 1:



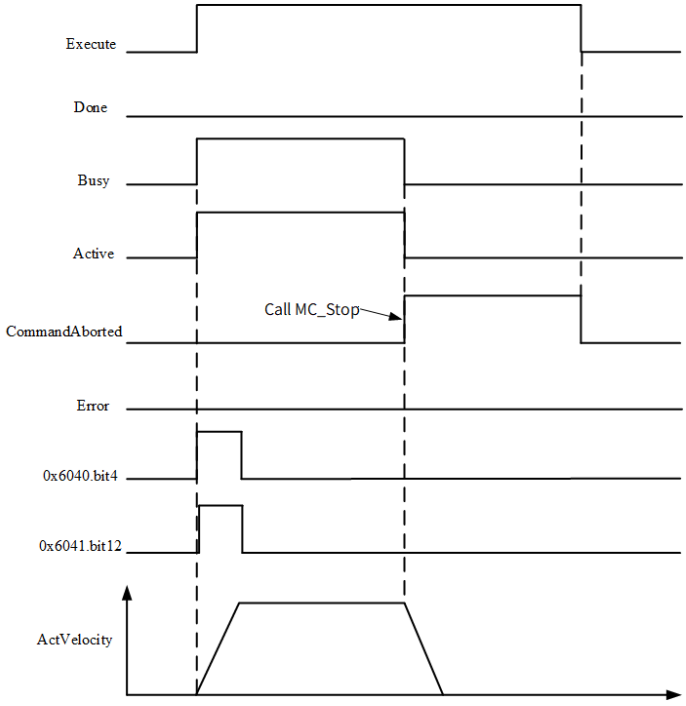
- When two profile positioning instructions are called, the second instruction is executed in aborting and no transition mode (BufferMode set to 0).



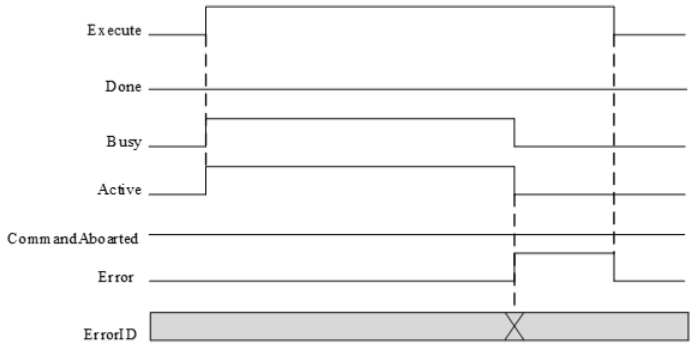
- When two profile positioning instructions are called, the second instruction is executed in buffered and no transition mode (BufferMode set to 1).



- Axis motion is aborted by the MC_Stop instruction.



- The drive fails during motion of the axis.



NOTE

When the servo works in profile mode, if servo stop is triggered by a fault or hard limit, bit 8 in the control word 0x6040 is set to 1. At this time, the deceleration rate in ramp stop is determined by the following factors:

1. Setpoint of 0x6084
2. Maximum downtime in ramp stop specified by H0A.72

The smaller one of the preceding two factors prevails for servo stop deceleration.

H0A.72 **Maximum downtime in ramp stop**

Hexadecimal parameter: 200A-49h
Minimum: 0
Maximum: 65535
Default: 10000
Value range: 0 ms to 65535 ms

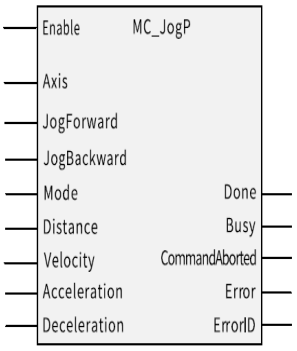
Effective mode: At once
Unit: ms
Data type: UInt16
Change mode: At stop

Description

Defines the maximum time taken by the motor in decelerating from 6000 rpm to 0 rpm when the stop mode is set to "Ramp to stop as defined by 6084h/609Ah (HM)" or "Ramp to stop as defined by 6085h".

5.13.1.33 MC_JogP

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_JogP	Jogging based on the profile mode		Not supported

The following table describes the instruction format.

16-bit Instruction	-

32-bit Instruction	MC_JogP: Jogging based on the profile mode					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_ INFO
S2	JogForward	Jogging in forward direction, triggered on the rising edge	No	-	-	BOOL
S3	JogBackward	Jogging in reverse direction, triggered on the rising edge	No	-	-	BOOL
S4	Mode	Mode: 0: Jogging by target velocity 1: Inching by target distance	Yes	0	0 to 1	INT
S5	Distance	Target distance (valid when Mode is set to 1)	No	-	Positive number 0	REAL
S6	Velocity	Target velocity or profile velocity (determined by mode)	No	-	Positive number, less than the maximum velocity in jogging or inching mode	REAL
S7	Acceleration	Profile acceleration	No	-	Positive number, less than the maximum acceleration rate	REAL
S8	Deceleration	Profile deceleration	Yes	Acceleration rate	Positive number, less than the maximum acceleration rate	REAL
D1	Done	Target reached	Yes	FALSE	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	FALSE	ON or OFF	BOOL

D3	CommandAborted	Abortion of execution	Yes	FALSE	ON or OFF	BOOL
D4	Error	Error flag	Yes	FALSE	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit				Word			Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	Bits of Custom Word Variable	D, R, W	Pt	Custom Word Variable	K, H	E	
S1	-	-	-	-	√	√	√	√	-	-
S2	√	√	√	√	-	-	-	-	-	-
S3	√	√	√	√	-	-	-	-	-	-
S4	-	-	-	-	√	-	√	√	-	-
S5	-	-	-	-	√	√	√	-	√	-
S6	-	-	-	-	√	√	√	-	√	-
S7	-	-	-	-	√	√	√	-	√	-
S8	-	-	-	-	√	√	√	-	√	-
D1	√[1]	√	√	√	-	-	-	-	-	-
D2	√[1]	√	√	√	-	-	-	-	-	-
D3	√[1]	√	√	√	-	-	-	-	-	-
D4	√[1]	√	√	√	-	-	-	-	-	-
D5	-	-	-	-	√	-	√	-	-	-

NOTE

[1]: The X element is not supported.

Function Description

The MC_JopP instruction implements the jogging or inching function in profile position or velocity mode for the EtherCAT bus profile axis. This instruction can be called only when the profile mode option is selected in the basic axis configuration. It is active in case of high level.

- Specifying the axis
  - Axis is latched on the rising edge of the Enable input.
  - If Axis specifies the axis ID, modification on Axis is invalid when Enable is ON or OFF.
- Functions
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.
  - When Enable is ON, the instruction obtains input parameters on the left, such as Mode and Distance.
    - When Mode is set to 0, the PLCOpen state machine of the axis changes to the ContinuousMotion state.
    - When Mode is set to 1, the PLCOpen state machine of the axis changes to the DiscreteMotion state.
  - When Enable is ON:
    - If Mode is set to 0 and JogForward is active, the axis moves forward at the target velocity specified by Velocity.
    - If JogBackward is active, the axis moves backward at the target velocity specified by Velocity.
    - If JogForward or JogBackward is active on the falling edge, the axis decelerates.
  - When Enable is ON, if Mode is set to 1 and JogForward is active on the rising edge, the axis moves forward for the target distance specified by Distance.
    - If Mode is set to 1 and JogBackward is active on the rising edge, the axis moves backward for the target distance specified by Distance.
    - If both JogForward and JogBackward are active, the axis stops but does not enter the ErrorStop state, and the instruction reports an error.
  - When Enable is ON, if the axis reaches the limit when moving toward one direction, the instruction reports an error, and the axis stops but does not enter the ErrorStop state. In this case, the axis may move in the direction reverse to the limit.
    During execution of this instruction, you can call MC_Stop, MC_Halt, or MC_ImmediateStop to stop the motion of the axis.
    - The following table describes object dictionaries to be configured to implement the jogging function in profile velocity mode when Mode is set to 0.

RPDO	TPDO
6040h: control word	6041h: status word
60FFh: target velocity	-
-	6064h: position actual value

RPDO	TPDO
-	606Ch: velocity actual value
6083h: profile acceleration	-
6084h: profile deceleration	-
6060h: modes of operation	6061h: mode of operation display

- The following table describes object dictionaries to be configured to implement the inching function in profile position mode when Mode is set to 1.

RPDO	TPDO
6040h: control word	6041h: status word
607Ah: target position	6064h: position actual value
6081h: profile velocity	-
6083h: profile acceleration	-
6084h: profile deceleration	-
6060h: modes of operation	6061h: mode of operation display

Abortion

- When Enable is ON, if instructions such as MC_Stop and MC_MoveAbsRelPP are called, the MC_JogP instruction is aborted, and the CommandAborted output of MC_JogP instruction becomes active. It is aborted in inching mode (Mode set to 1) and when BufferMode is set to 0.
  When this instruction is active, the axis is in ContinuousMotion or DiscreteMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter ContinuousMotion or DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted. Instructions in cyclic synchronous position mode cannot be called. Otherwise, an error is reported.
- When Enable is TRUE and the Busy signal is active, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

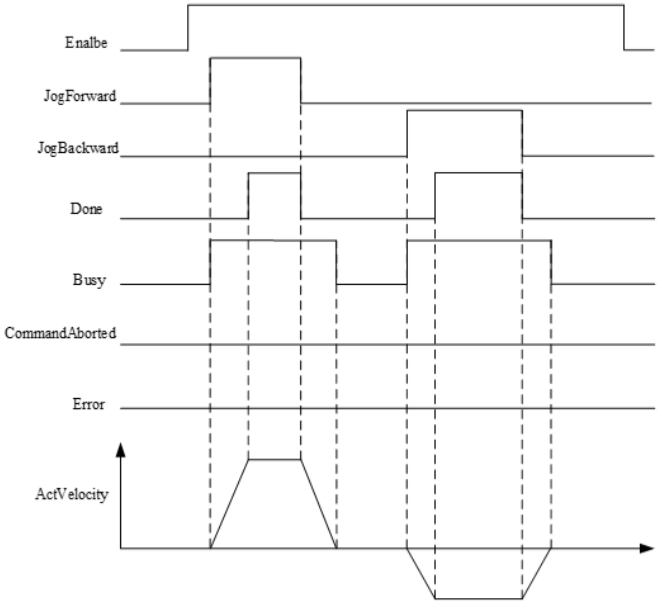
Errors

- Error 9101 is reported when the axis ID does not exist or the axis type does not match.
- Error 9102 is reported when axis initialization fails.
- Error 9108 is reported if this instruction is executed when the axis is not in StandStill, DiscreteMotion, or ContinuousMotion state.
  - Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Enable input.
  - Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Enable input.

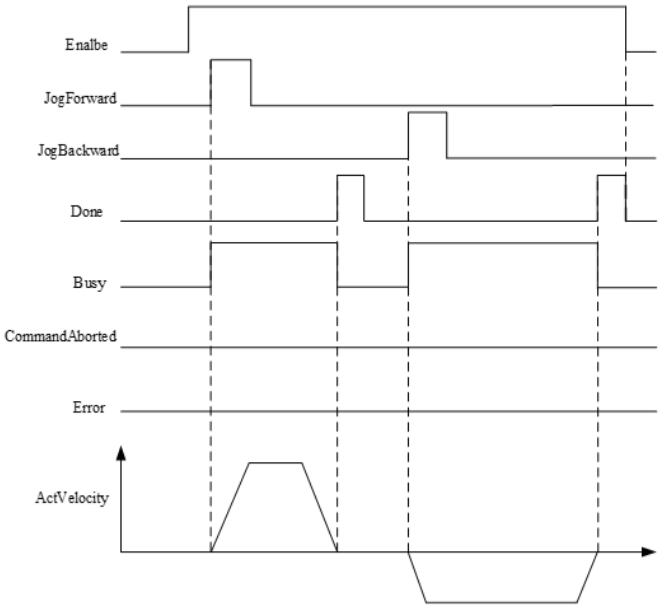
- An error is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Enable input.
- Error 9116 is reported if the axis enters the commissioning state when Enable is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Enable is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

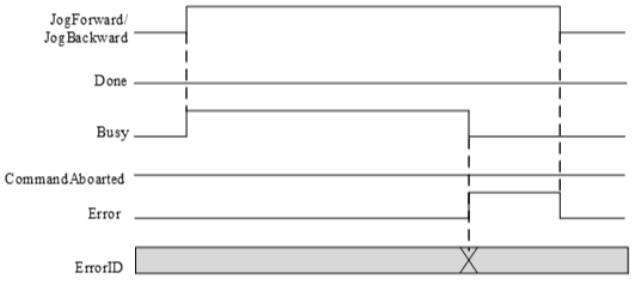
- This instruction is called when the axis is in the StandStill state. Enable is ON, Mode is set to 0, and JogForward or JogBackward is at high level.



- This instruction is called when the axis is in the StandStill state. Enable is ON, Mode is set to 1, and JogForward or JogBackward is active on the rising edge.



- The drive fails during motion of the axis.



NOTE

When the servo works in profile mode, if servo stop is triggered by a fault or hard limit, bit 8 in the control word 0x6040 is set to 1. At this time, the deceleration rate in ramp stop is determined by the following factors:

1. Setpoint of 0x6084
2. Maximum downtime in ramp stop specified by H0A.72

The smaller one of the preceding two factors prevails for servo stop deceleration.

H0A.72 **Maximum downtime in ramp stop**

Hexadecimal parameter: 200A-49h
Minimum: 0
Maximum: 65535
Default: 10000
Value range: 0 ms to 65535 ms

Effective mode: At once
Unit: ms
Data type: UInt16
Change mode: At stop

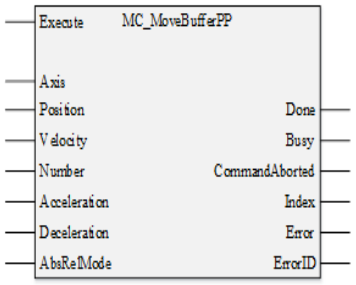
Description

Defines the maximum time taken by the motor in decelerating from 6000 rpm to 0 rpm when the stop mode is set to "Ramp to stop as defined by 6084h/609Ah (HM)" or "Ramp to stop as defined by 6085h".

5.13.1.34 MC_MoveBufferPP

MC_MoveBufferPP -- Multi-segment positioning in profile position mode

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveBufferPP	Multi-segment positioning in profile position mode		Not supported

The following table describes the instruction format.

	-

16-bit Instruction						
32-bit Instruction	MC_MoveBufferPP: Multi-segment positioning in profile position mode					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_ INFO
S2	Position	Target position or start address of the distance	No	-	Positive number Negative number 0	REAL and array*16
S3	Velocity	Start address of the profile velocity	No	-	Positive number, less than the maximum velocity	REAL and array*16
S4	Number	Number of buffer pairs	No	-	1 to 16	INT
S5	Acceleration	Profile acceleration	No	-	Positive number, less than the maximum acceleration rate	REAL
S6	Deceleration	Profile deceleration	Yes	Acceleration rate	Positive number, less than the maximum acceleration rate	REAL
S7	AbsRelMode	Positioning mode 0: Absolute positioning 1: Relative positioning	Yes	0	0 to 1	INT
D1	Done	Stop completed	Yes	FALSE	-	BOOL
D2	Busy	Busy flag	Yes	FALSE	-	BOOL
D3	CommandAborted	Abortion of execution	Yes	FALSE	-	BOOL
D4	Index	Current segment	Yes	0	0 to 16	INT
D5	Error	Error flag	Yes	FALSE	-	BOOL
D6	ErrorID	Error code	Yes	0	*1	INT

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit				Word			Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	Bits of Custom Word Variable	D, R, W	Pt	Custom Word Variable	K, H	E	
S1	-	-	-	-	√	√	√	√	-	-
S2	-	-	-	-	√	√	√	-	√	-
S3	-	-	-	-	√	√	√	-	√	-
S4	-	-	-	-	√	√	√	√	-	-
S5	-	-	-	-	√	√	√	-	√	-
S6	-	-	-	-	√	√	√	-	√	-
S7	-	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	√	-	-	-	-	-	-
D2	√[1]	√	√	√	-	-	-	-	-	-
D3	√[1]	√	√	√	-	-	-	-	-	-
D4	-	-	-	-	√	-	√	-	-	-
D5	√[1]	√	√	√	-	-	-	-	-	-
D6	-	-	-	-	√	-	√	-	-	-

NOTE

[1]: The X element is not supported.

Function Description

The MC_MoveBufferPP instruction implements the multi-segment positioning function in profile position mode for the EtherCAT bus profile axis. This instruction can be called only when the profile mode option is selected in the basic axis configuration. It is active on the rising edge.

The following table describes the object directories that need to be configured for a multi-segment positioning instruction.

RPDO	TPDO
6040h: control word	6041h: status word
607Ah: target position	6064h: position actual value
6081h: profile velocity	-
6083h: profile acceleration	-
6084h: profile deceleration	-
6060h: modes of operation	6061h: mode of operation display

- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis ID, modification on Axis is invalid when Execute is ON or OFF.
- Functions
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.
  - On the rising edge of the Execute input, the instruction latches the input parameters on the left, such as Position, Velocity, Number, Acceleration, and Deceleration and triggers the multi-segment positioning function.
  - The axis performs absolute positioning (AbsRelMode = 0) or relative positioning (AbsRelMode = 1) in profile position aborting mode based on the value of AbsRelMode. The supported position can be up to 16 segments.
  - Position specifies the target position. It is of the array type and supports up to 16 levels. It specifies the target position of the axis in absolute positioning mode or target travel distance of the axis in relative positioning mode. It specifies the shortest distance to the target position in absolute positioning in rotation mode.
  - Velocity specifies the target velocity. It is of the array type and supports up to 16 levels.
  - Number specifies the number of target positions and the number of profile velocity arrays. The value ranges from 1 to 16. An error is reported if the value is out of range.
  - Acceleration and Deceleration specify profile acceleration and profile deceleration in profile mode.

  During execution of this instruction, you can call MC_Stop, MC_Halt, or MC_ImmediateStop to stop the motion of the axis.

Abortion

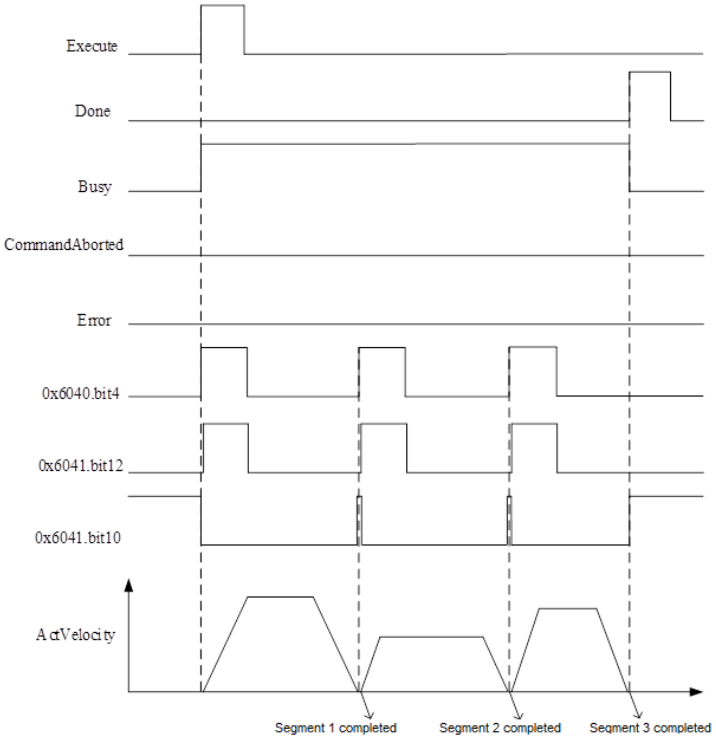
- When this instruction is active, the axis is in DiscreteMotion state in PLCopen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCopen state machine switching. CommandAborted is active when this instruction is aborted. Instructions in cyclic synchronous position mode cannot be called. Otherwise, an error is reported.
- When Enable is TRUE and the Busy signal is active, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

Errors

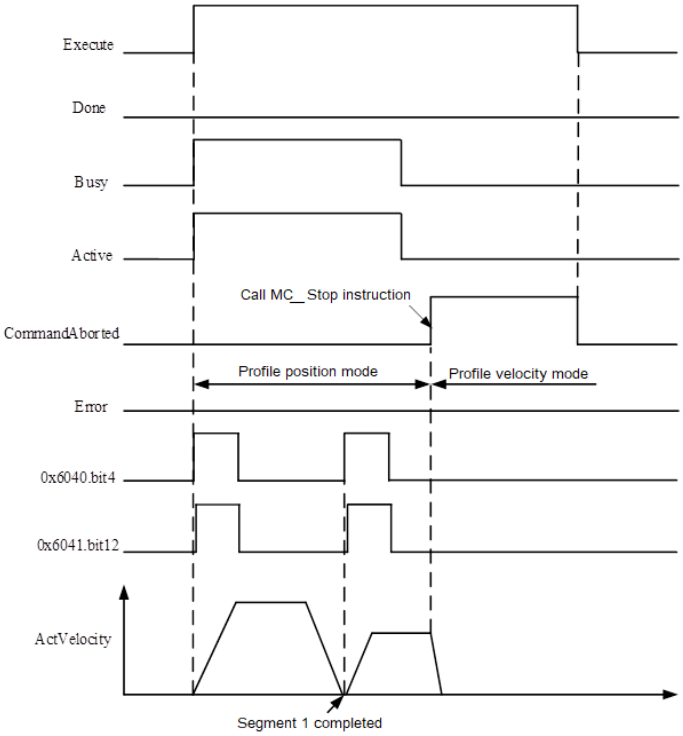
- Error 9101 is reported when the axis ID does not exist or the axis type does not match.
- Error 9102 is reported when axis initialization fails.
- Error 9108 is reported if this instruction is executed when the axis is not in StandStill, DiscreteMotion, or ContinuousMotion state.
  - Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
  - Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
  - An error is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
  - Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
  - If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

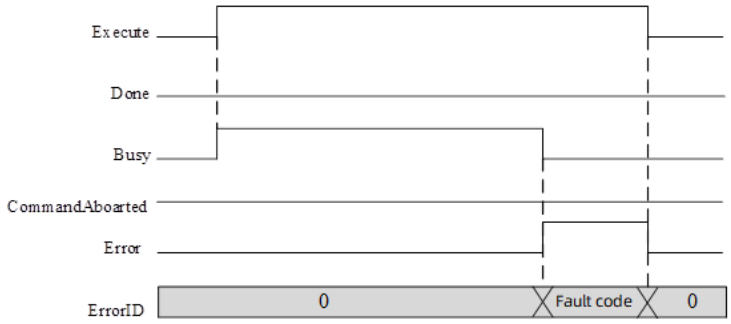
- The instruction is called when the axis is in StandStill state to switch the axis to profile position mode to perform 3-segment positioning.



- Axis motion is aborted by the MC_Stop instruction.



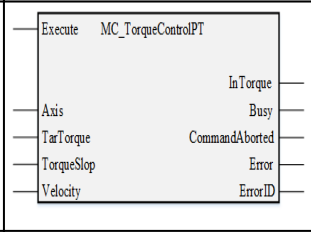
- The drive fails during motion of the axis.



5.13.1.35 MC_TorqueControlPT

MC_TorqueControlPT -- Torque control in profile torque mode

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_TorqueControlPT	Torque control in profile torque mode		Not supported

The following table describes the instruction format.

16-bit Instruction	-

32-bit Instruction	MC_TorqueControlPT: Torque control in profile torque mode					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID	No	-	0 to 32,767	INT _sMCAXIS_INFO
S2	TarTorque	Target torque, in 1%	No	-	Positive number Negative number 0	REAL
S3	TorqueSlope	Torque slope, in %/s	No	-	Positive number	REAL
S4	Velocity	Velocity (This parameter is valid when 0x607F is mapped.)	Yes	0	Positive number 0	REAL
D1	InTorque	Torque reached	Yes	FALSE	-	BOOL
D2	Busy	Busy flag	Yes	FALSE	-	BOOL
D3	Command Aborted	Abortion of execution	Yes	FALSE	-	BOOL
D4	Error	Error flag	Yes	FALSE	-	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit				Word			Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	Bits of Custom Word Variable	D, R, W	Pt	Custom Word Variable	K, H	E	
S1	-	-	-	-	√	√	√	√	-	-
S2	-	-	-	-	√	√	√	-	√	-
S3	-	-	-	-	√	√	√	-	√	-
S4	-	-	-	-	√	√	√	-	√	-
D1	√[1]	√	√	√	-	-	-	-	-	-
D2	√[1]	√	√	√	-	-	-	-	-	-
D3	√[1]	√	√	√	-	-	-	-	-	-
D4	√[1]	√	√	√	-	-	-	-	-	-
D5	-	-	-	-	√	-	√	-	-	-

NOTE

[1]: The X element is not supported.

Function Description

The MC_TorqueControlPT instruction implements the torque control function in profile torque mode for the EtherCAT bus profile axis. This instruction can be called only when the profile mode option is selected in the basic axis configuration. It is active on the rising edge.

The following table describes the object directories that need to be configured for a multi-segment positioning instruction.

RPDO	TPDO
6040h: control word	6041h: status word
6071h: target torque	6064h: position actual value
6087h: torque slope	-
-	6077h: torque actual value
6060h: modes of operation	6061h: mode of operation display

- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis ID, modification on Axis is invalid when Execute is ON or OFF.
- Functions
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.
  - The instruction latches the input parameters TarTorque, TorqueSlope, and Velocity on the rising edge of the Execute input. The axis enters the ContinuousMotion state and performs torque motion in profile torque mode.
  - TarTorque specifies the target torque, which is limited by the maximum positive and negative torque set in the configuration.
  - TorqueSlope specifies the torque slope, which is the torque reference acceleration rate in profile torque mode, that is the torque reference increment per second.
  - Velocity control in torque mode:
    For servo drives of Inovance, if 0x607F is mapped, this instruction limits the maximum velocity of the servo motor through 0x607F. If 0x607F is not mapped, the velocity limit is invalid.

    If the torque instruction is aborted by another instruction, the maximum velocity of the axis can be limited by specifying **Max. Velocit**y on the configuration interface.
  - Stop control in torque mode:
    In torque mode, the MC_Stop instruction can be executed to stop the drive. Upon receiving the MC_Stop instruction, the drive switches to the profile velocity mode and decelerates according to the deceleration rate specified in the MC_Stop instruction.

Abortion

- When this instruction is active, the axis is in ContinuousMotion state in PLCopen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCopen state machine switching in profile mode. CommandAborted is active when this instruction is aborted.
- When Execute is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.
- When Execute is ON and the Done signal is inactive, if the axis is disabled, CommandAborted is active.

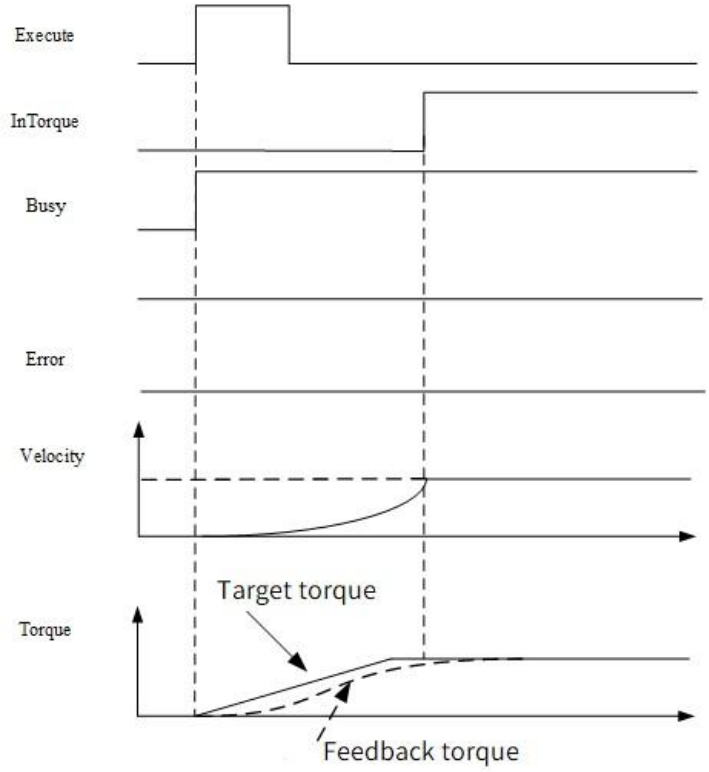
Errors

- Error 9101 is reported when the axis ID does not exist or the axis type does not match.
- Error 9102 is reported when axis initialization fails.
- Error 9108 is reported if this instruction is executed when the axis is not in StandStill, DiscreteMotion, or ContinuousMotion state.
  - Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
  - Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
  - An error is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.

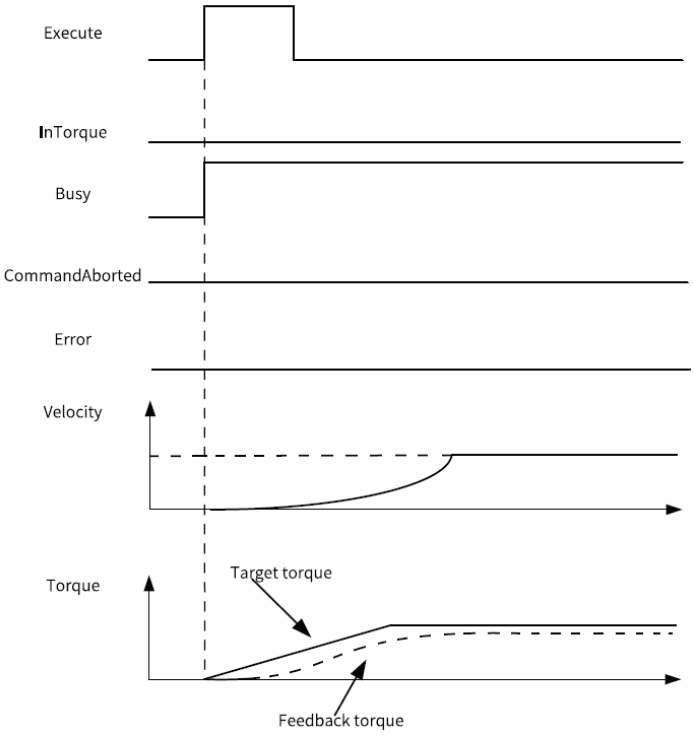
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

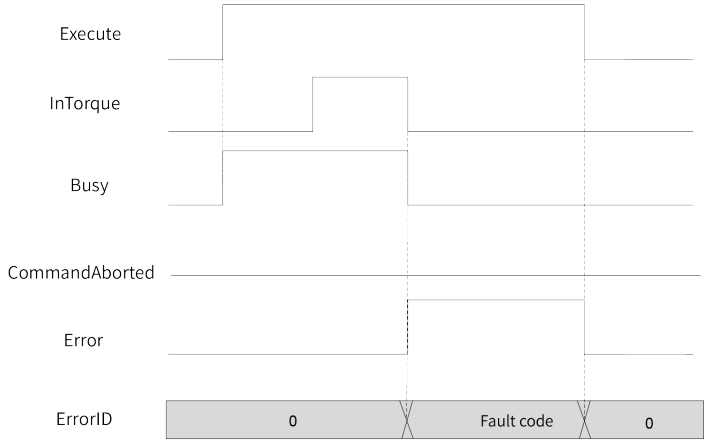
- The instruction is called to switch from profile torque mode to torque control mode when the axis is in StandStill state.



- The MC_Stop instruction aborts the instruction execution during torque operation.



- The drive reports an error during torque operation.



Note

Software-limit stop in profile torque mode triggers the servo to stop by setting the control word bit 8 to 1 when the actual position that is detected reaches the software limit. In this case, the final position exceeds the software limit.

5.13.1.36 MC_ResetFollowingError

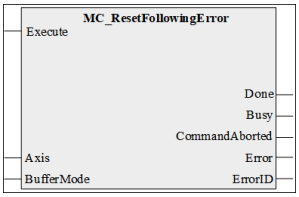
Version

Software version: AutoShop V4.11.0.0 and later

Firmware version: V6.5.0.0 and later

Introduction

MC_ResetFollowingError -- Following error reset

Instruction	Name	LD Expression	LiteST Expression
MC_ResetFollowingError	Following error reset		MC_ResetFollowingError (Execute := ???, Axis := ???, BufferMode := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	MC_ResetFollowingError -- Reset following error					
32-bit Instruction	---					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type

S1	Axis	Axis name or axis ID	No	-	0 to 32767	INT _sMCAXIS_ INFO
S2	BufferMode	Buffer mode: 0: Aborted Others: Reserved	Yes	0	0	INT
D1	Done	Reset completed	Yes	FALSE	-	BOOL
D2	Busy	Executing	Yes	FALSE	-	BOOL
D3	CommandAborted	Abortion of execution	Yes	FALSE	-	BOOL
D4	Error	Error flag	Yes	FALSE	-	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Axis Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit				Word			Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	Bits of Custom Word Variable	D, R, W	Pt	Custom Word Variable	K, H	E	
S1	-	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	√	-	-	-	-	-	-
D2	√[1]	√	√	√	-	-	-	-	-	-
D3	√[1]	√	√	√	-	-	-	-	-	-
D4	√[1]	√	√	√	-	-	-	-	-	-
D5	-	-	-	-	√	-	√	-	-	-

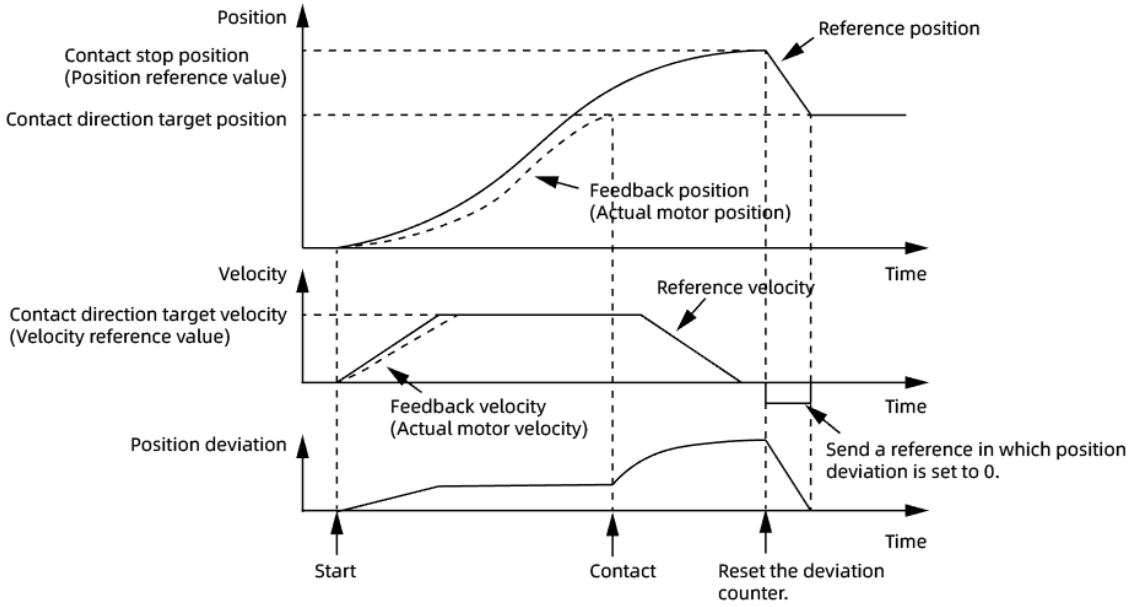
NOTE

[1]: The X element is not supported.

Function Description

- The MC_ResetFollowingError instruction resets the following error between the target position and the current position in CSP or PP mode. If it is triggered in other modes, an error will be reported. It is applicable to the EtherCAT bus axis, profile axis, and local axis, and is triggered on the rising edge.
- When the bus servo axis or profile axis is used, the object directory 0x6061 needs to be added to the PDO. Otherwise, an error will be reported.
- This instruction sets the target position of the axis to the current position on the active rising edge of the Execute signal to align the target position of the axis with the current position. After the positions are aligned, the Done output becomes active. Otherwise, the Error signal is active, and ErrorID specifies the reason for the reset failure.
- When you call this instruction to reset the following error in rotary mode, it resets the error on the motion direction by the shortest distance for the rotary axis. Modular calculation must be performed on the calculated target position to ensure that it is within the single turn.

As shown in the figure below, when this instruction is started during a contact action in which a position deviation occurs, the position instruction is issued in negative direction so that the position deviation is "0".



- Specifying the axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
  - If Axis specifies the axis ID, modification on Axis is valid when Execute is OFF.
- Reset the following error in the following ways based on the situation:
  - If this instruction is triggered during axis motion, this instruction will abort the current active instruction.
    - CSP mode: After this instruction is triggered, the target position is the same as the current position, the target velocity is the same as the maximum velocity configured for the axis, and the target acceleration/deceleration rate is not limited. The system re-plans the velocity and calculates the cyclical movement based on the curve type specified by the previous instruction. Superimposition of the target position and the cyclical movement makes the axis align with the current position and aborts the current action.
    - PP mode: After this instruction is triggered, the target position is the same as the current position, the profile velocity is the same as the maximum velocity configured for the axis, and the profile

acceleration/deceleration rate is not limited. The system aligns the target position with the current position and aborts the current action based on the control word and status word.

  - If this instruction is triggered when the axis is in StandStill state:
    The system immediately aligns the target position with the current position without aborting the current action.

- Deviation reset velocity

  If the MC_ResetFollowingError instruction is executed during axis motion, it resets the deviation by using the maximum velocity of the axis as the target velocity, not limited by the maximum acceleration/deceleration rate. This process does not engage acceleration/deceleration. The instruction does not check whether the maximum acceleration rate is exceeded during execution.

  - If this instruction is started at a high axis velocity during axis motion, it may cause a shock to the machine. Therefore, you must start the instruction at a low axis velocity.
  - If the master axis works in synchronous control mode, when this instruction is started, the slave axis may rotate in the reverse direction by following the master axis based on the gear ratio and cam data change.

Re-execution

Repeated triggering of the instruction will report an error indicating that the axis is in error reset.

Abortion

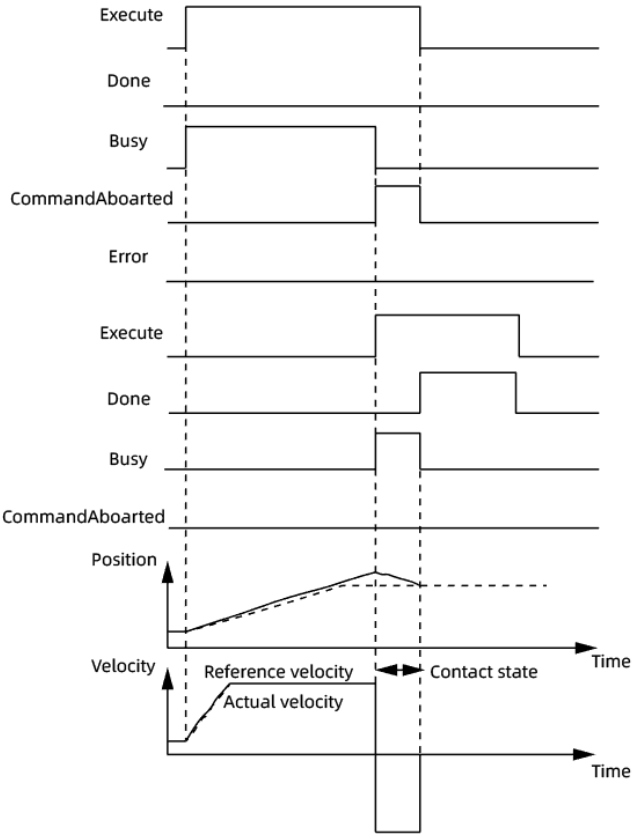
When this instruction is active, the axis is in Stopping state, and the instruction cannot be aborted by other motion instructions. After this instruction resets the following error, the axis switches from the Stopping state to the StandStill state, and other motion control instructions can run.

Errors

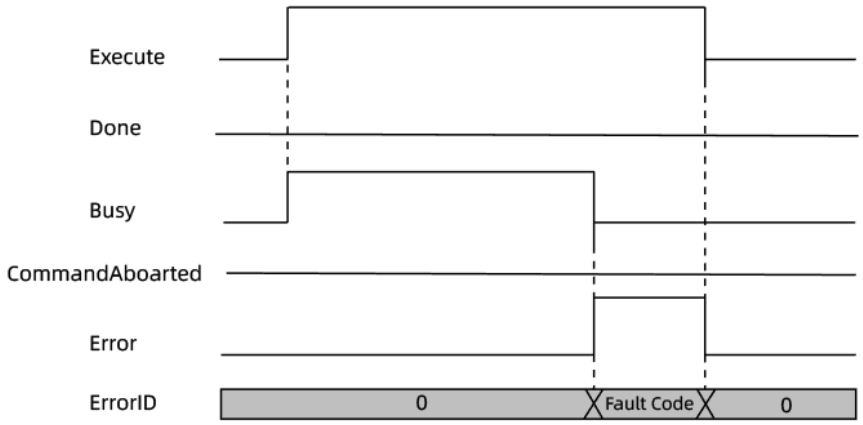
- Error 9101 is reported when the axis ID does not exist or the axis type does not match.
- Error 9102 is reported when axis initialization fails.
- Error 9142 is reported if this instruction is executed after the MC_ImmediateStop instruction is executed to put the axis in Stopping state.
  - Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
  - Error 9108 is reported if the axis is in Disabled, ErrorStop, Homing, or Stopping state on the rising edge of the Execute input.
  - If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the error code of the axis in the ErrorStop state.

Timing Diagram

- The MC_ResetFollowingError instruction is started in the contact status after the MC_MoveAbsolute instruction is started.



- The drive fails during motion of the axis.



5.13.1.37 Axis Fault Codes

Axis fault codes are divided into local pulse axis fault codes and motion control axis fault codes. If an axis instruction reports a fault, see the description of the corresponding fault code.

Local Pulse Axis Fault Codes

Fault code	Fault information	Solution	Stop triggered
9001 (0x2329)	Emergency stop The emergency stop terminal input is triggered.	Disable the emergency stop terminal input and then call the MC_Reset instruction to reset the fault.	Yes

Fault code	Fault information	Solution	Stop triggered
9002 (0x232a)	The following error is too large. (Reserved)	Adjust the target velocity of acceleration (deceleration).	Yes
9003 (0x232b)	Overspeed The pulse output frequency exceeds 200 kHz.	The pulse output frequency exceeds 200 kHz.	Yes
9020 (0x233c)	Homing error The negative limit is not mapped.	Map the negative limit on the configuration interface.	Yes
9021 (0x233d)	Homing error The positive limit is not mapped.	Map the positive limit on the configuration interface.	Yes
9022 (0x233e)	Homing error The home signal is not mapped.	Map the home switch on the configuration interface.	Yes
9023 (0x233f)	Homing error The output frequency exceeds 200 kHz when the axis runs at the homing velocity. The output frequency exceeds 200 kHz when the axis runs at the homing approach velocity.	• Modify the unit conversion setting to ensure that the homing velocity and homing approach velocity do not exceed 200 kHz. • Change the homing velocity to ensure that the output frequency does not exceed 200 kHz. • Change the homing approach velocity to ensure that the output frequency does not exceed 200 kHz.	Yes
9024 (0x2340)	Homing error Homing timed out.	• Check that the limit signal is conductive. • Check whether the homing timeout time is too short.	Yes
9025 (0x2341)	A homing error occurs. The limit signal is disordered during homing.	Check whether the limit signal that is not applicable to the current homing mode is triggered.	Yes
9030 (0x2342)	Limit function enabled	Check whether the limit is reached during normal running.	No
9031 (0x2343)	Synchronization error	Check whether the limit is reached during normal positioning.	No

Fault code	Fault information	Solution	Stop triggered
	The target number of transmitted pulses and the actual number of transmitted pulses do not match.		

NOTE

When a local pulse axis is faulty, refer to the preceding fault code list for fault information.

Motion Control Axis Fault Codes

Fault code	Fault information	Solution	Stop triggered
9101	The type of the axis specified by AxisID is incorrect. The axis specified by AxisID does not exist.	• Check whether the instruction supports the axis specified by AxisID. • Check whether the axis specified by AxisID exists.	No
9102	The axis configuration data is lost. The axis configuration parameters are improper.	Check whether the parameters are correct.	No
9103	The MC_Reset instruction is called when the axis is not faulty.	Check whether the MC_Reset instruction is called when the axis is not switched to ErrorStop state.	No
9104	The axis is in unknown state when the MC_ReadStatus instruction is called.	Check whether the current state of the axis is uncontrollable by using the online monitoring function.	No
9105	Current position setting not allowed	Check whether the MC_SetPositon instruction has been called.	No
9106	The axis is stopping upon a fault.	Execute the instruction after stop upon fault is completed and the fault is resolved.	No
9107	The parameters are improper.	Check whether the parameters on the left of the instruction are set properly.	Yes
9108	The PLCOpen state machine is improper.	Check whether the current PLCOpen state machine satisfies the execution conditions for this instruction. If not, call the relevant instruction to switch the axis to the required state.	No

Fault code	Fault information	Solution	Stop triggered
9109	The axis enters the Disabled state during instruction execution.	Check whether the axis has entered the Disabled state.	No
9110	The MC_Stop instruction is called repeatedly during stop.	Check whether the MC_Stop instruction is called repeatedly in the program.	No
9111	The instruction linked list is lost.	Check whether the software tool version and board version match.	No
9112	AxisID changed The axis number changes while the instruction flow is active.	Do not change the axis number while the flow is active for Enable instructions such as MC_Power and MC_Jog.	Yes
9113	Reset by executing the MC_Reset instruction timed out.	Check whether the drive fault can be reset. Check whether the fault type supports reset.	No
9114	The axis fails to write to 0x6060.	Check for interference in network communication.	No
9115	The MC_Halt instruction is called when the axis is in Stopping state.	Do not call the MC_Halt instruction when the axis is in Stopping state.	No
9116	The current axis is in online commissioning mode.	Check whether the current axis is in online commissioning mode.	No
9117	Reserved	Reserved	No
9118	The acceleration (deceleration) of the instruction exceeds the maximum acceleration.	Check whether the acceleration (deceleration) of the instruction exceeds the maximum acceleration.	Yes
9119	The target velocity of the MC_Jog instruction exceeds the maximum jogging velocity.	Check whether the target velocity of the MC_Jog instruction exceeds the maximum jogging velocity.	Yes
9120	The target velocity exceeds the maximum velocity.	Check whether the target velocity of the instruction exceeds the maximum velocity.	Yes
9121	The forward and reverse motion signals of the jog instruction are both active.	Ensure that the forward and reverse motion signals of the jog instruction are not active at the same time.	Yes
9122	The control word is not mapped to the EtherCAT bus axis.	Add the control word in the PDO and map it to the axis.	No
9123	The target position is not mapped to the EtherCAT bus axis.	Add the target position in the PDO and map it to the axis.	No

Fault code	Fault information	Solution	Stop triggered
9124	The target torque is not mapped to the EtherCAT bus axis.	Add the target torque in the POD and map it to the axis.	No
9125	The status word is not mapped to the EtherCAT bus axis.	Add the status word in the PDO and map it to the axis.	No
9126	The current position is not mapped to the EtherCAT bus axis.	Add the feedback position in the PDO and map it to the axis.	No
9127	The 0x60fd is not mapped to the EtherCAT bus axis.	Add 0x60fd in the PDO and map it to the axis.	No
9128	The current torque is not mapped to the EtherCAT bus axis.	Add the current torque in the PDO and map it to the axis.	No
9129	The probe control word is not mapped to the EtherCAT bus axis.	Add the probe control word in the PDO and map it to the axis.	Yes (interrupt positioning) No (probe)
9130	The probe status word is not mapped to the EtherCAT bus axis.	Add the probe status word in the PDO and map it to the axis.	Yes (interrupt positioning) No (probe)
9131	The probe position is not mapped to the EtherCAT bus axis.	Add the probe position in the PDO and map it to the axis.	Yes (interrupt positioning) No (probe)
9132	An interrupt positioning instruction is being executed and the probe channel is occupied.	The probe instruction and interrupt positioning instruction must not occupy the same probe channel at the same time. When the two instructions are called simultaneously in the program, the interrupt positioning instruction takes priority.	No
9133			No

Fault code	Fault information	Solution	Stop triggered
	The imaginary axis mode is enabled.	The current instruction does not support the imaginary axis mode.	
9134	The imaginary axis probe is being used.	Two imaginary axis probes are supported. Check whether the current probe is out of range.	No
9135	The interrupt signal is not triggered in the interrupt positioning instruction.	During execution of the interrupt positioning instruction, no interrupt signal is detected after positioning is completed.	No
9136	The probe channel is occupied by another instruction during the interrupt positioning process.	Ensure that the probe channel is not occupied during the interrupt positioning process.	Yes
9137	The control mode 0x6060 is not mapped to the bus driver.	Add 0x6060 in the PDO and map it to the axis.	No
9138	The control mode 0x6061 is not mapped to the bus driver.	Add 0x6061 in the PDO and map it to the axis.	No
9139	The MC_Home instruction is called repeatedly during homing.	Do not call the MC_Home instruction repeatedly during homing.	No
9140	The target torque of the instruction exceeds the maximum value.	Check whether the target torque of the instruction exceeds the positive and negative torque limits.	Yes
9141	The maximum velocity is not mapped to the bus driver.	Add 0x607f in the PDO and map it to the axis.	No
9142	The immediate stop instruction is active.	Check whether the immediate stop instruction has been called.	No
9143	The immediate stop instruction is called repeatedly.	Check whether the immediate stop instruction is called repeatedly.	No
9144	The limit is reached during jogging.	Check whether the limit is active.	No
9145	Target position exceeded 9999999 The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target position must not exceed this value.	• Check whether the target position is correct. Set the target position again. • Change the gear ratio to ensure that the target position is not greater than 9999999.	Yes
9146	Target velocity exceeded 9999999	• Check whether the target velocity is correct. Set the target velocity again.	Yes

Fault code	Fault information	Solution	Stop triggered
	The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target velocity must not exceed this value.	• Change the gear ratio to ensure that the target velocity is not greater than 9999999.	
9147	Target acceleration exceeded 9999999 The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target acceleration must not exceed this value.	• Check whether the target acceleration is correct. Set the target acceleration again. • Change the gear ratio to ensure that the target acceleration is not greater than 9999999.	Yes
9148	Target deceleration exceeded 9999999 The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target deceleration must not exceed this value.	• Check whether the target deceleration is correct. Set the target deceleration again. • Change the gear ratio to ensure that the target deceleration is not greater than 9999999.	Yes
9149	Execution of single-axis motion instructions is not allowed because the axis is in sync control mode.	Check whether the axis is executing the interpolation instruction. Execution of single-axis motion instructions is not allowed during interpolation.	No
9501	The servo drive is faulty.	• For the EtherCAT bus axis, check the slave fault type by using 0x603f and then eliminate the fault. • For the local axis, check the local axis fault list to troubleshoot the fault. • For the local axis, check the local axis fault list to troubleshoot the fault.	Yes
9502	The drive is disabled.	The drive status word 0x6041 switches to the disabled state during motion.	Yes
9503	The drive has reached the limit.	Check whether the limit is configured and whether the limit signal is active.	Yes
9504	Reserved	-	-
9505	The axis fails to write to 0x6060.	Check for interference in network communication.	Yes

Fault code	Fault information	Solution	Stop triggered
9506	Reserved	-	-
9507	Reserved	-	-
9508	Homing fault	• Identify the cause of the drive homing failure. • Check whether homing timed out. • Check whether the limit signal is incorrect.	Yes
9509	Loss of precision occurs.	Check whether the floating-point data of the instruction falls beyond the single-precision floating-point number range.	Yes
9510	The following error is too large. The difference between the set position and the feedback position exceeds the set threshold.	• Check whether acceleration is too large. • Check whether the set following error is too small.	Yes
9511	Reserved	-	-
9512	Drive communication failed during operation.	• Check whether the drive works properly. • Check whether the network cable is properly connected. • Check for interference in communication.	Yes
9513	Homing failed due to a drive fault	Check the fault code of the drive to eliminate the fault.	Yes
9514	Homing failed because the homing offset exceeded 32 bits.	Check whether the homing offset multiplied by the gear ratio exceeds 32 bits; if yes, change the gear ratio.	Yes
9515	Homing failed due to loss of the slave	Contact Inovance for technical support.	Yes
9516	Homing failed because the SDO failed to write to object dictionary 0x607C.	• Check whether the drive supports 0x607C. • Check the network communication quality.	Yes
9517	Homing failed because SDO failed to write 6 into object dictionary 0x6060.	• Set 0x6060 in the PDO. • Check the network communication quality.	Yes
9518	Homing failed because the SDO failed to read object dictionary 0x6061.	• Set 0x6061 in the PDO. • Check the network communication quality.	Yes

Fault code	Fault information	Solution	Stop triggered
9519	Homing failed because SDO failed to write 8 into object dictionary 0x6060.	• Set 0x6060 in the PDO. • Check the network communication quality.	Yes
9551	State switching failed.	Check for interference in network communication.	Yes
9552	The target velocity is less than 0.	Check whether the target velocity of position instructions is appropriate.	Yes
9601	The axis stops due to an error of the absolute positioning instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9602	The axis stops due to an error of the relative positioning instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9603	The axis stops due to an error of the velocity control instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9604	The axis stops due to an error of the jogging instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9605	Reserved	-	-
9606	The axis stops due to an error of the buffer control instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9607	The axis stops due to an error of the interrupt positioning instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9608	The axis stops due to an error of the stop instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9609	The axis stops due to an error of the torque control instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9610	The axis stops due to an error of the halt instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9616			Yes

Fault code	Fault information	Solution	Stop triggered
	The axis stops due to parameter exception of the MC_MoveAbsRelPP instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	
9617	The axis stops because the SDO failed to write to object dictionary 0x6060.	• Set 0x6060 in the PDO. • Check the network communication quality.	Yes
9618	The axis stops because the SDO failed to read object dictionary 0x6061.	• Set 0x6061 in the PDO. • Check the network communication quality.	Yes
9619	Target movement value pulses exceeding range in profile pulse mode	Reduce the pulses per revolution of the motor/ encoder or modify the gear ratio.	Yes
9620	The axis stops due to parameter exception of the MC_MoveVelocityPV instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9621	The axis stops due to an error reported by the MC_JogP instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9622	The axis stops due to parameter exception of the MC_MoveBufferPP instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9623	The axis stops due to parameter exception of the MC_TorqueControlPT instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9159	The profile velocity 0x6081 is not mapped to the bus driver.	Add 0x6081 in the PDO and map it to the axis.	No
9160	The profile acceleration 0x6083 is not mapped to the bus driver.	Add 0x6083 in the PDO and map it to the axis.	No
9161	The profile deceleration 0x6084 is not mapped to the bus driver.	Add 0x6084 in the PDO and map it to the axis.	No
9162	The target velocity of the MC_JogP instruction in the jog mode exceeds the maximum jogging velocity.	Check whether the target velocity of the MC_JogP instruction in the jog mode exceeds the maximum jogging velocity.	Yes
9163	The forward and reverse motion flags of the MC_JogP instruction are both active.	Check whether the forward and reverse motion flags of the MC_JogP instruction are both active.	Yes

Fault code	Fault information	Solution	Stop triggered
9164	The actual feedback speed velocity 0x606C is not mapped to the bus driver.	Add 0x606C in the PDO and map it to the axis.	No
9165	The target action is cached in the current buffer.	After the target action in the buffer is executed, trigger the new target action again.	Yes
9166	The torque slope 0x6087 is not mapped to the bus driver.	Add 0x6087 in the PDO and map it to the axis.	Yes
9167	The velocity limit of the profile torque instruction exceeds the maximum velocity specified on the axis configuration interface.	Check whether the velocity limit of the profile torque instruction exceeds the maximum velocity specified on the axis configuration interface.	Yes
9800	Failed to obtain the number of axes.	Change the software tool version.	Yes
9801	The number of axes is greater than 32.	Reduce the number of axes since the H5U supports at most 32 axes.	Yes
9802	Failed to request the memory.	Check whether the memory runs out.	Yes
9803	Failed to obtain parameters.	Check whether the board and the software tool version match.	Yes
9804	Failed to obtain the slave.	No	Yes

NOTE

When a motion control axis is faulty, refer to the preceding fault code list for fault information.

5.13.2 Instructions of Electronic Cam and Electronic Gear

5.13.2.1 List of Instruction

The following table lists the electronic cam instructions.

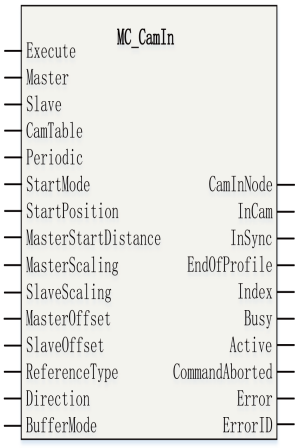
Instruction category	Instruction name	Description
Electronic cam instructions	MC_CamIn	Start cam operation
	MC_CamOut	End cam operation
	MC_GetCamTablePhase	Obtain cam table phase

Instruction category	Instruction name	Description
	MC_GetCamTableDistance	Obtain cam table displacement
	MC_DigitalCamSwitch	Electronic cam tappet control
	MC_GearIn	Start gear operation
	MC_GearOut	End gear operation
	MC_Phasing	Master axis phase shifting
	MC_SaveCamTable	Save cam table
	MC_GenerateCamTable	Update cam table
	MC_GearInPos	Start the gear operation at the specified position

5.13.2.2 MC_CamIn

MC_CamIn -- Start cam operation

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_CamIn	Start cam operation		MC_CamIn(Execute := ???, Master := ???, Slave := ???, CamTable := ???, Periodic := , StartMode := , StartPosition := , MasterStartDistance := , MasterScaling := , SlaveScaling := , MasterOffset := , SlaveOffset := , ReferenceType := , Direction := , BufferMode := , CamInNode => , InCam => , InSync => , EndOfProfile => , Index => , Busy => , Active => , CommandAborted => ,

Instruction	Name	LD Expression	LiteST Expression
			Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_CamIn: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Master	Master axis Bus servo axis, local pulse axis, bus encoder axis, or local encoder axis	No	-	-	_sMCAXIS_INFO _sENC_AXIS _sENC_EXT_AXIS _sMasterAxis
S2	Slave	Slave axis Bus servo axis or local pulse axis	No	-	-	_sMCAXIS_INFO
S3	CamTable	Cam table	No	-	-	_sMC_CAMTABLE
S4	Periodic	Periodic mode 0: Periodic Others: Periodic for a specified number of cycles	0	0	0 to 32,767	INT
S5	StartMode	Mode for specifying MasterStartDistance 0: Absolute mode 1: Relative mode 2: Start immediately	Yes	0	0 to 2	INT
S6	StartPosition	Start position of the cam table	Yes	0	Positive number 0 Negative number	REAL

S7	MasterStartDistance	Master following distance	Yes	0	Positive number 0 Negative number	REAL
S8	MasterScaling	Master coefficient	Yes	1	Positive number	REAL
S9	SlaveScaling	Slave coefficient	Yes	1	Positive number	REAL
S10	MasterOffset	Master offset	Yes	0	Positive number 0 Negative number	REAL
S11	SlaveOffset	Slave offset	Yes	0	Positive number 0 Negative number	REAL
S12	ReferenceType	Position type 0: Instruction position of the previous cycle 1: Instruction position of the current cycle[1] 2: Feedback position of the current cycle	Yes	0	0 to 2	INT
S13	Direction	Direction 0: Forward 1: Reverse 2: Not specified	Yes	0	0 to 2	INT
S14	BufferMode	Buffer mode 0: Wait until the previous one is completed 1: Data buffer	Yes	0	-	INT

D1	CamInNode	Cam engagement variable	Yes	-	-	_sMC_CAMIN
D2	InCam	Cam motion	Yes	OFF	ON OFF	BOOL
D3	InSync	Synchronizing	Yes	OFF	ON OFF	BOOL
D4	EndOfProfile	End of cam cycle	Yes	OFF	ON OFF	BOOL
D5	Index	Index	Yes	0	1 to 360	INT
D6	Busy	Executing	Yes	OFF	ON OFF	BOOL
D7	Active	Controlling	Yes	OFF	ON OFF	BOOL
D8	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D9	Error	Error	Yes	OFF	ON OFF	BOOL
D10	ErrorID	Error code	Yes	0	-	INT

NOTE

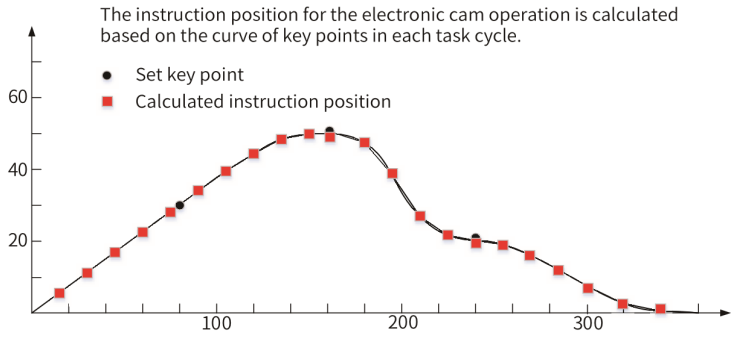
[1]: When the position set under the same task is selected, make sure that the axis ID of the master axis is smaller than that of the slave axis.

Relative Cam Table

The phase and displacement of the cam table are specified as relative quantities from a start point of 0.0. In each EtherCAT cycle, the cam calculation unit calculates the displacement of the slave axis corresponding to the phase of the master axis according to the selected cam curve type.

Cam table

	Phase	Displacement
Start point	0	0
	80	30
	160	50
	240	20
End point	360	0



Instruction Execution Condition

Instruction Execution Condition

You can execute this instruction while the master axis is stopped, during position control, velocity control, or synchronized control.

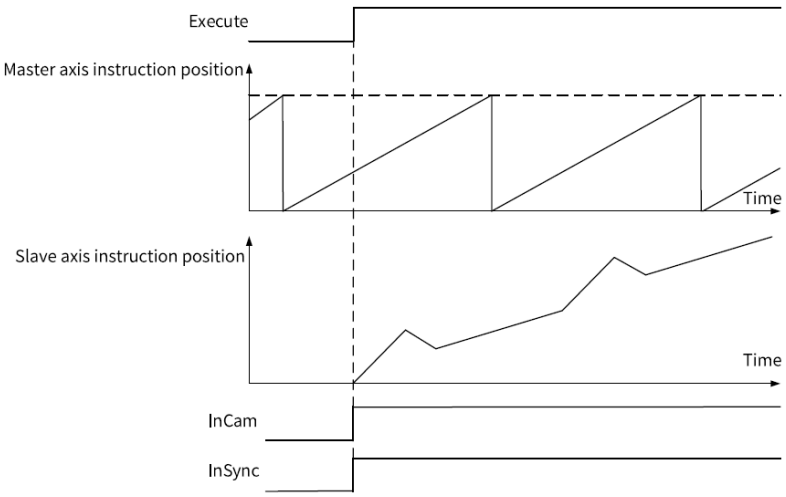
You can execute this instruction while the slave axis is in StandStill, DiscreteMotion, ContinuousMotion, or SynchronizedMotion (non-axis-group motion) state.

Software Limits

If the slave axis exceeds the software limit during cam operation, an error occurs and the axis stops running.

Starting Cam Operation

- StartMode = 2 (Start immediately)
  After the instruction is executed, when the master axis moves, the cam operation is performed immediately and the InCam and InSync signals are set to TRUE. The current position of the master axis is phase 0 of the cam, and that of the slave axis is displacement 0 of the cam.



> **NOTE**
>
> InCam and InSync signal outputs are available only after the master axis operates.

- StartMode = 0 or 1 (Start from specified position)
  After the instruction is executed, the master axis must reach the StartPosition (start position of the cam table).

After the master axis passes the StartPosition (start position of the cam table), the start point in the cam table is executed and the output variable InCam (cam motion) changes to ON.

The phases and displacements in the cam table are specified as relative quantities from zero. The absolute position of each axis at each phase is the relative value from the absolute position of the axis at the start point of the cam table. For example, if the count mode of the master axis is 0 to 360 in rotation mode and the cam table is shown in the following figure, the StartPosition (start position of the cam table) is 50. The absolute position of the master axis is the phase added to the StartPosition, as shown in the following cam table. The absolute position of the slave axis is the displacement from the cam table added to the absolute position of the slave axis at the start point of the cam table.

Cam table

	Phase	Displacement
Start point	0	0
	80	30
	120	50
	240	20
End point	360	0

StartPosition=50

Absolute positions of axes

Master axis	Slave axis
50	0+Absolute position of slave axis at starting point of cam table
130	30+Absolute position of slave axis at starting point of cam table
170	50+Absolute position of slave axis at starting point of cam table
290	20+Absolute position of slave axis at starting point of cam table
50	0+Absolute position of slave axis at starting point of cam table

When the MasterStartDistance (master following distance) is then passed, the cam operation of the slave axis starts and the output variable InSync (synchronizing) changes to ON.

The MasterStartDistance (master following distance) is specified either as an absolute position (StartMode = 0), or as a relative distance (StartMode = 1) from the StartPosition (start position of the cam table).

The cam table settings are as follows:

Phase	Displacement
0	0
80	120
120	80
360	140

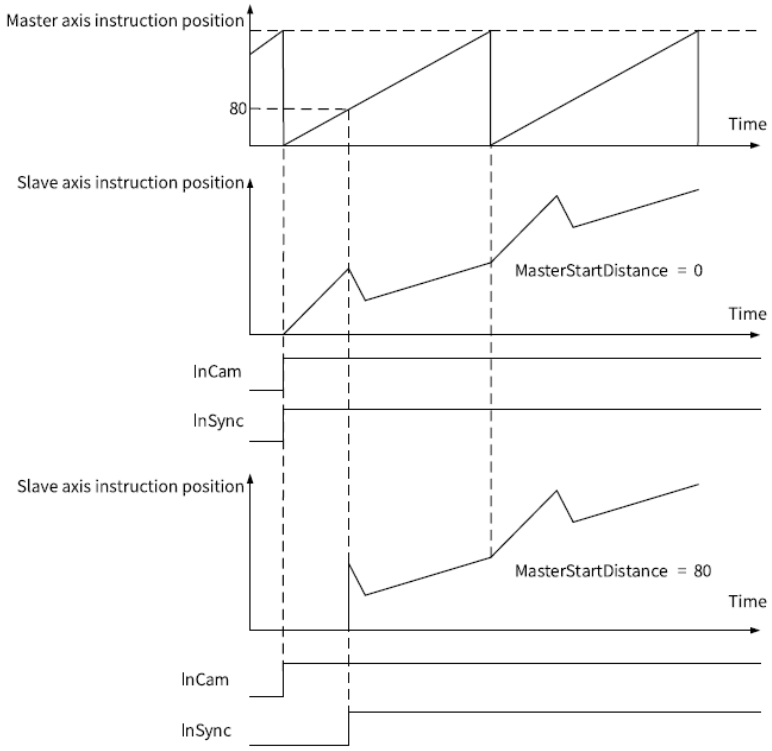
The conditions for starting cam operation are as follows:

Input Variable	Condition 1	Condition 2
Periodic (Periodic mode)	0	0
StartMode (Mode for specifying the start position)	Relative position	Relative position
StartPosition (Start position of the cam table)	0	0
MasterStartDistance (Master following distance)	0	80

Under condition 1, the output variables InCam (cam motion) and InSync (synchronizing) both change to ON and the slave axis starts cam operation when the master axis passes 0.

Under condition 2, the output variable InCam (cam motion) changes to ON when the master axis passes 0. Then, the output variable InSync (synchronizing) changes to ON and the slave axis starts cam operation when the master axis passes 80.

Under condition 2, cam operation starts in the middle of the cam table, so the slave axis will accelerate rapidly.



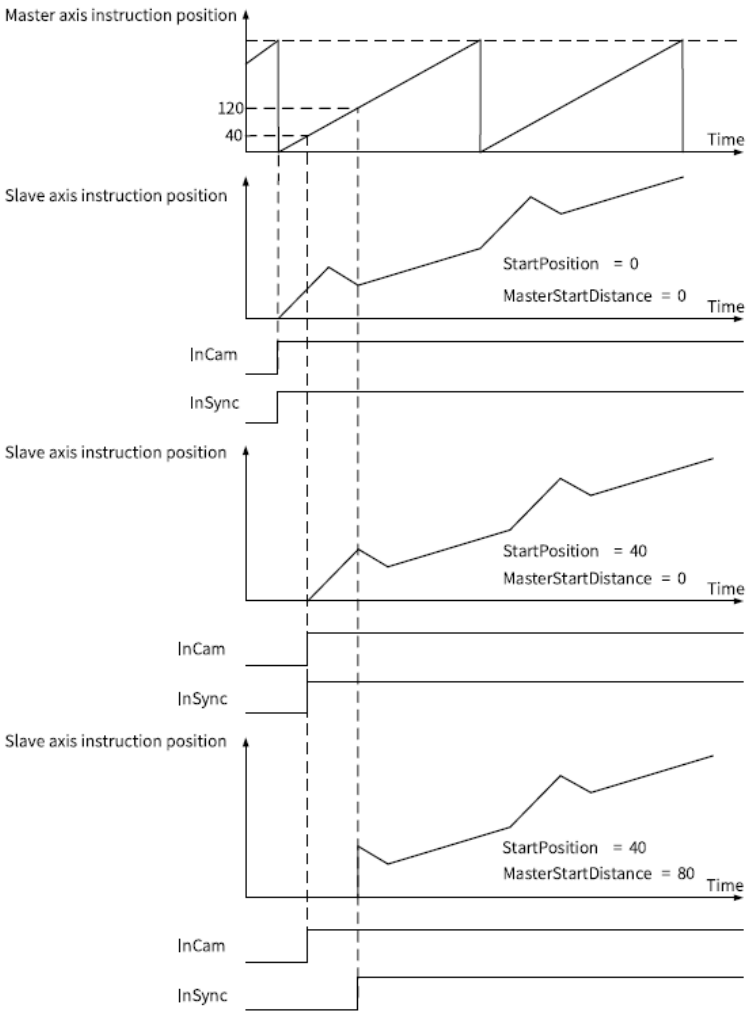
The cam table settings are the same as those in the previous example. The conditions for starting cam operation are modified as follows.

Input Variable	Condition 1	Condition 2	Condition 3
Periodic	0	0	0
StartMode	Relative position	Relative position	Relative position
StartPosition	0	40	40
MasterStartDistance	0	0	80

Under condition 1, the output variables InCam (cam motion) and InSync (synchronizing) both change to ON and the slave axis starts cam operation when the master axis passes 0.

Under condition 2, the output variables InCam (cam motion) and InSync (synchronizing) both change to ON and the slave axis starts cam operation when the master axis passes 40 specified by StartPosition (start position of the cam table).

Under condition 3, the output variable InCam (cam motion) changes to ON when the master axis passes 40. Then, the output variable InSync (synchronizing) changes to ON and the slave axis starts cam operation when the master axis passes 120.



You can use StartMode to specify whether the value specified by MasterStartDistance (master following distance) is treated as an absolute position or a relative position.

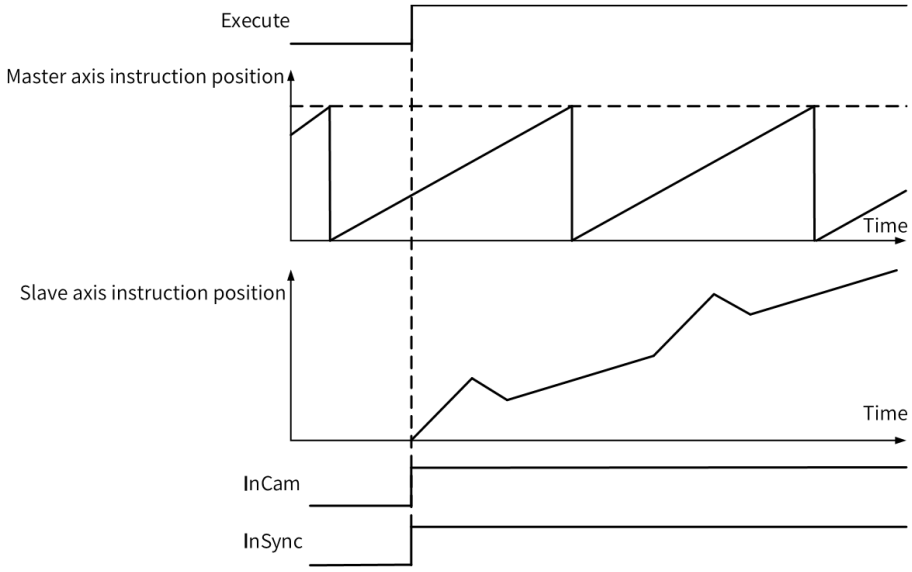
The following section describes the differences in starting cam operation of the slave axis based on differences in StartMode. The cam table settings are the same as those in the previous examples.

The conditions for starting cam operation are as follows.

Input Variable	Condition 1	Condition 2
Periodic	0	0
StartMode	Absolute position	Relative position
StartPosition	40	40
MasterStartDistance	80	80

Under conditions 1 and 2, the output variable InCam (cam motion) changes to ON when the master axis passes 40. Under condition 1, StartMode is set to 0 (absolute position), so the output variable InSync (synchronizing) changes to ON and the slave axis starts cam operation when the master axis passes 80.

Under condition 2, StartMode is set to 1 (relative position), so the output variable InSync (synchronizing) changes to ON and the slave axis starts cam operation when the master axis passes 120 (40 + 80).



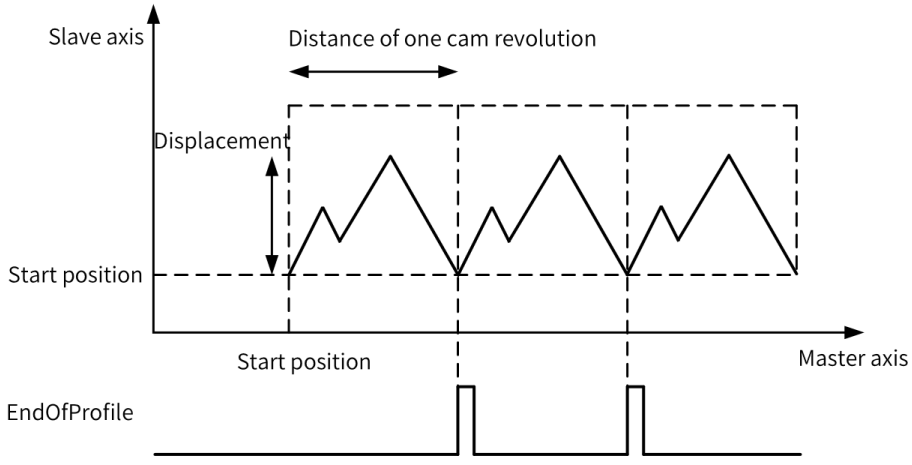
Periodic Mode

When Periodic (periodic mode) is set to 0, the cam motion is repeated from the start to the end point of the cam table. After each cam cycle ends, EndOfProfile is reset to OFF after one PLC scan cycle.

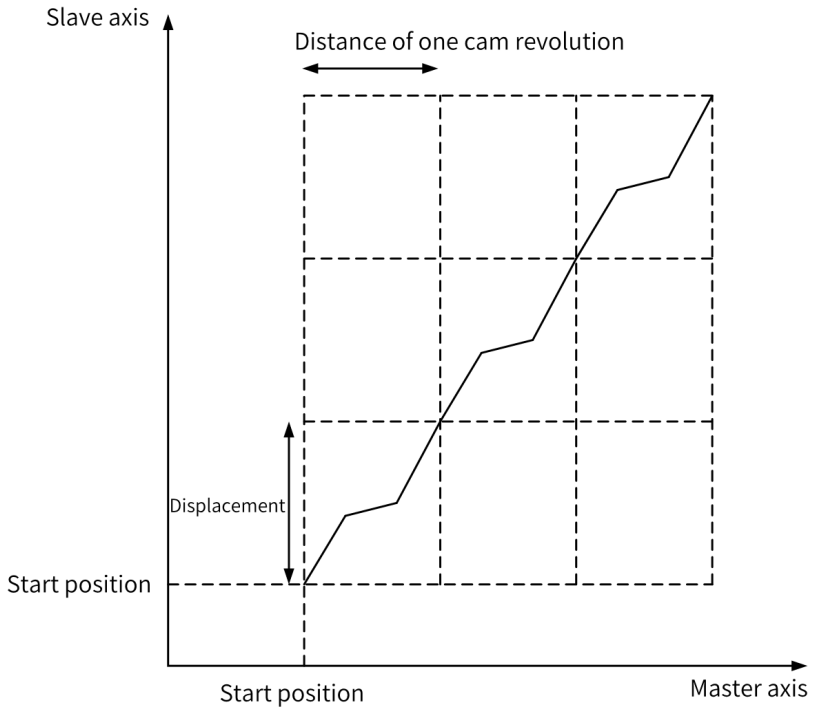
When Periodic (periodic mode) is set to N (N > 0), the cam motion is repeated N times and then ends. After the last cycle, if the Execute input is ON, EndOfProfile is always ON. If the Execute input is OFF, EndOfProfile is reset to OFF after one PLC scan cycle.

If the stroke position of the slave axis is the same at the start and end points of the cam table during the repeating process, the cam operates as a reciprocal cam. If the stroke position of the slave axis differs at the start point and end point, the cam operates as a feeding cam.

- Reciprocal cam operation



- Feeding cam operation



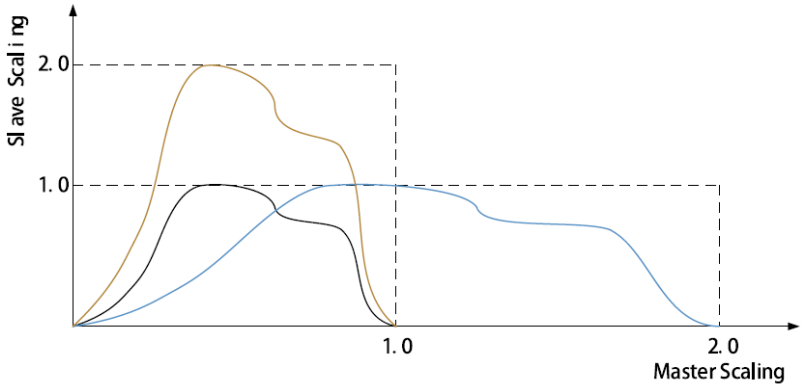
End of Cam Cycle

You can use the MC_CamOut (end cam operation) instruction or MC_Stop (forced stop) instruction to stop cam operation before it is completed.

Scaling Factor

You can specify a scaling factor to scale up or scale down the master axis phase and slave axis displacement of a specified cam table.

You can apply separate ratios to the master and slave axes.

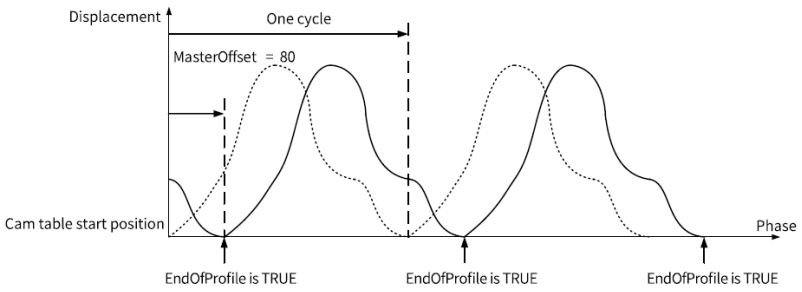


Offset

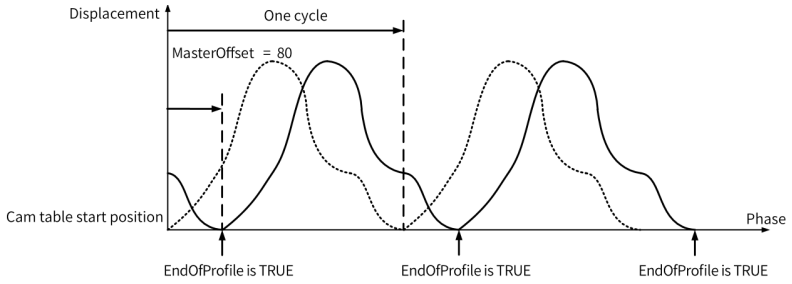
You can shift the phase and displacement by an offset from the specified cam table.

You can specify separate offsets for the master axis phase and slave axis displacement.

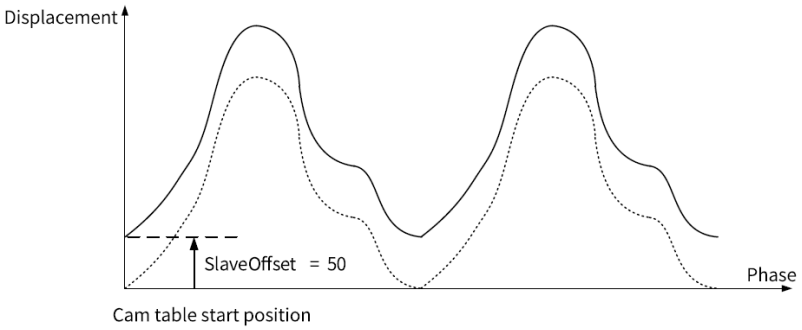
- MasterOffset > 0



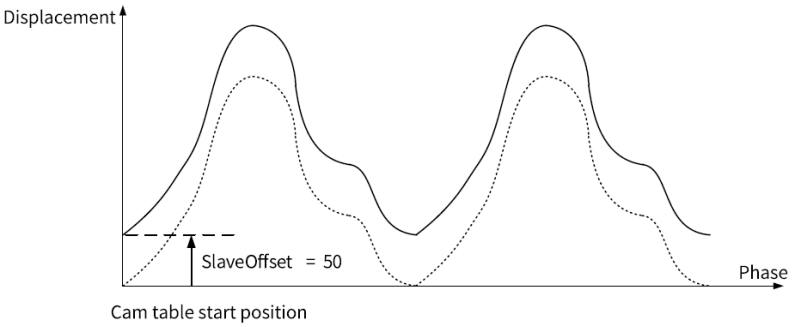
- MasterOffset < 0



- SlaveOffset > 0



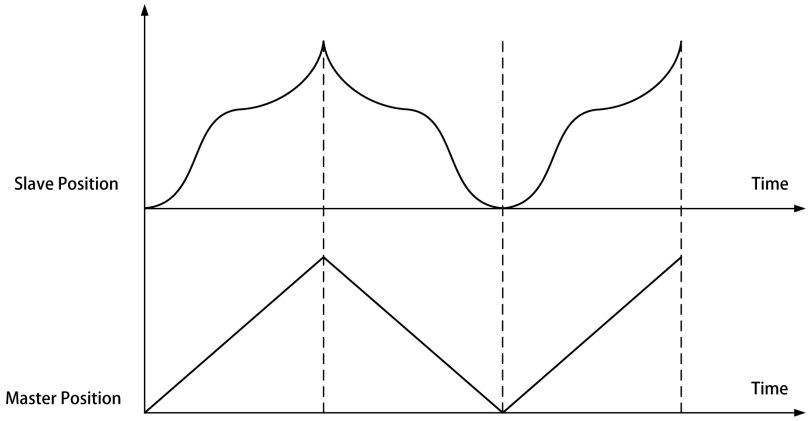
- SlaveOffset < 0



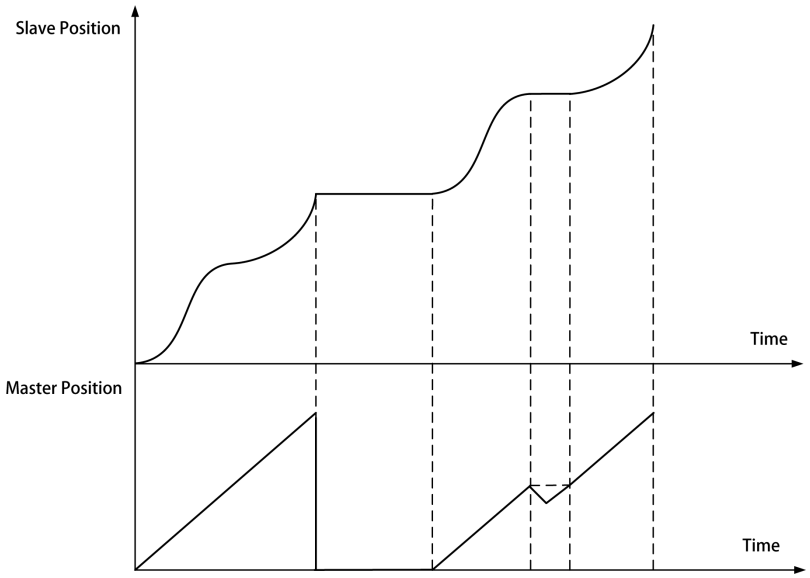
Direction

You can start cam operation for the slave axis only if the travel direction of the master axis matches that specified by Direction.

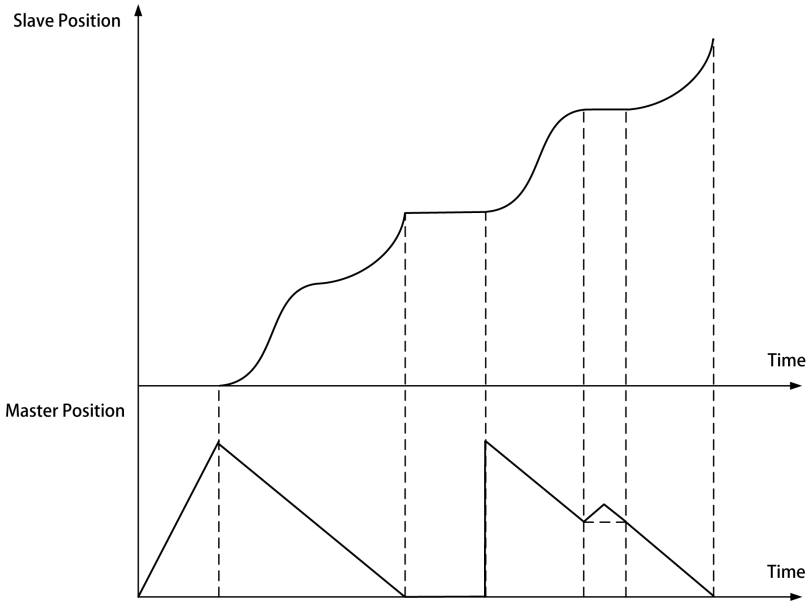
- No direction specified
  Cam operation starts regardless of whether the master axis is traveling in the forward or reverse direction.



- Forward direction
  Cam operation starts when the master axis is moving in the forward direction. In a cam cycle, if the master axis is reversed, the slave axis remains stationary until the master axis returns to its original position. Then, the slave axis continues to follow the master axis.



- Reverse direction
  Cam operation starts when the master axis is moving in the reverse direction. In a cam cycle, if the master axis is reversed, the slave axis remains stationary until the master axis returns to its original position. Then, the slave axis continues to follow the master axis.



Position Type

ReferenceType specifies the data source of the master axis position.

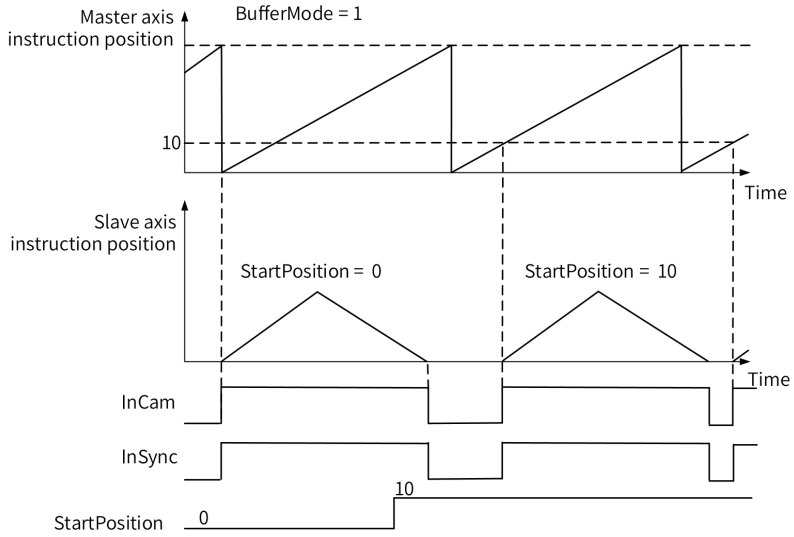
When the master axis is a local encoder axis, this parameter is invalid, and the feedback position of the current cycle is always used.

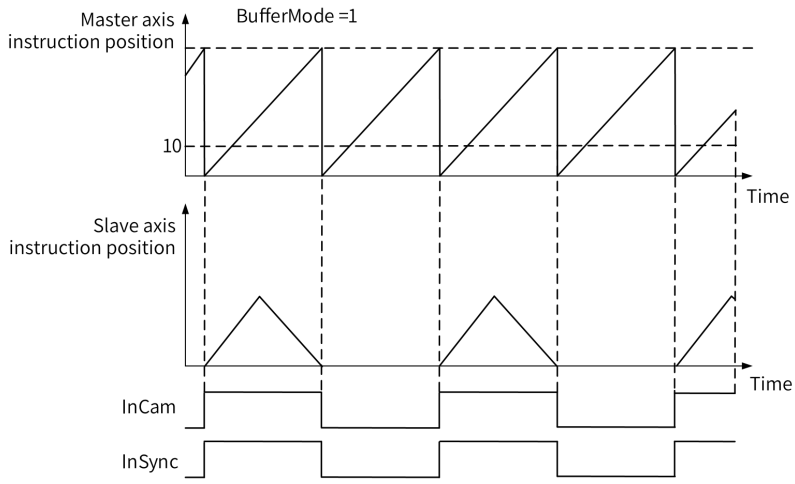
When the master axis is a bus servo axis or local pulse axis, you can use the instruction position of the previous cycle or the current cycle, or the feedback position of the current cycle.

Buffer Mode

When BufferMode is 1, the cam motion enters the data buffer mode.

In data buffer mode, when values of input parameters such as StarPosition and MasterStarDistance are changed in a cam operation cycle, the changes take effect in the next cam cycle.





> **NOTE**
>
> In data buffer mode, if the cam cycle is synchronized with the rotation cycle of the master axis, the cam motion will be triggered at regular intervals.

Re-execution

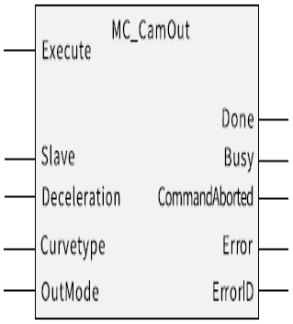
If the MC_CamIn instruction is re-triggered while the Busy signal is still active, parameters including Periodic, MasterScaling, SlaveScaling, RefrenceType, and Direction are buffered and take effect in the next cam cycle.

Multi-execution

If a second MC_CamIn instruction is triggered while the Busy signal of the MC_CamIn instruction is still active, the Busy signal of the second instruction becomes active but the Active signal is inactive. When a cam cycle ends, the first instruction is aborted, the Active signal of the second instruction becomes active, and the parameters (Periodic, MasterScaling, SlaveScaling, RefrenceType, and Direction) of the second instruction take effect.

5.13.2.3 MC_CamOut

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_CamOut	End cam operation		MC_CamOut(Execute := ???, Slave := ???, Deceleration := ???, CurveType := , OutMode := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	MC_CamOut: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Slave	Slave axis Bus servo axis or local output axis	No	-	-	_sMCAXIS_INFO
S2	Deceleration	Deceleration • Positive number: Stop according to deceleration • 0: Stop immediately	No	-	Positive number or 0	REAL
S3	Curvetype	Curve shape 0: T-shaped velocity curve	Yes	0	-	INT

S4	OutMode	Sync end mode • 0: Decelerate to stop • 1: Stop immediately after executing the current cycle	Yes	0	0 to 1	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	1	INT

Function Description

This instruction ends cam operation of the slave axis.

When Execute is set to ON, the MC_CamIn instruction is aborted, and the abortion flag is active. If OutMode is set to 0, the axis decelerates according to Deceleration. After it decelerates to 0, the Done output is active. The slave axis stays in ContinuousMotion state before it stops moving. If OutMode is set to 1, the axis stops immediately after the cam operation of the current cycle is completed. The slave axis stays in SynchronizedMotion state before the cam operation ends.

An error occurs when this instruction is executed on an axis that is not in cam operation.

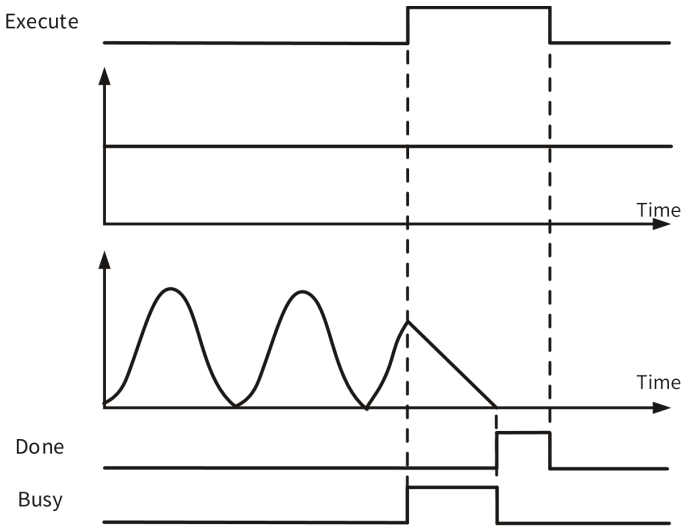
Re-triggering

When the MC_CamOut instruction is re-triggered, the axis stops according to the following rules:

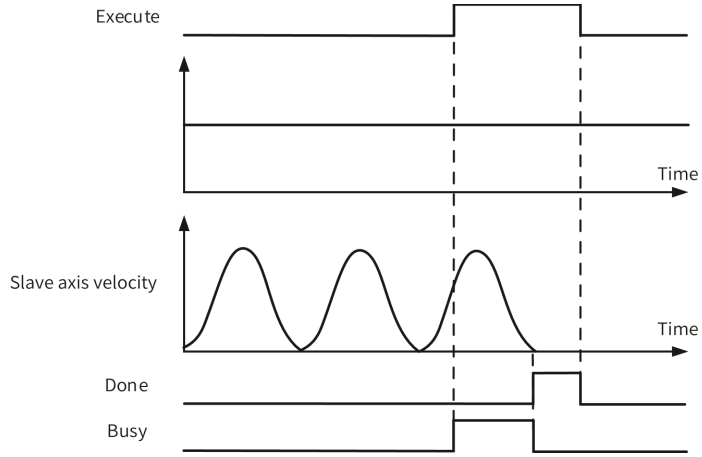
Initial stop mode	New stop mode	Execution result
Decelerate to stop	Stop immediately after executing the current cycle	The instruction reports an error, and the axis decelerates to stop and then enters the StandStill state.
Decelerate to stop	Decelerate to stop	The axis stops according to the new deceleration.
Stop immediately after executing the current cycle	Decelerate to stop	The axis decelerates to stop.
Stop immediately after executing the current cycle	Stop immediately after executing the current cycle	The axis stops after executing the current cycle.

Sequence Diagram

- Decelerate to stop



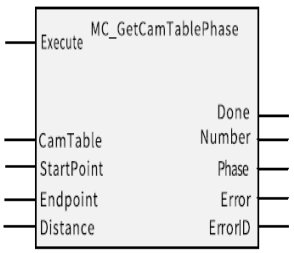
- Stop immediately after executing the current cycle



5.13.2.4 MC_GetCamTablePhase

MC_GetCamTablePhase -- Obtain cam table phase

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_GetCamTablePhase	Obtain cam table phase		MC_GetCamTablePhase(Execute := ???, CamTable := , StartPoint := ???, EndPoint := ???, Distance := ???, Done => , Number => , Phase => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_GetCamTablePhase: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	CamTable	Cam table Reserved	Yes	-	-	_sMC_CAMTABLE
S2	StartPoint	Start point	No	-		_sMC_CAM_NODE
S3	EndPoint	End point	No	-		_sMC_CAM_NODE
S4	Distance	Displacement of the slave axis	No	-	Positive number, negative number, or 0	REAL
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Number	Number of phases -1: Infinite number of identical solutions 0: None > 0: Actual number of phases	Yes	0	-1, 0 to 5	INT
D3	Phase	Obtained phase	Yes	0	Positive number or 0	REAL[6]
D4	Error	Error	Yes	-	ON OFF	BOOL
D5	ErrorID	Error code	Yes	-	1	INT

Function Description

The MC_GetCamTablePhase instruction obtains the phase (Phase) of the master axis according to the displacement (Distance) of the slave axis between two cam key points.

If the cam curve is a straight line and is parallel to the X axis, the Distance specified in the instruction is on the straight line, the instruction output parameter Number outputs -1, and Phase[0] outputs the abscissa of the start point.

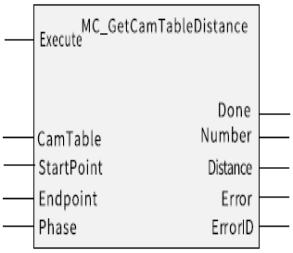
If the cam curve is a quintic curve, there may be multiple solutions. The instruction output parameter Number indicates the number of solutions, and the Phase array stores the specific values obtained.

If there is no solution, the output parameter Number is 0.

5.13.2.5 MC_GetCamTableDistance

MC_GetCamTableDistance: Obtain cam table displacement

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_GetCamTableDistance	Obtain cam table displacement		MC_GetCamTableDistance(Execute := ???, CamTable := , StartPoint := ???, EndPoint := ???, Phase := ???, Done => , Distance => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	MC_GetCamTablePhase: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	CamTable	Cam table Reserved	Yes	-	-	_sMC_CAMTABLE
S2	Start Point	Start point	No	-		_sMC_CAM_NODE
S3	EndPoint	End point	No	-		_sMC_CAM_NODE
S4	Phase	Phase of the master axis	No	-	Positive number, negative number, or 0	REAL
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Distance	Obtained displacement of the slave axis	Yes	0	Positive number, negative number, or 0	REAL
D3	Error	Error	Yes	-	ON	BOOL

					OFF	
D4	ErrorID	Error code	Yes	-	1	INT

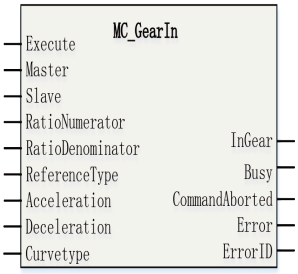
Function Description

This instruction is used to obtain the displacement (Distance) of the slave axis according to the phase (Phase) of the master axis between two cam key points.

5.13.2.6 MC_GearIn

MC_CamIn: Start cam operation

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_GearIn	Start gear operation		MC_GearIn(Execute := ???, Master := ???, Slave := ???, RatioNumerator := , RatioDenominator := , ReferenceType := , Acceleration := , Deceleration := , CurveType := , InGear => , Busy => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows

16-bit instruction	-					
32-bit instruction	MC_GearIn: Continuous execution					
Operands	Name	Description	Empty Allowed	Default	Value range	Data type
S1	Master	Bus servo axis, local pulse axis, bus encoder axis, or local encoder axis	No	-	-	_sMCAXIS_INFO _sENC_AXIS _sENC_EXT_AXIS

						_sMasterAxis
S2	Slave	Slave axis Bus servo axis or local pulse axis	No	-	-	_sMCAXIS_INFO
S3	RatioNumerator	Gear ratio numerator	Yes	1	Positive number Negative number	DINT
S4	RatioDenominator	Gear ratio denominator	Yes	1	Positive number	DINT
S5	ReferenceType	Position type 0: Instruction position of the previous task cycle 1: Instruction position of the current task cycle[1] 2: Feedback position of the current task cycle	Yes	0	0 to 2	INT
S6	Acceleration	Acceleration 0: No acceleration, target speed immediately reached	No	0	0 Positive number	REAL
S7	Deceleration	Deceleration 0: No deceleration	Yes	Acceleration	0 Positive number	REAL
S8	Curvetype	Curve Shape 0: T-shaped acceleration curve	Yes	0	ON OFF	BOOL
D1	InGear	Gear ratio reached	Yes	-	ON OFF	BOOL
D2	Busy	Executing	Yes	-	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	-	ON OFF	BOOL
D4	Error	Error	Yes	-	ON OFF	BOOL

D5	ErrorID	Error code	Yes	-	ON OFF	BOOL

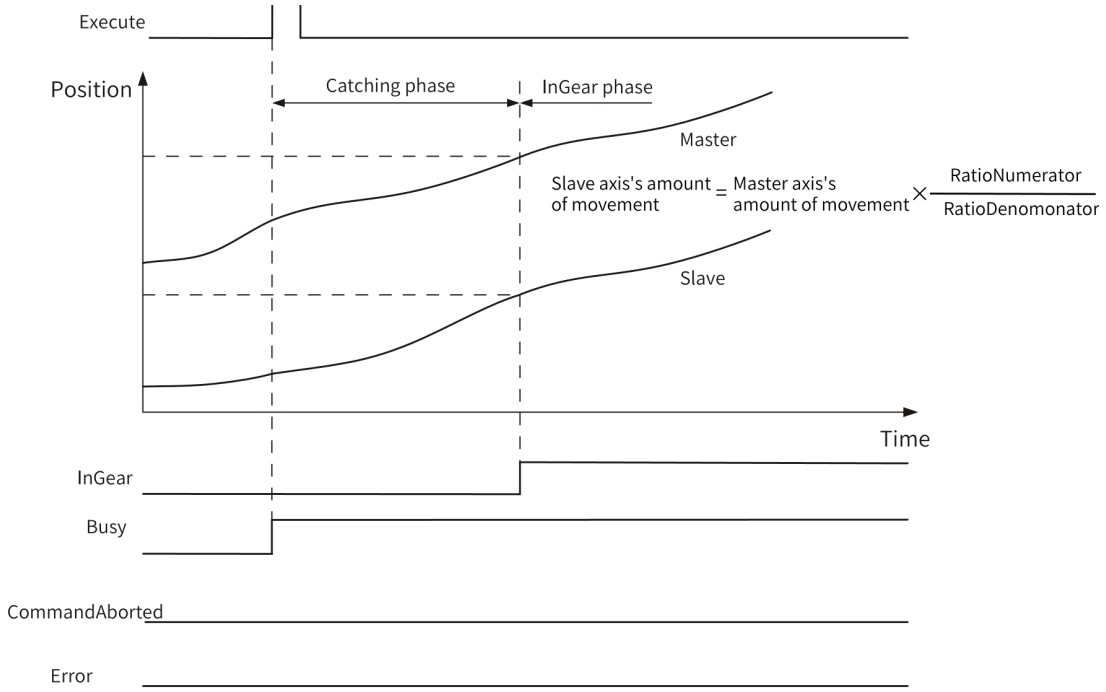
NOTE

[1]: When selecting the set position under the same task, make sure that the axis ID of the master axis is smaller than that of the slave axis.

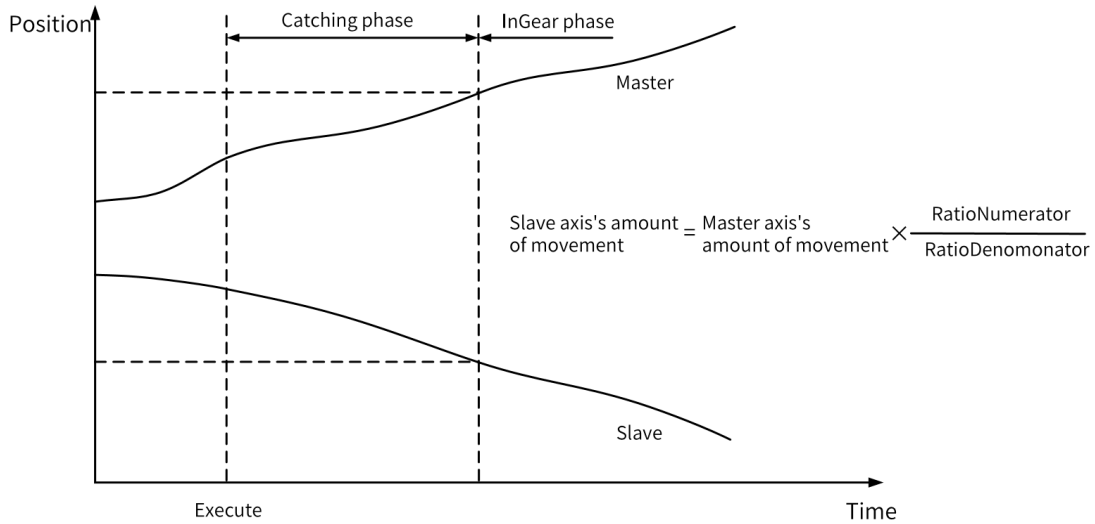
After coming into action, the slave axis uses the velocity obtained by multiplying the master axis velocity by the gear ratio as the target velocity to perform the acceleration and deceleration actions.

The phase is called Catching phase before the axis reaches the target position, and the InGear phase after the axis reaches the target position.

If the gear ratio is positive, the slave axis moves in the same direction as the master axis.

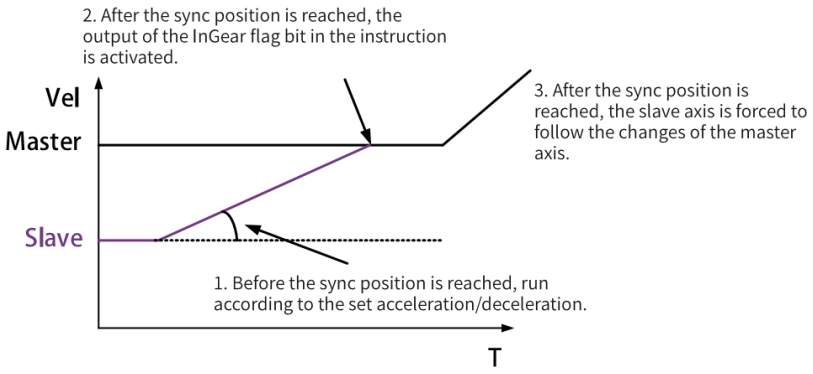


If the gear ratio is negative, the slave axis moves in the opposite direction from the master axis.

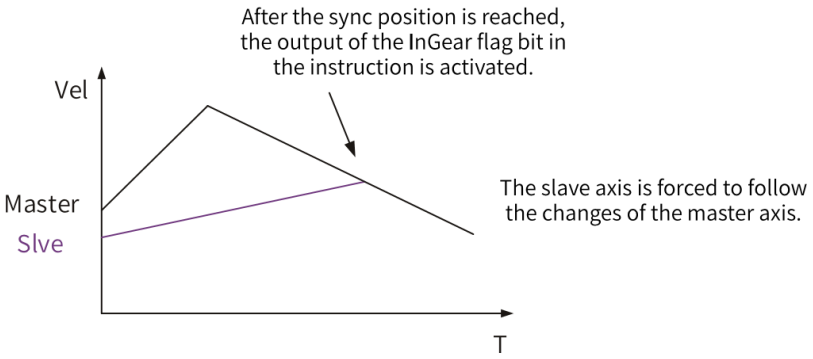


Before reaching synchronization, the slave axis moves at the set acceleration (deceleration). When the slave axis velocity is equal to the master axis velocity multiplied by the gear ratio, the gear is considered to be engaged. After that, the slave axis completely follows the changes of the master axis.

Scenario 1: Before synchronization, the master axis maintains a uniform motion (triggering the MC_GearIn instruction at the gear ratio of 1:1).



Scenario 2: Before synchronization, the master axis performs a variable motion (triggering the MC_GearIn instruction at the gear ratio of 1:1).



Filtering Function

During use, in order to reduce the velocity fluctuation of the master axis caused by the velocity fluctuation of the slave axis, you can adjust the velocity filtering coefficient of the master axis by setting the system variable fFilter parameter of the slave axis. The calculation formula is as follows:

MstVel = fFilter[0]*MstVel[0]+fFilter[1]*MstVel[1]+fFilter[2]*MstVel[2];

Where, MstVel is the synthetic velocity. fFilter is the filtering parameter, and the sum of three must be equal to 1. MstVel indicates the actual master axis velocity in the current cycle, previous cycle, and cycle before last, respectively.

Re-execution

When the MC_GearIn instruction is triggered again while the Busy signal of this instruction is still valid, the master axis velocity will be recalculated based on the gear ratio numerator and denominator. The slave axis will follow the calculation result and determine whether the InGear flag is set.

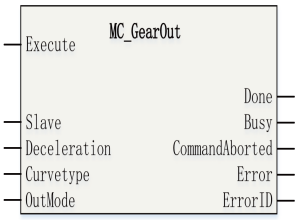
Multi-execution

When the MC_GearIn instruction is triggered again while the Busy signal of this instruction is still valid, the Busy signal of the second instruction is valid, interrupting the first instruction. At the same time, the master axis velocity will be recalculated based on the gear ratio numerator and denominator. The slave axis will follow the calculation result and determine whether the InGear flag is set.

5.13.2.7 MC_GearOut

MC_GearOut -- Cancel gear operation

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_GearOut	Cancel gear operation		MC_GearOut(Execute := ???, Slave := ???, Deceleration := ???, CurveType := , OutMode := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_Gearout: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Slave	Slave axis Bus servo axis or local pulse axis	No	-	-	_sMCAXIS_INFO
S2	Deceleration	Deceleration rate	No	-	Positive number or 0	REAL
S3	Curvetype	Curve type 0: T-shaped velocity curve	Yes	0	0	INT
S4	OutMode	Clear sync mode 0: Decelerate to stop	Yes	0	0	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL

D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	-	INT

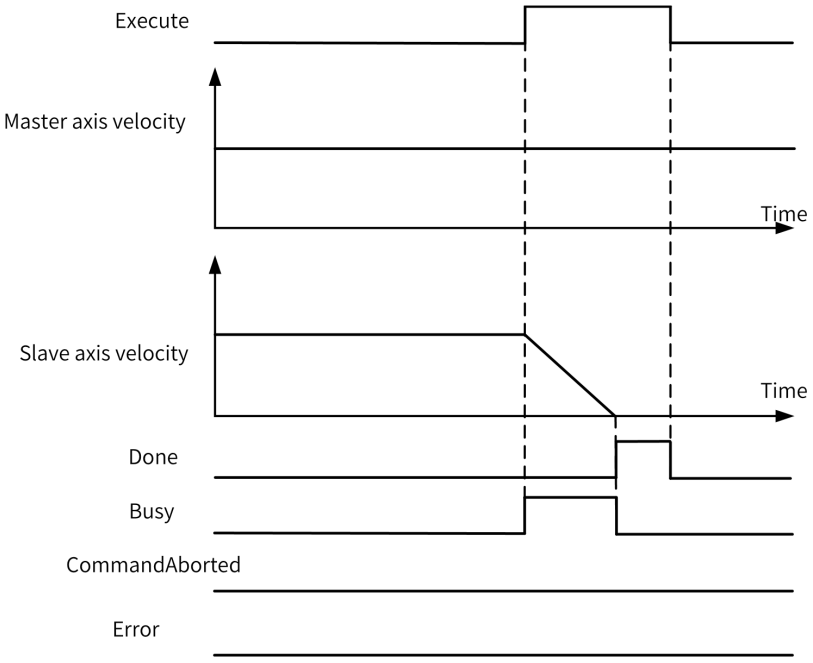
Function Description

The MC_GearOut instruction aborts execution of the MC_GearIn (start gear operation) instruction for the axis specified by Slave at the deceleration rate specified by Deceleration.

This instruction does not affect the MC_GearIn (start gear operation) operation of the master axis.

Timing Diagram

The axis decelerates to stop.

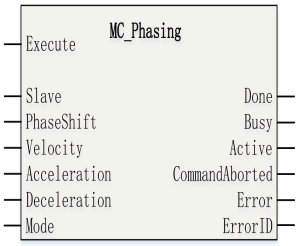


5.13.2.8 MC_Phasing

MC_Phasing: Master axis phase shifting

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_Phasing	Master axis phase shifting		MC_Phasing(Execute := ???, Slave := ???, PhaseShift := ???,

Instruction	Name	LD Expression	LiteST Expression
			Velocity := ???, Acceleration := ???, Deceleration := , Mode := , Done => , Busy => , Active => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	MC_Phasing: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data Format
S1	Slave	Slave axis Bus servo axis or local pulse axis	No	-	-	_sMCAXIS_INFO
S2	Phase Shift	Phase compensation	No	-	Positive number 0 Negative number	REAL
S3	Velocity	Target speed	No	-	Positive number	REAL
S4	Acceleration	Acceleration	No	-	Positive number	REAL
S5	Deceleration	Deceleration	Yes	Acceleration	Positive number	REAL
S6	Mode	Mode • 0: Reserved • 1: Pause when the velocity of the master axis is 0	Yes	0	0 to 1	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL

D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	Active	Executing instruction	Yes	OFF	ON OFF	BOOL
D4	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D5	Error	Error	Yes	OFF	ON OFF	BOOL
D6	ErrorID	Error code	Yes	0	ON OFF	INT

Function Description

If the MC_Phasing instruction is executed when single-axis synchronized control is in progress, the phase of the master axis is shifted according to the settings of PhaseShift (phase shift), Velocity (target velocity), Acceleration (acceleration), and Deceleration (deceleration).

- When working with cam operation, this instruction can be called only after the MC_CamIn instruction is called. When InSync of the MC_CamIn instruction is OFF, the MC_Phasing instruction is in the buffered state, in which the Busy signal is active but the Active signal output is inactive. When InSync of the MC_CamIn instruction becomes ON, the cam is fully engage. At this time, the Active signal output of the MC_Phasing instruction becomes active, and phase shifting starts.
- When working with gear operation, this instruction can be called only after the MC_GearIn instruction is called. The MC_GearIn instruction is triggered first to establish a gear relationship between the master and slave axes. After the slave axis enters the SynchronizedMotion state, the MC_Phasing instruction is triggered and starts to perform the corresponding shifting operation.

During execution, the set position (feedback position) of the master axis does not change, and the relative shift compensated for the set position (feedback position) is taken as the phase of the master axis. The slave axis is synchronized to the shifted master axis phase.

The Done signal changes to ON when the PhaseShift (phase shift) is reached.

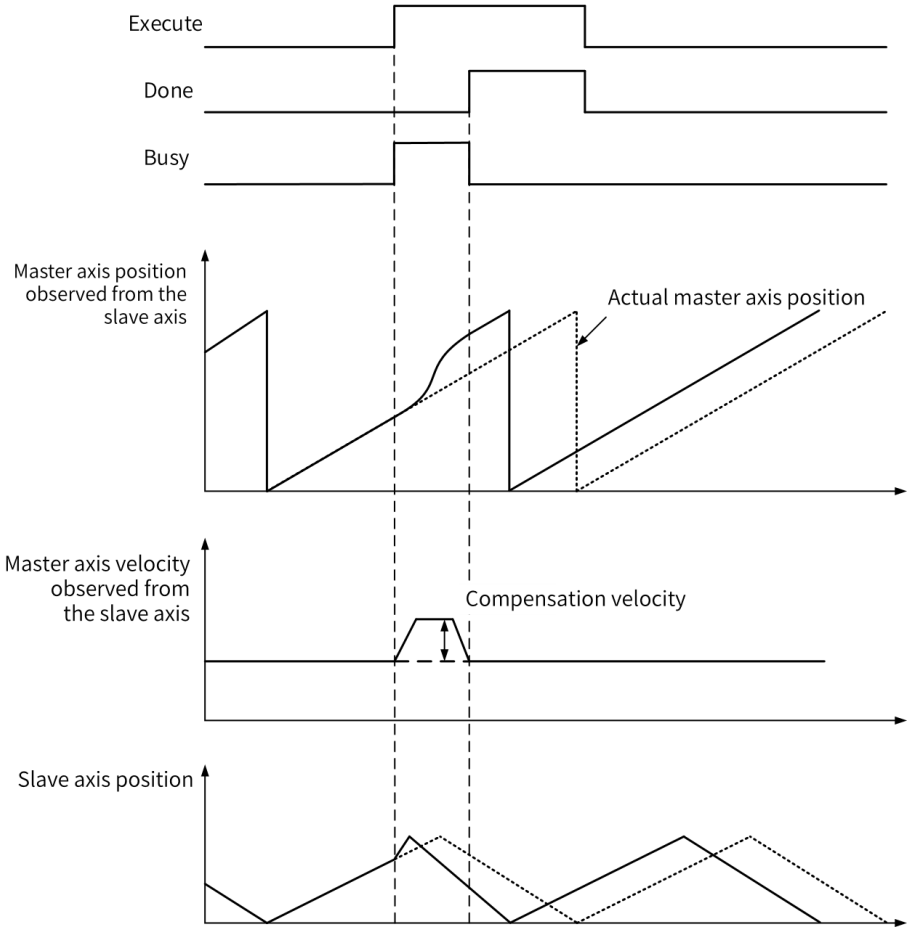
Shifting ends when execution of the synchronized control instruction is completed. If a synchronized control instruction is executed again, the previous amount of shift is not affected.

You can shift the phase of the master axis for the following synchronized control instructions: MC_CamIn (start cam operation) and MC_GearIn (start gear operation).

Mode Selection

When Mode is set to 1, if the master axis stops running (the velocity of the master axis is 0), phase shifting automatically stops. When the master axis starts running again, phase shifting continues from the original position where it was suspended.

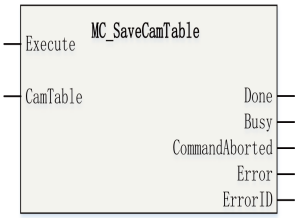
Sequence Diagram



5.13.2.9 MC_SaveCamTable

MC_SaveCamTable -- Save cam table

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_SaveCamTable	Save cam table		MC_SaveCamTable(Execute := ???, CamTable := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-
32-bit Instruction	MC_SaveCamTable: Continuous execution

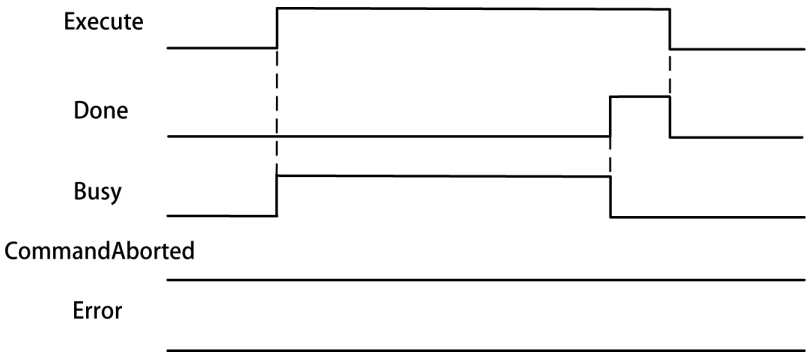
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	CamTable	Cam table	No	-		_sMC_CAMTABLE
D1	Done	Completion flag	Yes	-	ON OFF	BOOL
D2	Busy	Executing	Yes		ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes		ON OFF	BOOL
D4	Error	Error	Yes		ON OFF	BOOL
D5	ErrorID	Error code	Yes		ON OFF	INT

Function Description

The MC_SaveCamTable instruction saves the cam table specified by CamTable to non-volatile memory on the rising edge of the Execute input.

Do not turn off the power supply of the controller during execution of this instruction. Otherwise, data saving may fail, which results in cam data loss.

Timing Diagram

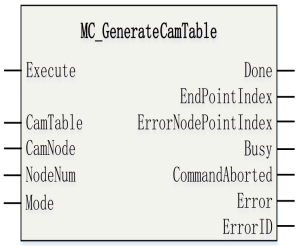


5.13.2.10 MC_GenerateCamTable

MC_GenerateCamTable -- Update cam table

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_GenerateCamTable	Update cam table		MC_GenerateCamTable(Execute := ???,

Instruction	Name	LD Expression	LiteST Expression
			CamTable := ???, CamNode := , NodeNum := , Mode := , Done => , EndPointIndex => , ErrorNodePointIndex => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_GenerateCamTable: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	CamTable	Cam table	No	-		_sMC_CAMTABLE
S2	CamNode	Cam node array If it is left empty, the original cam node array is used. Note: For details, see "System Variables > Cam Node" in the *H5U and Easy Series Programmable Logic Controllers Programming and Application Guide*.	Yes	-		_sMC_CAM_NODE
S3	NodeNum	Number of cam nodes If it is left empty, the original cam node quantity is used.	Yes	-	2 to 361	INT
S4	Mode	Effective mode 0: Effective upon the next cam cycle Others: Reserved	Yes	0	0	INT
D1	Done	Completion flag	Yes	-	ON OFF	BOOL
D2		End point index	Yes	0		INT

	EndPointIndex				0 to 360	
D3	ErrorNodePointIndex	Error node number	Yes	0	0 to 360	INT
D4	Busy	Executing	Yes	OFF	ON OFF	BOOL
D5	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	OFF	ON OFF	INT

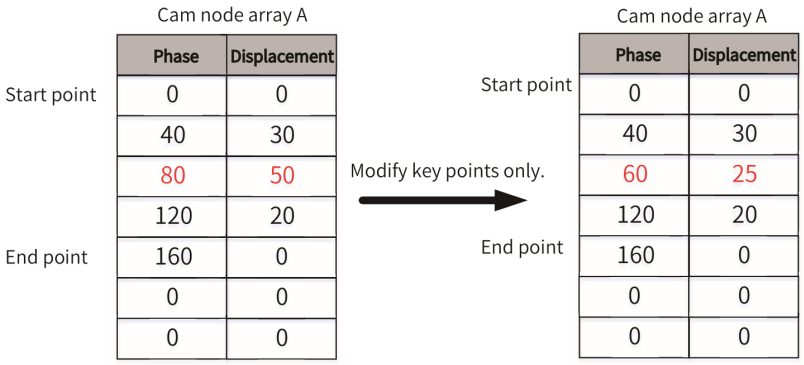
Function Description

The MC_GenerateCamTable instruction calculates cam data based on input variables CamNode and NodeNum on the rising edge of Execute and updates the data to the cam table specified by CamTable. The update takes effect upon the next cam cycle.

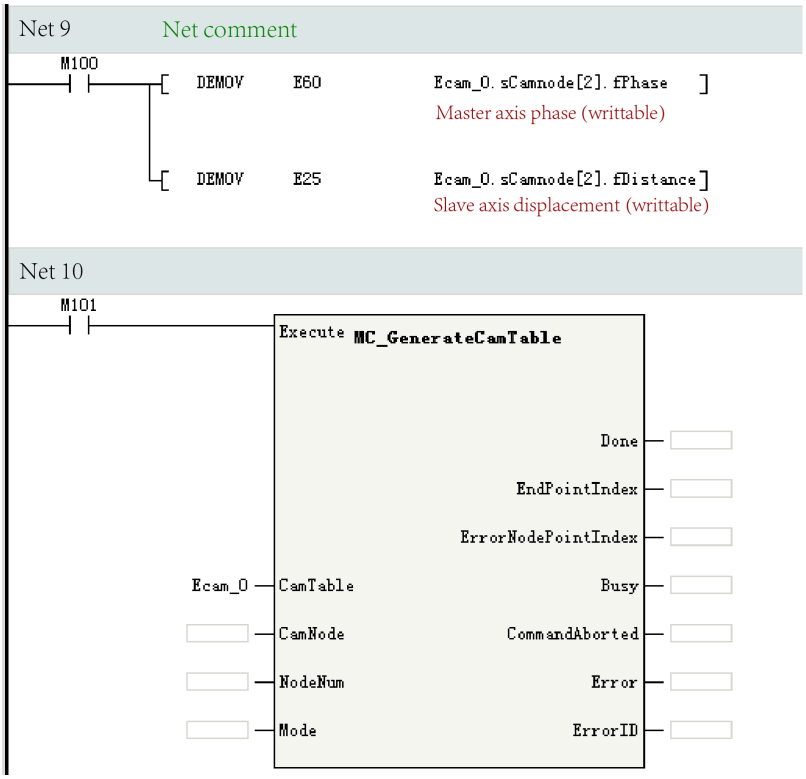
Function of CamNode

CamNode specifies whether to use a new cam node array. When it is empty, the original cam table node array specified by CamTable is adopted. When it is not empty, the cam node array specified by CamNode is adopted.

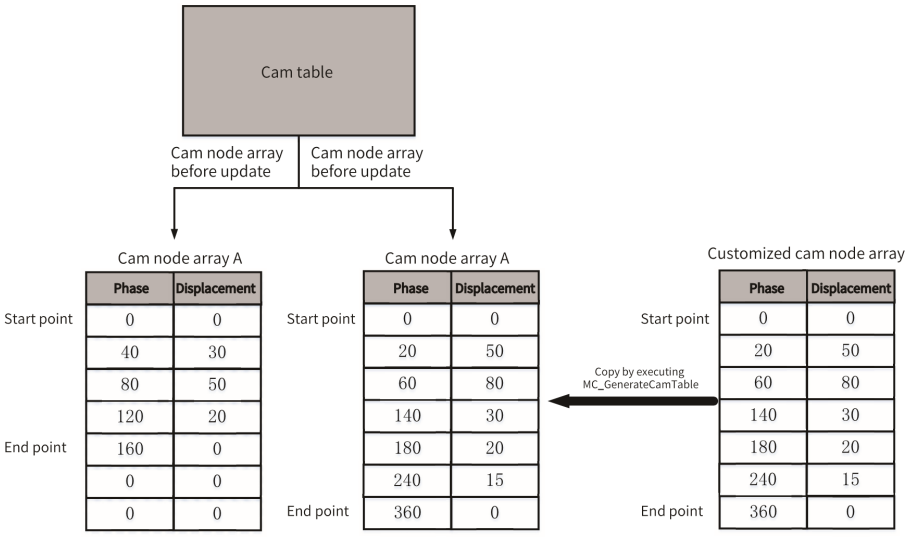
- CamNode is empty.
  You can modify the value of the cam node array in the cam table by using the PLC program and make the modification take effect in the next cam cycle by executing the MC_GenerateCamTable instruction.



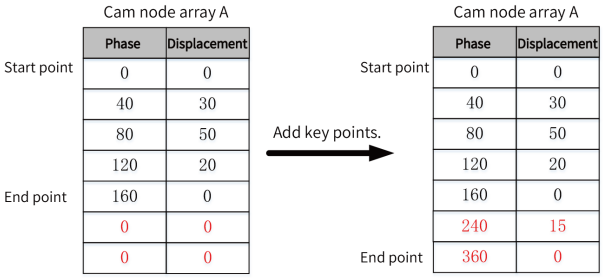
The program example is as follows.

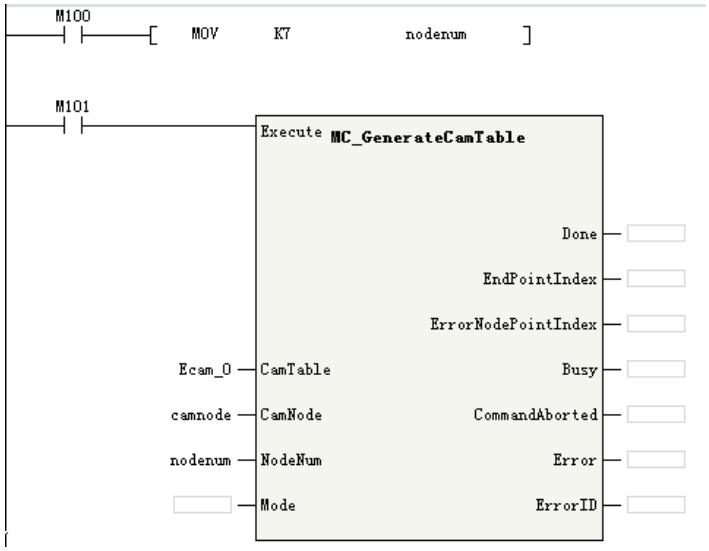


- CamNode is specified.
  Create a completely new cam node array through the PLC program, and then copy the values in the cam node array to the cam table by using the MC_GenerateCamTable instruction. The modified cam table takes effect in the next cam cycle.



The program example is as follows.





Function of NodeNum

NodeNum specifies the number of nodes in the newly generated cam table. When it is empty, the number of nodes in the cam table remains unchanged. When it is not empty, the node quantity specified by NodeNum is adopted.

You can modify the number of key points in the cam table and make the modification take effect in the next cam cycle by executing the MC_GenerateCamTable instruction.

Cam node array A

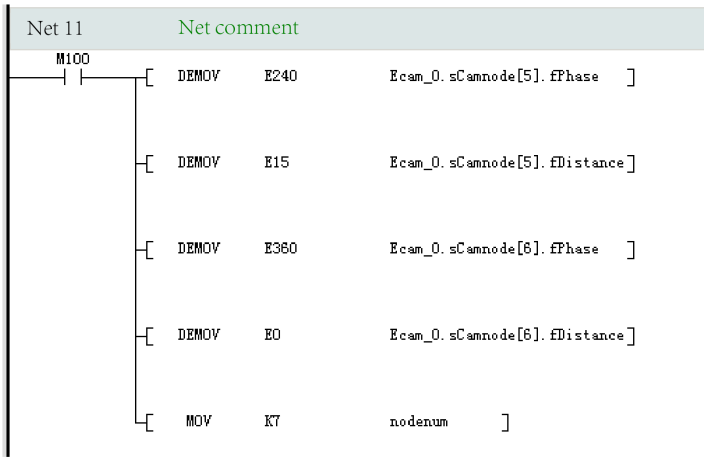
	Phase	Displacement
Start point	0	0
	40	30
	80	50
	120	20
End point	160	0
	0	0
	0	0

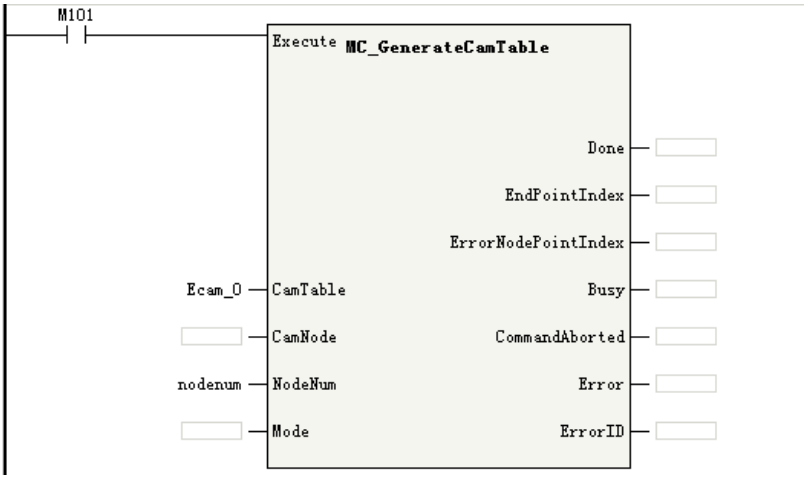
Add key points.

Cam node array A

	Phase	Displacement
Start point	0	0
	40	30
	80	50
	120	20
	160	0
	240	15
End point	360	0

The program example is as follows.





Parameter Check

This instruction first checks the cam table data when it is called.

- Both the phase and displacement of the first point must be 0. Otherwise, the instruction reports an error.
- The absolute values of the phase, displacement, and velocity ratio cannot be greater than 9999999. Otherwise, the instruction reports an error.
- The node quantity cannot be greater than 361. Otherwise, the instruction reports an error.
- The node quantity cannot be less than 2. Otherwise, the instruction reports an error.
- The phases must be sorted in ascending order. Otherwise, the instruction reports an error.
- The difference between two adjacent master axis phases must be greater than 0.0001. Otherwise, the instruction reports an error.
- The node curve type is set to linear or quintic curve. Otherwise, the instruction reports an error.

Velocity Ratio Adjustment Rules

If the velocity ratio of key points is improper, this instruction automatically adjusts the velocity ratio of cam nodes based on the following rules:

- If the current segment is a straight line, the velocity ratio is automatically adjusted according to the formula.
  For example, if the curve between points A1 and A2 is a straight line, the calculated velocity ratio is written to A2.

  The coordinates of A1 are (x1, y1) and the coordinates of A2 are (x2, y2). Then, the velocity ratio of straight line A1-A2 is as follows:

  $$V2 = \frac{y2 - y1}{x2 - x1}$$

- If a quintic curve is followed by a straight line, adjustment is required to ensure the continuity of the velocity ratio at the link point between the quintic curve and the straight line and prevent jumping. Assume that the curve between points A1 and A2 is a quintic curve, and that between points A2 and A3 is a straight line.

  The velocity ratio of the straight line segment is calculated and written to A3. Then, the velocity ratio of the end point of the quintic curve is adjusted and written to A2.

  The coordinates of A2 are (x2, y2) and the coordinates of A3 are (x3, y3). Then, the velocity ratio of straight line A2-A3 is as follows:

$$V3 = \frac{y2 - y1}{x2 - x1}$$

The velocity ratio of point A2 is set to a value same as that of point A3.

V2 = V3

- If a quintic curve is followed by a quintic curve, no adjustment is required.
- If a straight line is followed by a straight line, the link velocity of each segment needs to be calculated separately. In this case, sudden change in the link velocity ratio is allowed.
  For example, assume that A1-A2 is straight line segment 1, and A2-A3 is straight line segment 2. The velocity ratio of segment 1 is calculated first and written to A2, and then the link velocity of segment 2 is calculated and written to A3. In this case, there is a sudden change in velocity caused by unequal link velocities between segment 1 and segment 2.

Re-execution

If this instruction is re-triggered while the Busy signal is still active, the cam table is modified according to the new parameter settings.

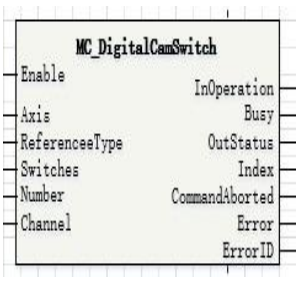
Multi-execution

If a new MC_GenerateCamTable instruction is triggered while the Busy signal of the current MC_GenerateCamTable instruction is still active, the current instruction is aborted, the CommandAborted signal output becomes active, and the cam table is modified according to the parameter settings of the newly triggered instruction.

5.13.2.11 MC_DigitalCamSwitch

MC_DigitalCamSwitch: Tappet control

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_DigitalCamSwitch	Electronic cam tappet control		MC_DigitalCamSwitch(E nable := ???, Axis := ???, ReferenceType := , Switches := ???, Number := ???, Channel := ???, InOperation => , Busy => , OutStatus => , Index => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	MC_DigitalCamSwitch: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name	No	-	-	_sMCAXIS_INFO _sENC_AXIS
S2	ReferenceType	Position type 0: Set position of the previous cycle 1: Set position of the current cycle 2: Feedback position of the current cycle 3: Phase of the master axis when the axis specified by Axis works as the cam slave axis Set to 0, 1, and 2: Position of axis when the axis works as the cam master axis Set to 3: Position of the master axis when the axis works as the cam slave axis	Yes	0	0 to 3	INT
S3	Switches	Switch	No	-	-	_sMC_DigitalSwitch [1-32]
S4	Number	Qty	No	-	1 to 32	INT
S5	Channel	Tappet terminal selection: 0 to 13 indicate actual terminals. 1000 to 1007 indicate virtual tappets.	No	-	-	INT
D1	InOperation	Executing tappet	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D4	OutStatus	Output status	Yes	OFF	ON/OFF	BOOL

D5	Index	Index Comparison point to be executed	Yes	0	0 to 31	INT
D6	Command Aborted	Abortion of execution	Yes	OFF	ON/ OFF	BOOL
D7	Error	Error flag	Yes	OFF	ON/ OFF	BOOL
D8	ErrorID	Fault code	Yes	0	-	INT16

Function Description

This instruction works with the cam to implement the tappet function. Switches specifies the tappet output points, and Channel specifies the tappet terminal. The instruction allows a DI terminal of the body but not an expansion module to be used as a tappet terminal.

- The instruction latches the input parameters on the left on the rising edge of Enable to execute the tappet output comparison function.
- When Enable is ON, modifications on the input parameters on the left do not take effect. The Enable signal must remain ON during the entire tappet output process.
- The instruction stops comparison output on the falling edge of Enable and aborts the tappet terminal that outputs ON.

Used to select the output terminal hardware source.

The tappet terminal is specified by the variable Channel. 0 to 7 correspond to Y00 to Y07, 8 to 13 corresponding to Y10 to Y15, and 1000 to 1007 indicate virtual tappets, which are only counted as tappets but not output to actual hardware terminals.

Tappet point setting

Tappet comparison points are specified by the variable Switches. This variable is a _sMC_DigitalSwitch structure array.

Variable name	Data type	Description
fPosition	REAL	Absolute position for the output to turn ON
iMode	INT	Switch mode 0: Disable 1: Position type 2: Time type
iDirection	INT	Direction of the master axis 0: Forward 1: Reverse 2: None

Variable name	Data type	Description
fParameter	REAL	Position type: ON end position Time type: ON output time (unit: ms). The decimal part after the decimal point is ignored in time mode, and the value cannot be greater than 10000 ms.

NOTE

The ON start point of the tappet point array in one direction must remain unique. For example, in the set of points of which iDirection is 0/2, fPosition must be unique.

Start and end of tappet comparison output

After the tappet instruction is executed, it internally sorts the tappet point array, determines the tappet point closest to the current position of the axis, and sets the tappet to ON immediately when the axis moves to this point.

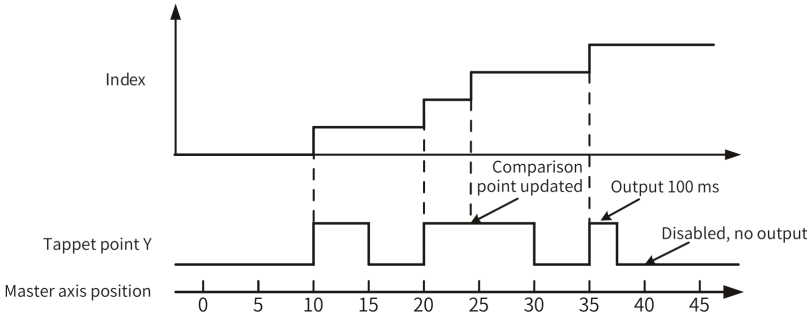
iMode specifies the comparison mode.

- When iMode is set to 0, the comparison point is disabled. When the axis passes this point, the tappet has no output.
- When iMode is set to 1, fParameter specifies the position where the tappet output turns OFF.
- When iMode is set to 2, fParameter specifies the time duration (in ms) during which the tappet output remains ON.

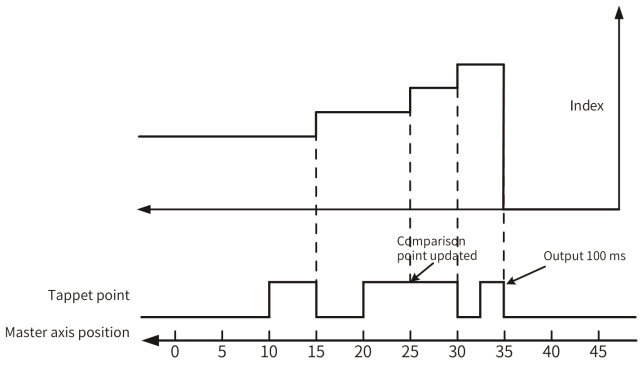
iDirection specifies the running direction of the master axis. It needs to work with iMode.

No.	fPosition	fParameter	iMode	iDirection
0	10	15	1	0
1	20	25	1	0
2	24	30	1	0
3	35	100	2	0
4	40	45	0	0
5	10	15	1	1
6	20	25	1	1
7	24	30	1	1
8	35	100	2	1

When the master axis runs in forward direction:



When the master axis runs in reverse direction:



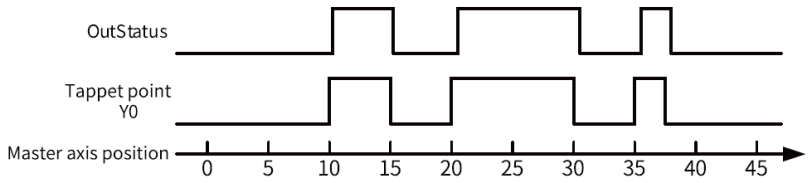
Note

NOTE

When the width of the tappet output ON is less than one EtherCAT cycle, the actual output terminal will last for one EtherCAT cycle.

Tappet Output Status Monitoring

OutStatus indicates the tappet output state. In the main task, this variable is used to monitor the tappet output state. Assume that the tappet point is set to Y0.



Re-triggering

This instruction is an instruction controlled by Enable and does not involve re-triggering.

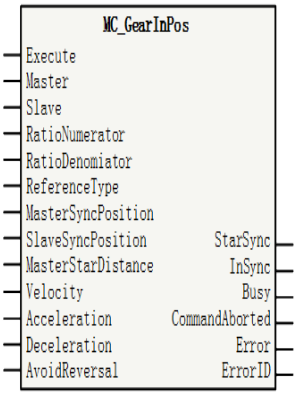
Multi-execution

If the values of Channel of two MC_DigitalCamSwitch instructions are the same, and the second instruction is triggered while the Busy signal of the first instruction is still active, the first instruction is aborted and the tappet point is output under the control of the second tappet instruction.

5.13.2.12 MC_GearInPos

MC_GearInPos -- Start the gear operation at the specified position

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_GearInPos	Start the gear operation at the specified position		MC_GearInPos(Execute := ??? , Master := ???, Slave :=??? , RationNumberator := , RationDenominator := , ReferenceType := , MasterSyncPosition :=??? , SlaveSyncPosition :=??? , MasterStarDistance :=??? , Velocity := , Acceleration := , Deceleration := , AvoidReversal := , StartSync => , InSync =>, Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_GearInPos: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Master	Master axis	No	-	-	_sMCAXIS_INFO

		Bus servo axis, local pulse axis, bus encoder axis, or local encoder axis				_sENC_AXIS_sENC_EXT_AXIS _sMasterAxis
S2	Slave	Slave axis Bus servo axis or local pulse axis	No	-	-	_sMCAXIS_INFO
S3	RatioNumerator	Gear ratio (numerator)	Yes	1	Positive or negative number	DINT
S4	RatioDenominator	Gear ratio (denominator)	Yes	1	Positive number	DINT
S5	ReferenceType	Position type 0: Set position of the previous cycle 1: Set position of the current cycle 2: Feedback position of the current cycle	Yes	0	0 to 2	INT
S6	MasterSyncPosition	Sync position of the master axis	No	-	Positive number, negative number, or 0	REAL
S7	SlaveSyncPosition	Sync position of the slave axis	No	-	Positive number, negative number, or 0	REAL
S8	MasterStarDistance	Movement distance of the master axis in the Catching phase	No	-	Positive number or 0	REAL
S9	Velocity	Velocity (Reserved)	Yes	1	Positive number	REAL
S10	Acceleration	Acceleration rate (Reserved)	Yes	0	Positive number or 0	REAL
S11	Deceleration	Deceleration rate (Reserved)	Yes	Acceleration	Positive number or 0	REAL
S12	AvoidReversal	Reversal prohibited (Reserved)	Yes	0	0	INT
D1	InSync	Synchronizing	Yes	OFF	ON or OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON or OFF	BOOL

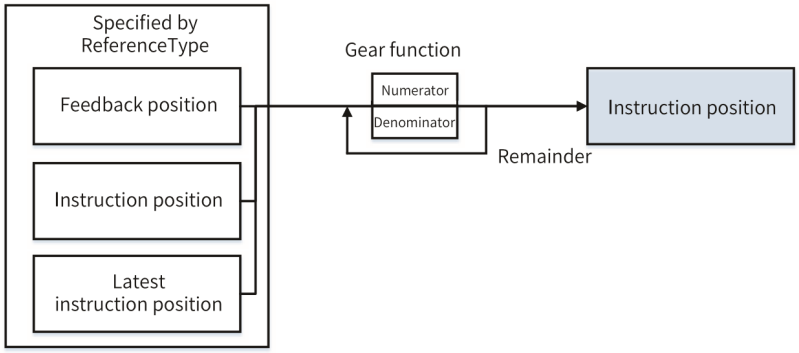
D4	CommandAborted	Abortion of execution	Yes	OFF	ON or OFF	BOOL
D5	Error	Error	Yes	OFF	ON or OFF	BOOL
D6	ErrorID	Error code	Yes	0	-	INT16

Function Description

The MC_GearInPos instruction specifies the axis to be operated through Slave and determines a motion plan based on the input parameters RatioNumerator, RatioDenominator, ReferenceType, MasterSyncPosition, SlaveSyncPosition, and MasterStarDistance to implement gear operation.

The instruction allows you to select the master axis position through ReferenceType:

- _mcCommand: Instruction position (value calculated in the latest task cycle). For the current cycle, use the master axis instruction position calculated during the previous task cycle. In the fixed-cycle task before the master axis instruction position is calculated, use the calculated master axis instruction position.
- _mcLatestCommand: Instruction position (value calculated in the same task cycle). Use the master axis instruction position calculated in the same task cycle.
- _mcFeedback: Value obtained in the same task cycle. Use the master axis feedback position obtained in the same task cycle.



The interval from the point that the slave axis starts catching up to the point that the axis reaches the sync position is called Catching phase. After the axis reaches the sync position, the phase is called InGear phase. After gear synchronization, the slave axis moves synchronously with the master axis at any interval.

The nature of the motion process in the Catching phase is that the slave axis follows an electronic cam of the master axis. At this time, based on the master axis range (MasterCatchPosition, MasterSyncPosition) and the slave axis range (slave axis position when the Catching phase is triggered, SlaveSyncPosition), the instruction plans a quintic cam curve based on the set gear ratio, velocity ratio of the master and slave axes when the Catching phase is triggered, and the above position parameters, so that the slave axis follows the master axis to complete the cam movement in the Catching phase.

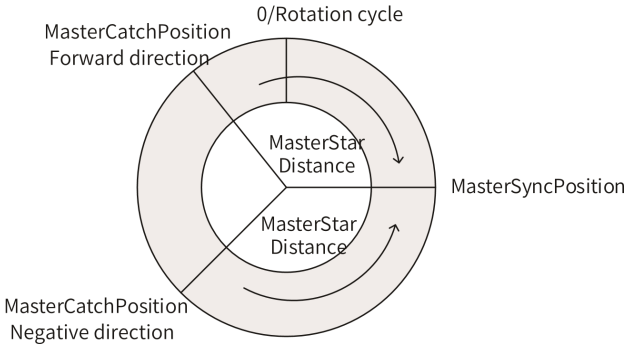
The instruction determines the catch-up start position of the master axis (MasterCatchPosition) based on the movement direction of the master axis at the instruction start time. When the master axis is a linear axis:

If the master axis movement direction is forward: MasterCatchPosition = MasterSyncPosition - MasterStartDistance

If the master axis movement direction is reverse: MasterCatchPosition = MasterSyncPosition + MasterStartDistance

If the instruction determines that the current position and movement direction of the master axis do not allow the master axis to reach the catch-up start position to trigger the Catching phase, it reports error 9304.

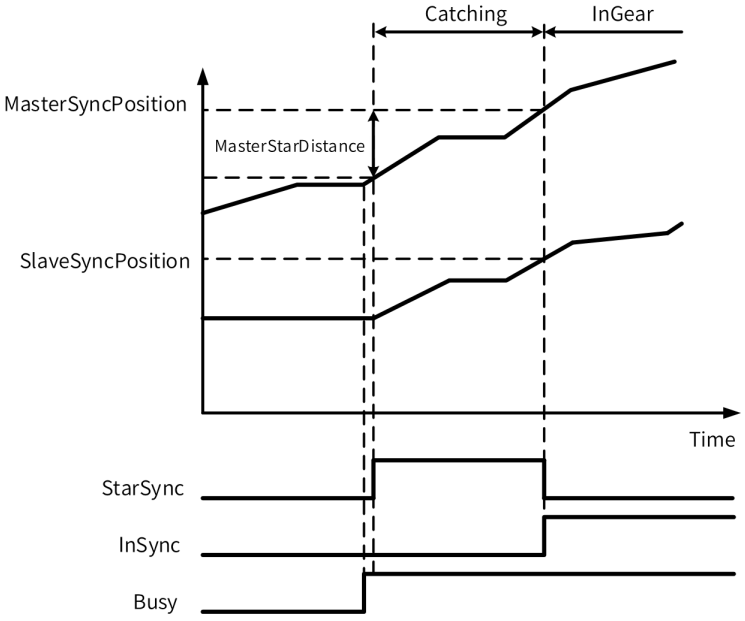
When the master axis is in rotation mode, the calculation principle of its catch-up start position is shown in the following figure.



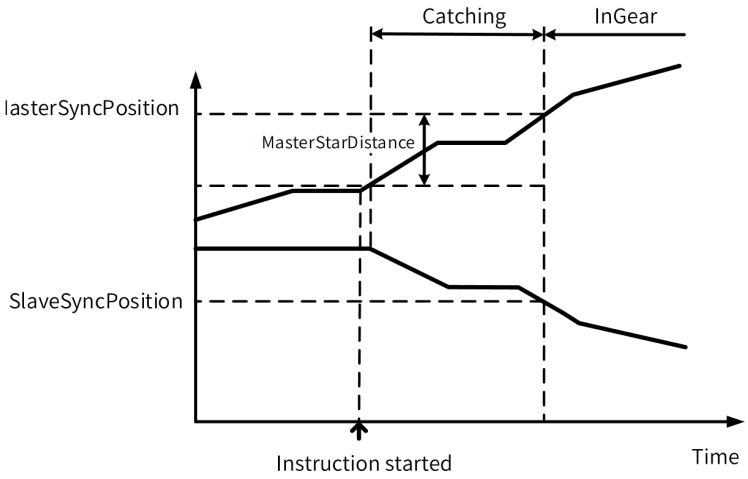
- When the master axis velocity is 0 at the instruction start time, the instruction waits until a movement direction is available for the master axis to determine the catch-up start position.
- If ReferenceType is set to 2 (Feedback position of the current cycle) or the axis is an encoder axis, the instruction can determine the movement direction and start calculating the catch-up start position only after the master axis movement direction remains unchanged for five consecutive cycles.

After gear synchronization, the slave axis multiplies the master axis velocity by the gear ratio (RatioNumerator/RatioDenominator) to obtain the target velocity, and synchronizes the acceleration and deceleration actions with the master axis.

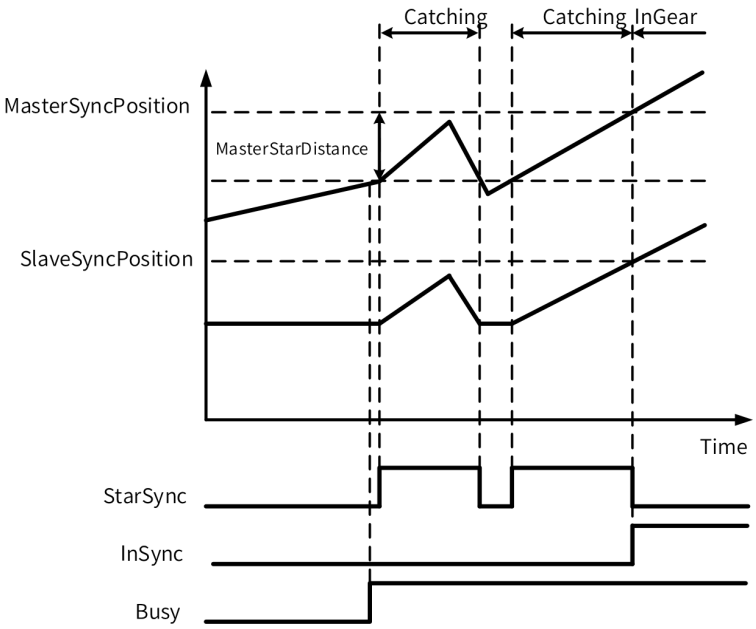
When the gear ratio is positive, the slave axis moves in the same direction as the master axis after it reaches the sync position.



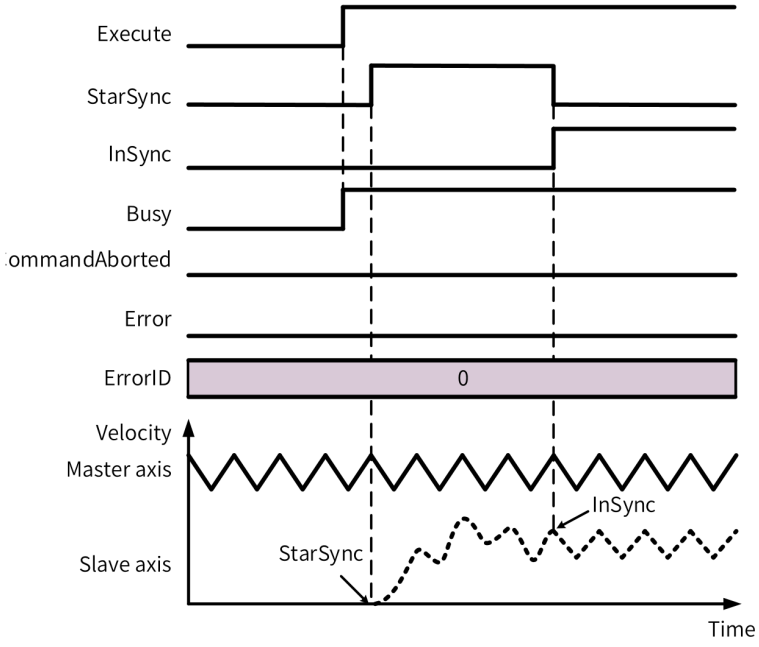
When the gear ratio is negative, the slave axis moves in the opposite direction of the master axis after it reaches the sync position.



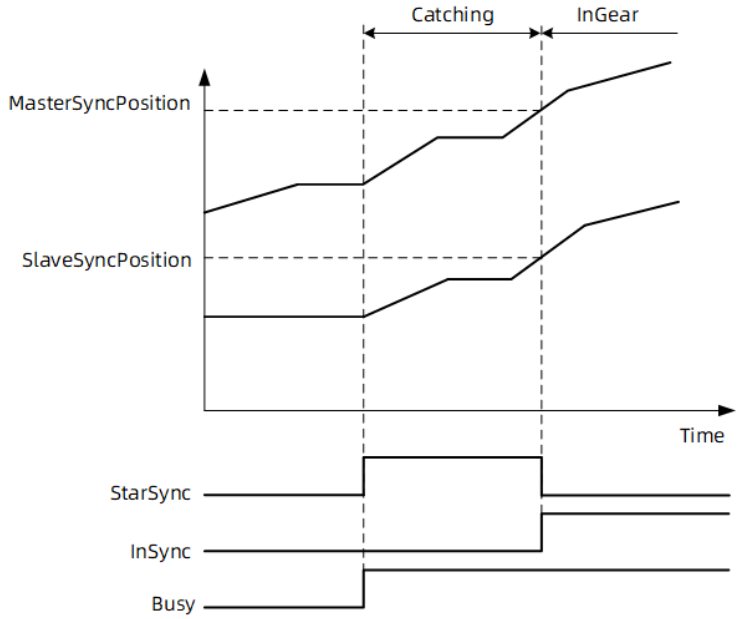
In the catching process, if the master axis position deviates from the catching area due to vibration, the instruction exits the Catching phase and sets StarSync to FALSE.



In addition, when the master axis velocity varies greatly in different cycles, the slave axis velocity is also not fixed. Before the InSync phase is reached, the slave axis tracks the master axis position. The following figure shows the details.



If MasterStarDistance is set to 0, the instruction immediately enters the Catching phase while starting Execute, StartSync changes to TRUE, and the slave axis starts the gear catching action.



Re-triggering

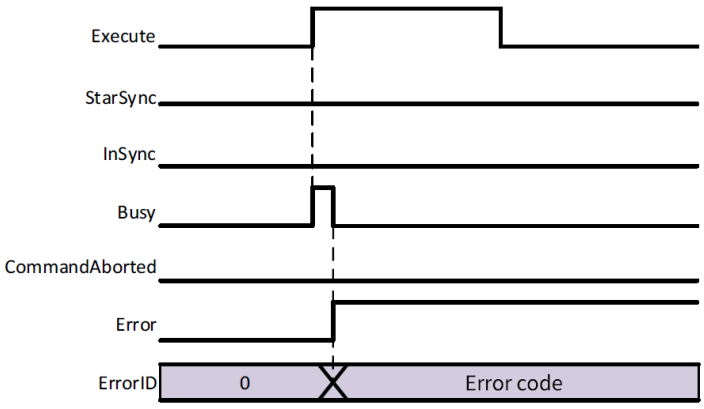
If the MC_GearInPos instruction is triggered again when the Busy signal of the previous MC_GearInPos instruction is active and StarSync is OFF, the slave axis motion from the Catching phase to the InSync phase is re-planned based on the input parameters, and the StarSync and InGear flag bits are reset. If the axis already enters the Catching phase and StarSync is ON, you need to call MC_GearOut to restart the MC_GearInPos instruction. Otherwise, the instruction reports error 9303.

Multi-execution

If the MC_GearInPos instruction is triggered again when the Busy signal of the previous MC_GearInPos instruction is active and StarSync is OFF, the Busy signal of the second MC_GearInPos instruction is active, the first MC_GearInPos instruction is aborted, and the slave axis motion from the Catching phase to the InSync phase is re-planned based on the input parameters. If the first MC_GearInPos instruction already enters the Catching phase and StarSync is ON, you need to call MC_GearOut and then trigger the second MC_GearInPos instruction. Otherwise, the instruction reports error 9303.

Abnormalities

When the instruction encounters an abnormality, the timing diagram is as follows.



It is prohibited to execute the MC_SetPositionon instruction on the master and slave axes during execution of the MC_GearInPos instruction. When the MC_SetPosition instruction is being executed on the master axis, slave axis rapid tracking may occur, which is dangerous. Unbind the master axis from the slave axis before executing the MC_SetPosition instruction on the master or slave axis.

5.13.2.13 Fault Codes

When a fault occurs during use of the electronic cam functions, refer to the fault codes listed in the following table for troubleshooting.

Fault code	Fault information	Solution	Stop triggered
9200	Failed to obtain the cam table configuration file.	1. Check whether the board software and software tool software match. 2. Re-download the cam configuration table.	No
9201	Failed to obtain the master axis.	1. Check whether the master axis called in the program exists. 2. Check whether the master axis has reported an error. 3. Check whether the master axis is a profile axis.	Yes
9202	Failed to obtain the slave axis.	1. Check whether the slave axis called in the program exists. 2. Check whether the slave axis has reported an error.	No
9203	Failed to obtain the cam table.	Check whether the cam table called exists.	Yes
9204	The number of cams executed simultaneously in the PLC program exceeds the maximum allowable value.	Check whether the number of cams executed simultaneously in the program exceeds the threshold.	Yes
9205	The corresponding cam node is not found.	1. This instruction can be called only when the slave axis is in cam engagement state. 2. The specified slave axis is not a cam node. Set the parameters again.	Yes
9206	The master axis is changed during cam engagement.	Do not change the master axis during cam engagement.	Yes
9207	StartMode of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes

Fault code	Fault information	Solution	Stop triggered
9208	StartPosition of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9209	MasterStartDistance of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9210	MasterScaling of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9211	SlaveScaling of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9212	MasterOffset of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9213	SlaveOffset of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9214	MasterScaling of the MC_CamIn instruction is not a positive number.	Set this parameter to a positive number.	Yes
9215	SlaveScaling of the MC_CamIn instruction is not a positive number.	Set this parameter to a positive number.	Yes
9216	ReferenceType of the MC_CamIn/ MC_GearIn instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9217	Direction of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9218	BufferMode of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9219	The master axis phases in the node array of the cam table are not sorted in ascending order.	Sort the master axis phases in ascending order when customizing cam table nodes.	Yes
9220	The curve type setting of the node array of the cam table is out of range.	Check whether the curve type of the cam node array is set incorrectly.	Yes

Fault code	Fault information	Solution	Stop triggered
9221	The target deceleration of the MC_CamOut instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9222	MC_CamOut target deceleration out of range	Ensure that the target deceleration is within the specified range.	Yes
9223	The target acceleration of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9224	The target acceleration of the MC_Phasing instruction is out of range.	Ensure that the target acceleration is within the specified range.	Yes
9225	The target velocity of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9226	The target velocity of the MC_Phasing instruction is out of range.	Ensure that the target velocity is within the specified range.	Yes
9227	The curve type setting of the MC_GearOut instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.	Yes
9228	OutMode of the MC_CamOut instruction is out of range.	Ensure that OutMode is within the specified range.	Yes
9229	The MC_GenerateCamTable instruction detects that the cam node array is empty.	Contact Inovance for technical support.	No
9230	The node quantity specified by the MC_GenerateCamTable instruction exceeds the maximum allowable value.	Check whether the target node quantity specified in the instruction is beyond the specified range.	No
9231	The value of Mode of the MC_GenerateCamTable instruction is out of range.	Ensure that the parameter value is within the specified range.	No
9232	The node quantity specified by the MC_GenerateCamTable instruction is too small.	Ensure that the node quantity is 2 or more.	No

Fault code	Fault information	Solution	Stop triggered
9233	The RatioNumerator parameter in the gear instruction is set to 0.	Set this parameter to a non-zero integer.	Yes
9234	The RatioDenominator parameter in the gear instruction is not greater than 0.	Set this parameter to an integer greater than 0.	Yes
9235	The MC_GenerateCamTable instruction is being executed when the MC_SaveCamTable instruction is called.	Do not call the MC_SaveCamTable instruction before the cam table data update operation is completed.	No
9236	The MC_SaveCamTable instruction is being executed on the cam table when the MC_GenerateCamTable instruction is called.	Do not call the MC_GenerateCamTable instruction before the cam table is saved.	No
9237	Failed to open the cam table file during execution of the MC_SaveCamTable instruction.	1. Check whether the PLC memory runs out. 2. Replace the PLC.	No
9238	Failed to write the cam point quantity when the cam table is being saved.	1. Check whether the PLC memory runs out. 2. Replace the PLC.	No
9239	Failed to write data when the cam table is being saved.	1. Check whether the PLC memory runs out. 2. Replace the PLC.	No
9240	The phase of the first point is not 0.	Ensure that the phase of the first point is 0.	Yes
9241	The displacement of the first point is not 0.	Ensure that the displacement of the first point is 0.	Yes
9242	The value of Mode of the MC_GearOut instruction is out of range.	Ensure that the value of Mode is within the specified range.	Yes
9243	The target deceleration of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9244	The target deceleration of the MC_GearIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9245		Ensure that the parameter value is within the specified range.	Yes

Fault code	Fault information	Solution	Stop triggered
	The value of Periodic of the MC_CamIn instruction is out of range.		
9246	The phase in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.	Yes
9247	The absolute value of the displacement in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.	Yes
9248	The absolute value of the link velocity in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.	Yes
9249	The gear node is empty.	Contact the manufacturer.	Yes
9250	The master axis and slave axis are the same.	Do not use the same axis as both the master axis and slave axis of the cam gear.	Yes
9251	The configuration address of the master axis is greater than or equal to that of the slave axis.	When ReferenceType is set to set position of the current cycle, ensure that the configuration address of the master axis is less than that of the slave axis.	Yes
9252	The master axis filter coefficient fFilter[0] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).	Yes
9253	The master axis filter coefficient 2fFilter[1] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).	Yes
9254	The master axis filter coefficient 3fFilter[2] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).	Yes
9255	The sum of the master axis filter coefficients corresponding to the slave axis is not 1.	Ensure that the sum of the master axis filter coefficients corresponding to the slave axis is 1.	Yes
9256	The start position and start distance of the master axis in the MC_CamIn instruction are improper.	If the master axis works in linear mode and Direction in the instruction is set to positive, ensure that the cam synchronization point is not less than the cam engagement point.	Yes
9257		If the master axis works in linear mode and Direction in the instruction is set to negative, ensure	Yes

Fault code	Fault information	Solution	Stop triggered
	The start position and start distance of the master axis in the MC_CamIn instruction are improper.	that the cam synchronization point is not greater than the cam engagement point.	
9258	The target deceleration of the MC_GearOut instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9259	MC_Phasing target deceleration out of range	Ensure that the target deceleration is within the specified range.	Yes
9260	MC_GearIn target deceleration out of range	Ensure that the target deceleration is within the specified range.	Yes
9261	MC_GearOut target deceleration out of range	Ensure that the target deceleration is within the specified range.	Yes
9262	The target acceleration of the MC_GearIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9263	The target acceleration of the MC_GearIn instruction is out of range.	Ensure that the target acceleration is within the specified range.	Yes
9264	MC_Phasing curve type setting out of range	Ensure that the curve type setpoint in the instruction is within the specified range.	Yes
9265	MC_GearIn curve type setting out of range	Ensure that the curve type setpoint in the instruction is within the specified range.	Yes
9266	MC_GearOut curve type setting out of range	Ensure that the curve type setpoint in the instruction is within the specified range.	Yes
9267	The slave axis is modified during the cam operation.	Do not modify the slave axis during the cam operation.	Yes
9268	Mode of the MC_Phasing instruction is out of range.	Ensure that the value of the parameter is within the specified range.	Yes
9269	The current axis is not in cam control mode when the MC_CamOut instruction is called.	Ensure that the axis works in cam control mode when the MC_CamOut instruction is called.	No
9270	The current axis is not in gear control mode when the MC_GearOut instruction is called.	Ensure that the axis works in gear control mode when the MC_GearOut instruction is called.	No

Fault code	Fault information	Solution	Stop triggered
9271	The position change of the master axis is too large within a single EtherCAT cycle during cam/gear operation.	Ensure that the position change of the master axis is not greater than half a cam cycle within a single EtherCAT cycle.	Yes
9272	The point specified by Phase in the MC_GetCamTableDistance instruction does not fall between the start and end points.	Ensure that the point specified by Phase is within the specified curve.	No
9273	The slave axis is changed during execution of the MC_GearIn instruction.	Do not change the slave axis during execution of the MC_GearIn instruction.	Yes
9274	The value of Channel of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.	No
9275	The axis is not found.	Ensure that the axis specified by Axis exists.	No
9276	The number of tappets allowed to be executed at the same time is out of range.	Ensure that the number of tappets allowed to be executed at the same time must be no more than 8.	No
9277	The value of ReferenceType of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.	No
9278	The value of Number of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.	No
9279	The Switches array of the MC_DigitalCamSwitch instruction is empty.	Check whether the length of the Switches array meets requirements.	No
9280	The value of fPosition of the tappet array is out of range.	Ensure that the parameter value is within the specified range.	No
9281	The value of iMode of the tappet array is out of range.	Ensure that the parameter value is within the specified range.	No
9282	The value of iDirection of the tappet array is out of range.	Ensure that the parameter value is within the specified range.	No
9283	The value of fParameter of the tappet array is out of range.	Ensure that the parameter value is within the specified range.	No

Fault code	Fault information	Solution	Stop triggered
9284	When the tappet comparison point is set to time mode, the time setting is out of range.	Ensure that the parameter value is within the specified range.	No
9285	The selected axis is not under cam control when ReferenceType of the MC_DigitalCamSwitch instruction is set to 3.	Call the MC_DigitalCamSwitch instruction after cam control takes effect.	No
9286	Axis communication is interrupted during tappet execution.	Ensure that axis communication is not interrupted during tappet execution.	No
9287	The comparison position start points are the same during tappet execution.	Ensure that the start points are not duplicate.	No
9288	The comparison position start and end point are the same during tappet execution.	Ensure that the start and end points are not duplicate.	No
9289	The selected tappet terminal is being used by another function.	Check whether the terminal is set as the pulse output axis.	No
9290	The MC_DigitalCamSwitch instruction cannot be executed because the state of the motion control axis is improper.	Do not execute the MC_DigitalCamSwitch instruction in homing mode.	No
9291	The MasterSyncPosition setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9292	The SlaveSyncPosition setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9293	The MasterStarDistance setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9294	The Velocity setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.	Yes
9295	The Velocity setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.	Yes

Fault code	Fault information	Solution	Stop triggered
9296	The Acceleration setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.	Yes
9297	The Acceleration setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.	Yes
9298	The Deceleration setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.	Yes
9299	The Deceleration setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.	Yes
9300	The AvoidReversal setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9301	The master axis speed is zero when the MC_GearInPos instruction is started.	Ensure that the master axis speed is not zero when starting this instruction.	Yes
9302	The master axis did not move during the catching phase of the MC_GearInPos instruction.	When MasterStarDistance is set to 0, ensure that the input MasterSyncPosition does not overlap with the current position of the master axis.	Yes
9303	When the MC_GearInPos instruction is started, the speed of the slave axis is not zero before entering the catching phase.	Ensure that the slave axis remains stationary before entering the catching phase.	Yes
9304	Failed to enter the catching phase when the MC_GearInPos instruction is executed.	Ensure that the master axis can enter the catching phase under the current position and motion direction conditions.	Yes
9305	The velocity of the slave axis exceeds the limit during execution of the MC_GearInPos instruction.	Ensure that the parameter value is within the specified range.	Yes

5.13.3 Axis Group Control Instructions

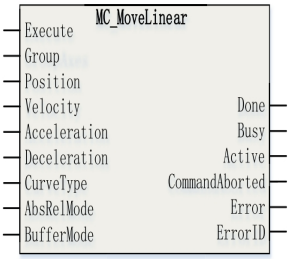
5.13.3.1 List of Instructions

The following table lists the axis group control instructions.

Instruction category	Name	Function
Axis group control instruction	MC_MoveLinear	Linear interpolation
	MC_MoveCircular	Circular interpolation
	MC_MoveEllipse	Elliptical interpolation
	MC_GroupStop	Stop axis group operation
	MC_GroupPause	Pause axis group operation

5.13.3.2 MC_MoveLinear

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveLinear	Linear interpolation		MC_MoveLinear(Execute := ???, Group := ???, Position := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , AbsRelMode := , BufferMode := , Done => , Busy => , Active => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_MoveLinear: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Group	Axis group ID	No	-	-	Axis group, INT

S2	Position	Target position	No	-	Positive number Negative number 0	REAL[0-3] or sMCGROUP_INFO
S3	Velocity	Target velocity	No	-	Positive number	REAL
S4	Acceleration	Acceleration rate	No	-	Positive number	REAL
S5	Deceleration	Deceleration rate	Yes	Acceleration rate	Positive number	REAL
S6	CurveType	Velocity curve type 0: T-shaped velocity curve 1: 5-segment S-shaped curve Others: T-shaped velocity curve	Yes	0	0 to 1	INT
S7	AbsRelMode	Absolute or relative positioning mode 0: Absolute positioning 1: Relative positioning	Yes	0	0 to 1	INT
S8	BufferMode	Buffer mode 0: Aborting and no transition 1: Buffered and no transition 2: Previous velocity and no transition 3: Superimpose corners	Yes	0	0 to 3	INT
D1	Done	Target position reached ON after the target position is reached	Yes	OFF	ON OFF	BOOL

D2	Busy	Busy flag	Yes	OFF	ON OFF	BOOL
D3	Active	Controlling ON when starting to execute the current curve segment	Yes	OFF	ON OFF	BOOL
D4	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D5	Error	Error flag	Yes	OFF	ON OFF	BOOL
D6	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Error Codes.

Function and Instruction Description

The MC_MoveLinear instruction performs linear interpolation on axis groups. It is active on the rising edge.

- Specifying the axis
  Group is latched on the rising edge of the Execute input.

  Modification on Group is invalid when Execute is ON.

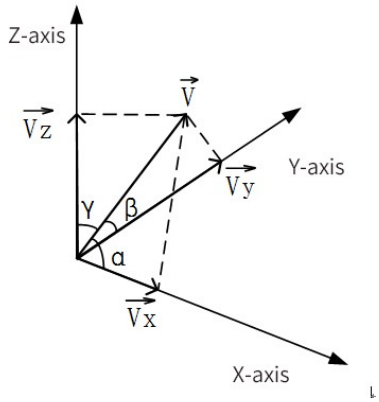
  Modification on Group is invalid when Execute is OFF.
- Relationship with single-axis control instructions
  This instruction can be triggered only after all axes in the axis group are switched to the StandStill state by executing the MC_Power instruction.

  This instruction is invalid if it is triggered during single-axis operations, such as jogging, torque control, homing, and stop.

  After this instruction is triggered, the single-axis PLCOpen state machine is in SynchronizedMotion state. During operation, this instruction cannot be aborted by single-axis motion instructions. After the interpolation curve is completed, the single-axis PLCOpen state machine enters the StandStill state, and single-axis motion instructions can be executed at this time.

  Velocity specifies the target velocity of the interpolator. The velocity of each coordinate axis is resolved according to formulas (1), (2), and (3).
- Parameter description
  Position specifies the target position or displacement. Position[0] indicates the position displacement component of the x-axis, Position[1] indicates the position displacement component of the y-axis, Position[2] indicates the position displacement component of the z-axis, and Position[3] indicates the position displacement component of the auxiliary axis.

Velocity specifies the target velocity of the interpolator. The velocity of each coordinate axis is resolved according to formulas (1), (2), and (3).



$$Vx = V \times \cos\alpha \quad (1)$$

$$Vy = V \times \cos\beta \quad (2)$$

$$Vz = V \times \cos\gamma \quad (3)$$

$$V = \sqrt[2]{Vx^2 + Vy^2 + Vz^2} \quad (4)$$

The interpolation velocity of the auxiliary axis differs in the following two cases:

1. When the point on the coordinate axes does not move and the auxiliary axis moves independently, the auxiliary axis moves according to the target velocity specified by Velocity.
2. When the point on the coordinate axes moves, the auxiliary axis will reach the target position at the same time as the point on the coordinate axes. Assume that the linear interpolation length is L1, the target displacement of the auxiliary axis is L2, and the linear interpolation rate at a certain moment is V0. The velocity Va of the auxiliary axis is calculated as follows:

$$Va = V0 \times \frac{L2}{L1} \quad (5)$$

- Selecting the curve type

  If CurveType is set to 0, acceleration/deceleration is planned for axis group interpolation based on the T-shaped velocity curve.

  If CurveType is set to 1, acceleration/deceleration is planned for axis group interpolation based on the S-shaped velocity curve.

  If CurveType is set to Others, acceleration/deceleration is planned for axis group interpolation based on the T-shaped velocity curve.

- Selecting the relative or absolute mode
  If AbsRelMode is set to 0, the absolute positioning mode is used. After this instruction is triggered, the three coordinate axes move to the position specified by (Position[0], Position[1], Position[2]), and the auxiliary axis moves to the position specified by Position[3].

  If AbsRelMode is set to 1, the relative positioning mode is used. Set the position of the three coordinate axes of the axis group to (Px, Py, Pz) and the current position of the auxiliary axis to Pa. After this instruction is triggered, the three coordinate axes move to the position specified by (Px + Position[0], Py + Position[1], Pz + Position[2]), and the auxiliary axis moves to the position specified by (Pa + Position[3]).

- Buffer and transition
  There are four optional buffer and transition modes. For details, see the "Interpolation Function" section in the "AutoShop Software Programming and Application Manual".

No.	Buffer Mode	Description
0	Aborting and no transition	Immediately switch to the next function block. There is no transition curve.
1	Buffered and no transition	Execute the buffered function block after the first segment of deceleration is completed. There is no transition curve.
2	Previous velocity and no transition	Move to the end of the first segment at the current velocity and start the second segment at the rate of the first segment.
3	Superimpose corner transition	Add acceleration of the second segment when deceleration starts in the first segment. There is a transition curve.

When buffer mode 1, 2, or 3 is selected, the interpolation instruction allows up to 8 curves to be buffered. When this instruction enters the buffer state, the Busy signal is active.
When it is executed, the Active output becomes active. When the execution is completed, the Done signal output becomes active.

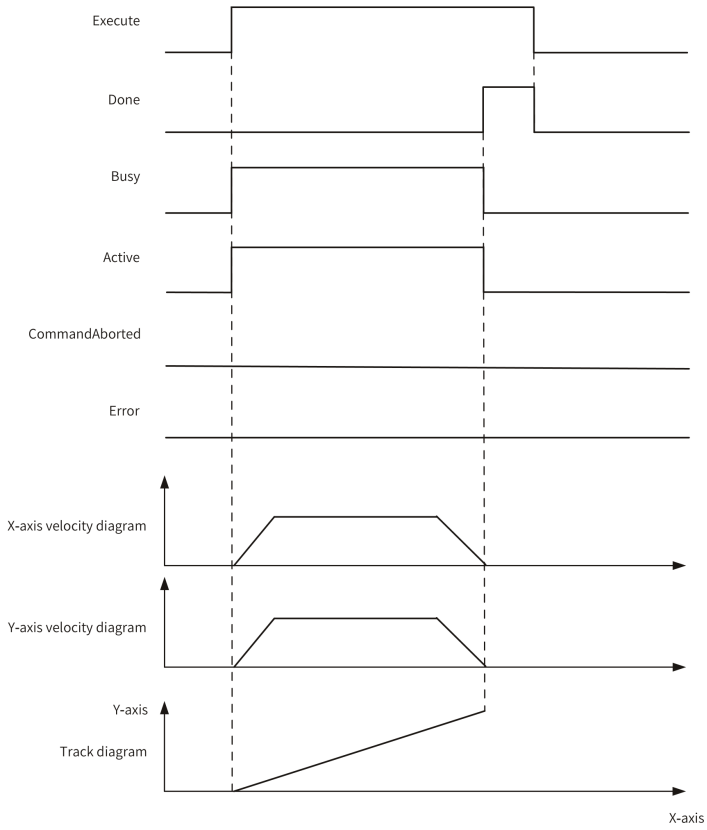
When mode 0 (Aborting and no transition) is selected for a new interpolation instruction, it will abort all interpolation instructions that are being executed or buffered. The CommandAborted output of the aborted interpolation instructions becomes active.

- Re-execution
  This instruction cannot be re-executed.

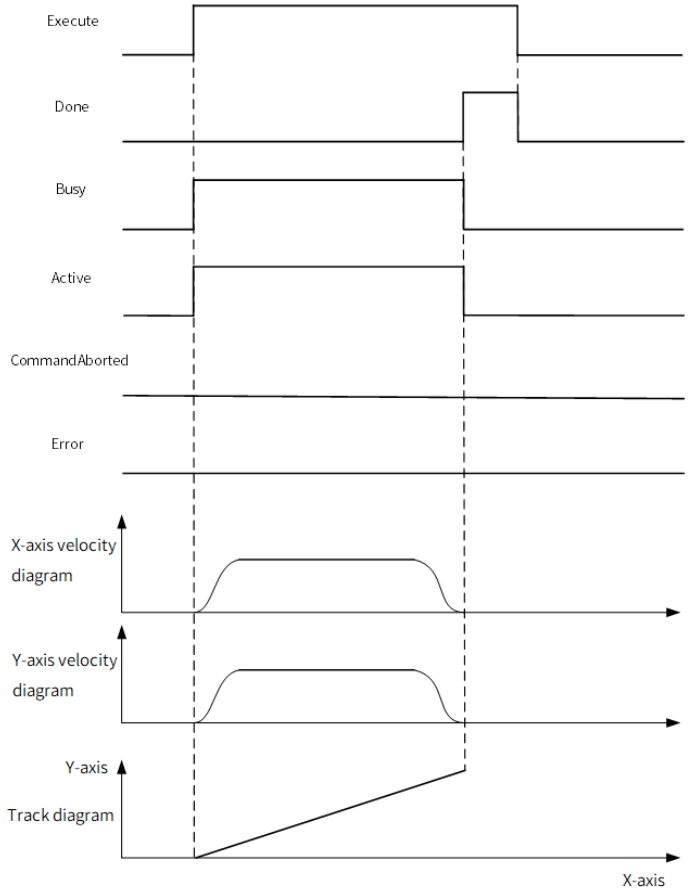
  If this instruction is triggered repeatedly when the Busy output is ON, the axis reports error 9421 (instruction re-execution error), and all axes stop operation immediately and enter the ErrorStop state.

Timing Diagram

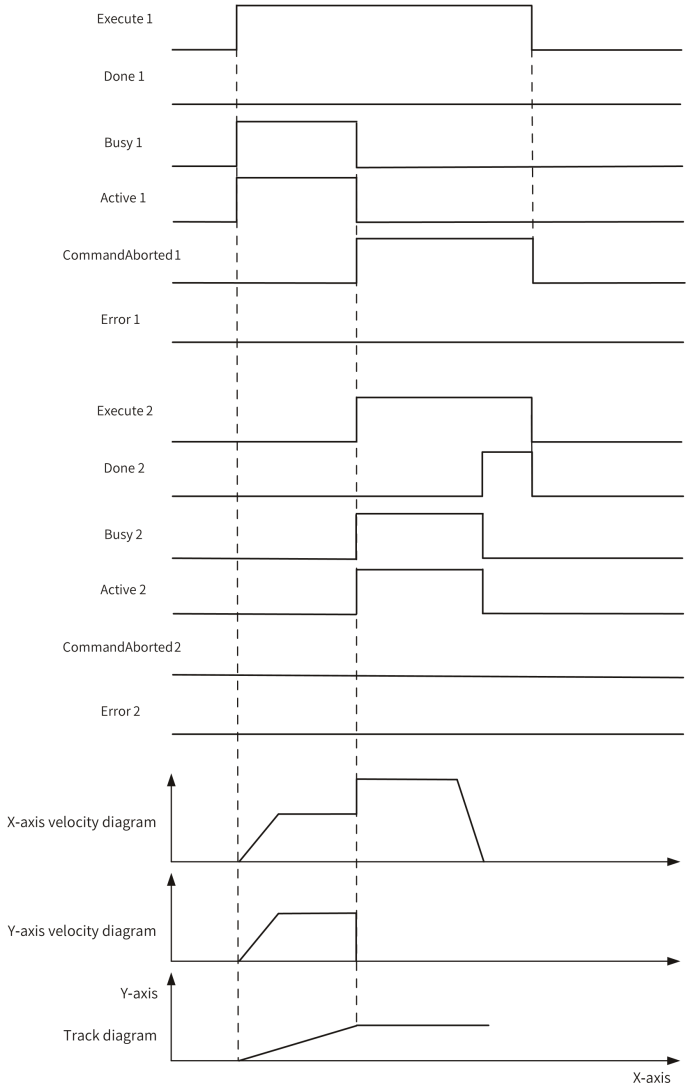
- A linear interpolation instruction is called to perform interpolation along the x-axis and y-axis. CurveType is set to 0, so motion is planned based on the T-shaped velocity curve.



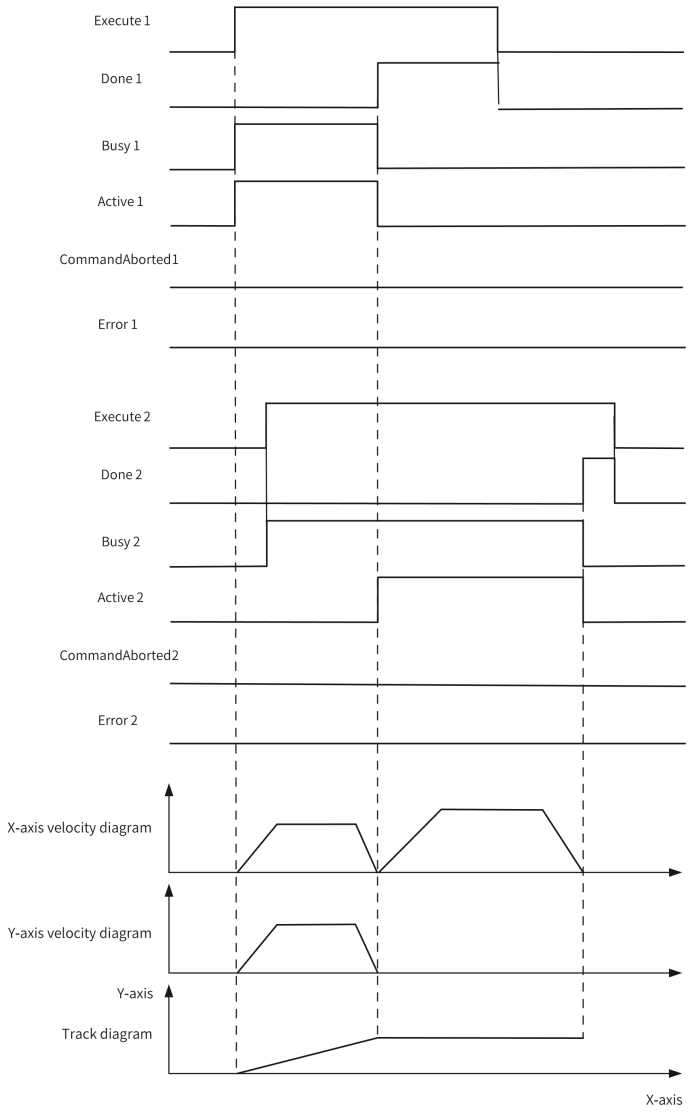
- A linear interpolation instruction is called to perform interpolation along the x-axis and y-axis. CurveType is set to 1, so motion is planned based on the 5-segment S-shaped velocity curve.



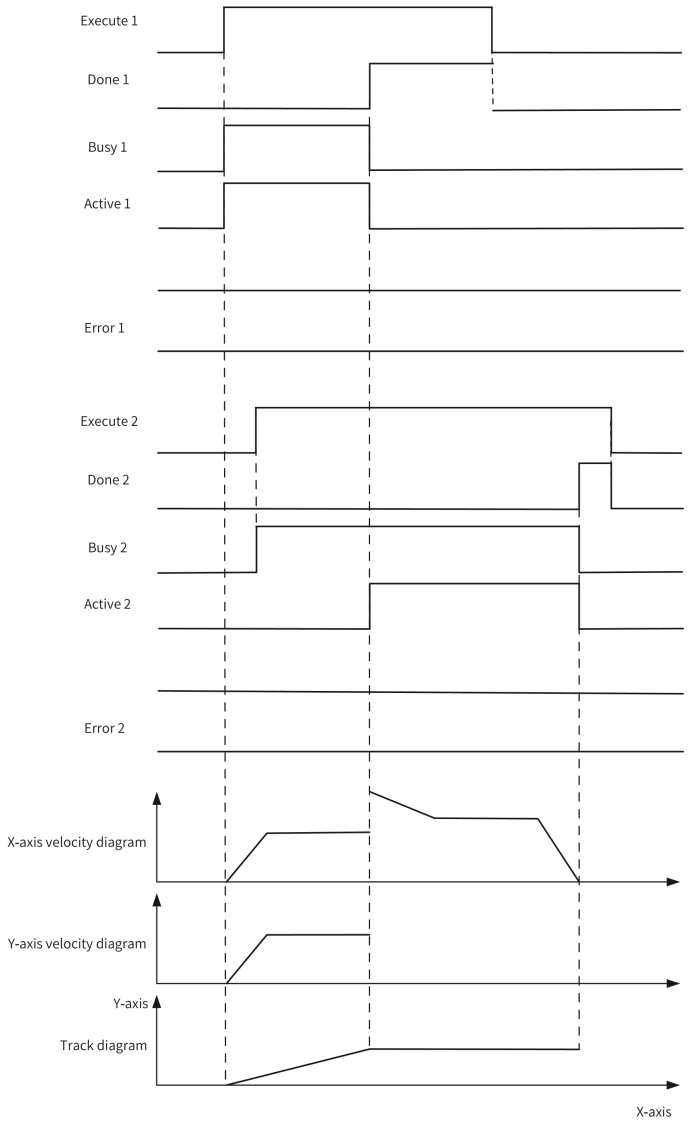
- Two linear interpolation instructions are called, of which the second one is triggered during execution of the first one to abort the first one.



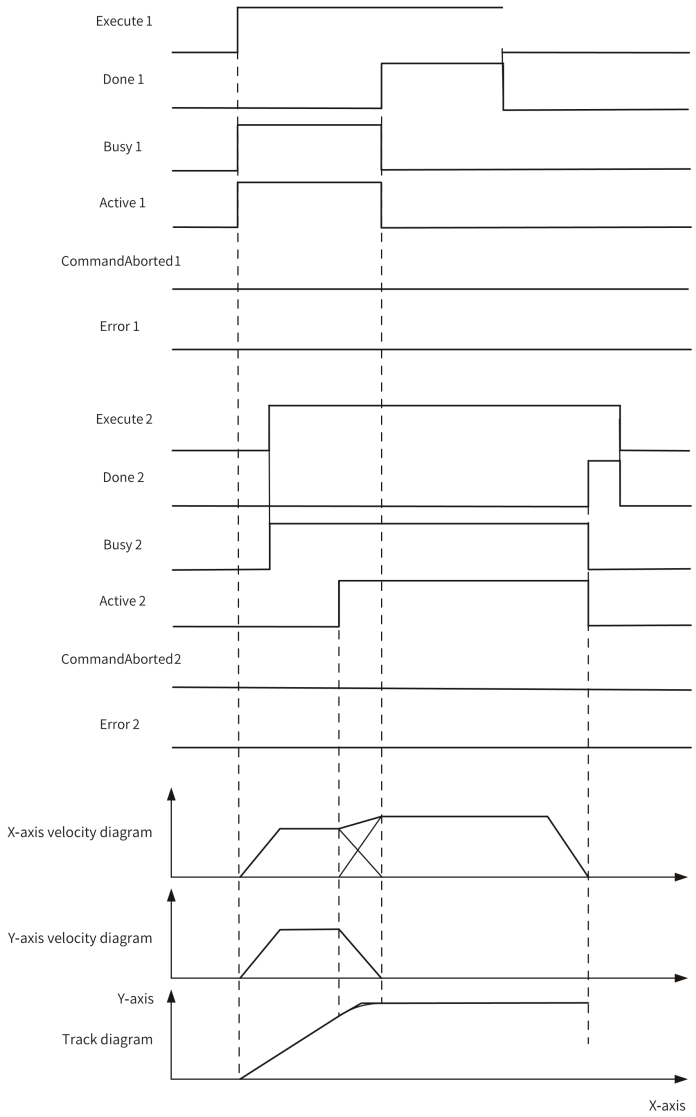
- Two linear interpolation instructions are called, the second instruction is executed in buffered and no transition mode.



- Two linear interpolation instructions are called, and the second instruction is executed in previous velocity and no transition mode.

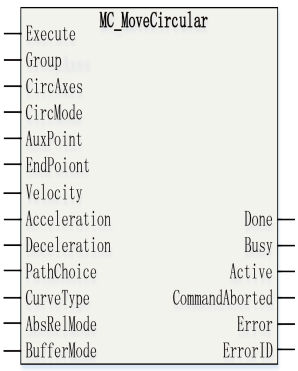


- Two linear interpolation instructions are called, and the second instruction is executed in superimpose corner mode.



5.13.3.3 MC_MoveCircular

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveCircular	Circular interpolation		MC_MoveCircular(Execute := ???, Group := ???, CircAxes := , CircMode := , AuxPoint := ???, EndPoint := ???, Velocity := ???, Acceleration := ???, Deceleration := , PathChoice := , CurveType := , AbsRelMode := , BufferMode := , Done => ,

Instruction	Name	LD Expression	LiteST Expression
			Busy => , Active => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-
32-bit Instruction	MC_MoveCircular: Continuous execution

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Group	Axis ID	No	-	-	Axis group, INT
S2	CircAxes	Circular axis 0: x-y axis plane 1: y-z axis plane 2: x-z axis plane	Yes	0	0 to 2	INT
S3	CircMode	Circular interpolation mode 0: Border point 1: Center 2: Radius	Yes	0	0 to 2	INT
S4	AuxPoint	Auxiliary point	No	-	Positive number Negative number 0	REAL[0-3]
S5	EndPoint	End point	No	-	Positive number Negative number 0	REAL[0-3] or _SGROUPPOS_INFO

S6	Velocity	Target velocity	No	-	Positive number	REAL
S7	Acceleration	Acceleration rate	No	-	Positive number	REAL
S8	Deceleration	Deceleration rate	Yes	Acceleration rate	Positive number	REAL
S9	PathChoice	Path choice 0: CW 1: CCW	Yes	0	0 to 1	INT
S10	CurveType	Velocity curve type 0: T-shaped velocity curve 1: 5-segment S-shaped curve Others: T-shaped velocity curve	Yes	0	0 to 1	INT
S11	AbsRelMode	Absolute or relative positioning mode 0: Absolute positioning 1: Relative positioning	Yes	0	0 to 1	INT
S12	BufferMode	Buffer mode 0: Aborting and no transition 1: Buffered and no transition 2: Previous velocity and no transition 3: Superimpose corners	Yes	0	0 to 3	INT
D1	Done	Target position reached ON after the target position is reached	Yes	OFF	ON OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON OFF	BOOL

D3	Active	Controlling ON when starting to execute the current curve segment	Yes	OFF	ON OFF	BOOL
D4	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D5	Error	Error flag	Yes	OFF	ON OFF	BOOL
D6	ErrorID	Error code	Yes	0	*1	INT



*1: For details, see Error Codes.

Function and Instruction Description

The MC_MoveCircular instruction performs circular interpolation on axis groups. It is active on the rising edge.

- Specifying the axis
  Group is latched on the rising edge of the Execute input.

  Modification on Group is invalid when Execute is ON.

  Modification on Group is invalid when Execute is OFF.
- Relationship with single-axis control instructions
  This instruction can be triggered only after the axes are switched to the StandStill state by executing the MC_Power instruction.

  This instruction is invalid if it is triggered during single-axis operations, such as jogging, torque control, homing, and stop.

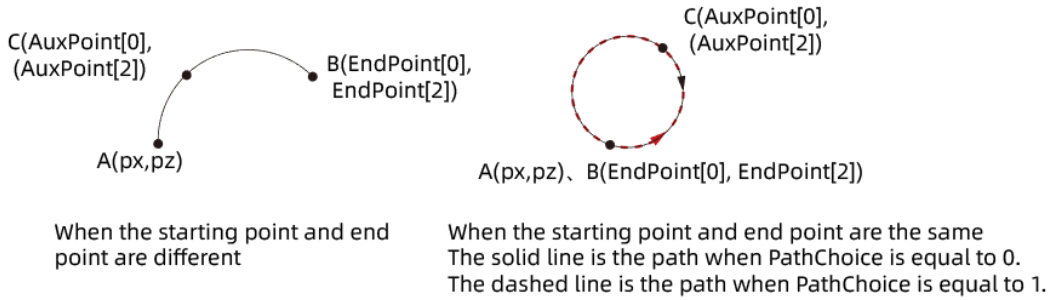
  After this instruction is triggered, the single-axis PLCOpen state machine is in SynchronizedMotion state. During operation, this instruction cannot be aborted by single-axis motion instructions. After the interpolation curve is completed, the single-axis PLCOpen state machine enters the StandStill state, and single-axis motion instructions can be executed at this time.
- Specifying circular axes
  CircAxes specifies the coordinate plane. To be specific:

  When CircAxes is set to 0, the x-y coordinate plane is selected. The motion axes specified by AxisID_x and AxisID_y perform circular interpolation, and the axes specified by AxisID_z and AxisID_a are auxiliary axes and perform linear interpolation.

  When CircAxes is set to 1, the y-z coordinate plane is selected. The motion axes specified by AxisID_y and AxisID_z perform circular interpolation, and the axes specified by AxisID_x and AxisID_a are auxiliary axes and perform linear interpolation.

When CircAxes is set to 2, the x-z coordinate plane is selected. The motion axes specified by AxisID_x and AxisID_z perform circular interpolation, and the axes specified by AxisID_y and AxisID_a are auxiliary axes and perform linear interpolation.

- Selecting the interpolation mode
  1. When CircMode is set to 0, circular interpolation is performed based on the border point.



When the x-y plane is selected, the border point is (AuxPoint[0], AuxPoint[1]) and the end point is (EndPoint[0], EndPoint[1]). When the y-z plane is selected, the border point is (AuxPoint[1], AuxPoint[2]) and the end point is (EndPoint[1], EndPoint[2]). When the x-z plane is selected, the border point is (AuxPoint[0], AuxPoint[2]) and the end point is (EndPoint[0], EndPoint[2]).

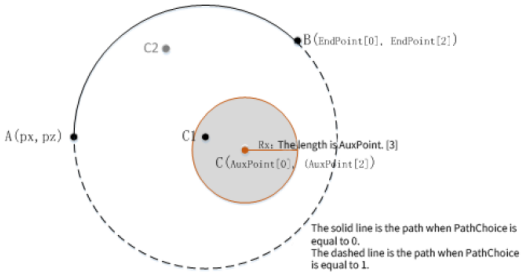
Take the x-y plane as an example. The start position of the x-axis is Px and that of the y-axis is Py. The instruction performs circular interpolation from the start point (Px, Py) through the border point (AuxPoint[0], (AuxPoint[1]) to the end point (EndPoint[0], EndPoint[1]) after it is triggered.

If the start point and the end point are the same point, a complete circle is drawn with the straight line between the start point (Px, Py) and the border point (AuxPoint[0], (AuxPoint[1]) as the diameter. In this case, PathChoice specifies the circular interpolation direction.

If the start point, border point, and end point are along the same line, they cannot form a circle. In this case, an error occurs and execution of the interpolation instruction is aborted.

If the start point and the border point are the same point, or the end point and the border point are the same point, an error occurs and execution of the interpolation instruction is aborted.

  2. When CircMode is set to 1, circular interpolation is performed based on the center.



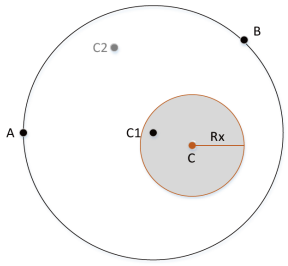
When the x-y plane is selected, the center is (AuxPoint[0], AuxPoint[1]) and the end point is (EndPoint[0], EndPoint[1]). When the y-z plane is selected, the center is (AuxPoint[1], AuxPoint[2]) and the end point is (EndPoint[1], EndPoint[2]). When the x-z plane is selected, the center is (AuxPoint[0], AuxPoint[2]) and the end point is (EndPoint[0], EndPoint[2]).

Take the x-z plane as an example. The start position of the x-axis is Px, and that of the z-axis is Pz. The instruction performs circular interpolation from the start point (Px, Pz) to the end point (EndPoint[0], EndPoint[2]) for the circle specified by the center point (AuxPoint[0], (AuxPoint[2]) after it is triggered. PathChoice specifies the circular interpolation direction.

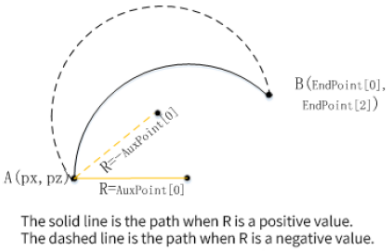
If the distance R1 from the specified center (AuxPoint[0], AuxPoint[2]) to the start point (Px, Pz) differs from the distance R2 to the end point (EndPoint[0], EndPoint[2]) (the difference between R1 and R2 greater than 1), the average value R of R1 and R2 is calculated as the radius, and the center (Cx, Cy) is

calculated in the same way as specifying the radius. Circular interpolation is performed using the calculated radius and center.

If two centers are calculated, calculate the distance between each of the two calculated centers and the set center and select the one closer to the set center. This point must be located inside the circle with (AuxPoint[0], AuxPoint[2]) as the center and AuxPoint[3] as the radius. As shown in the following figure, C1 is selected as the new center.



3. When CircMode is set to 2, circular interpolation is performed based on the specified radius.



No matter which plane is selected, the radius is always determined by |AuxPoint[0]|. When the x-y plane is selected, the end point is (EndPoint[0], EndPoint[1]). When the y-z plane is selected, the end point is (EndPoint[1], EndPoint[2]). When the x-z plane is selected, the end point is (EndPoint[0], EndPoint[2]).

Take the y-z plane as an example. The start position of the y-axis is Py and that of the z-axis is Pz. The instruction performs circular interpolation from the start point (Py, Pz) to the end point (EndPoint[1], EndPoint[2]) with the radius |AuxPoint[0]|.

If the sign of the radius is negative, a circle with a long arc is drawn. If the sign is positive, a circle with a short arc is drawn. PathChoice specifies the circular interpolation direction.

- Selecting the curve type

  If CurveType is set to 0, acceleration/deceleration is planned for axis group interpolation based on the T-shaped velocity curve.

  If CurveType is set to 1, acceleration/deceleration is planned for axis group interpolation based on the S-shaped velocity curve.

  If CurveType is set to Others, acceleration/deceleration is planned for axis group interpolation based on the T-shaped velocity curve.

- Selecting the positioning mode

  1. Absolute mode

     When the border point is selected, the auxiliary point and end point are absolute points in the coordinate system.

     When the center is selected, the center point and end point are absolute points in the coordinate system.

     When the radius is selected, the end point is the absolute point in the coordinate system.

  2. Relative mode

When the border point is selected, the auxiliary point and end point are relative points relative to the start point.

When the center is selected, the center point and end point are relative points relative to the start point.

When the radius is selected, the end point is the relative point relative to the start point.

- Buffer and transition
  Four optional buffer and transition modes are supported. For details, see the "Buffer and Transition" section in the *AutoShop Software Programming and Application Manual*.

  When buffer mode 1, 2, or 3 is selected, the interpolation instruction allows up to 8 curves to be buffered. When this instruction enters the buffer state, the Busy signal is active. When it is executed, the Active output becomes active. When the execution is completed, the Done signal output becomes active.

  When mode 0 (Aborting and no transition) is selected for a new interpolation instruction, it will abort all interpolation instructions that are being executed or buffered. The CommandAborted output of the aborted interpolation instructions becomes active.

- Re-execution
  This instruction cannot be re-executed.

  If this instruction is triggered repeatedly when the Busy output is ON, the axis reports error 9421 (instruction re-execution error), and all axes stop operation immediately and enter the ErrorStop state.

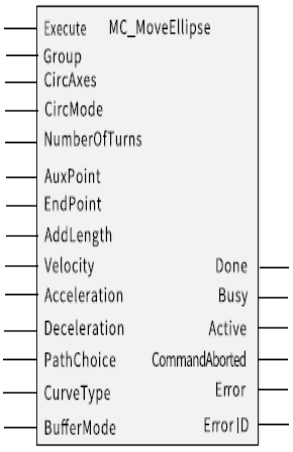
Timing Diagram

For details, see the timing diagram of the MC_MoveLinear instruction.

5.13.3.4 MC_MoveEllipse

MC_MoveEllipse: Elliptical interpolation

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_MoveEllipse	Elliptical interpolation		MC_MoveEllipse(Execute := ???, Group := ???, CircAxes := , CircMode := , NumOfTurns := , AuxPoint := ???, EndPoint := ???, AddLength := , Velocity := ???, Acceleration := ???, Deceleration := , PathChoice := , CurveType := , BufferMode := , Done => , Busy => ,

Instruction	Name	LD Expression	LiteST Expression
			Active => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	MC_MoveEllipse: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Group	Axis group	No	-	0 to 32767	INT
S2	CircAxes	Ellipse axis 0: x-y axis plane 1: y-z axis plane 2: x-z axis plane	Yes	0	0 to 2	INT
S3	CircMode	Elliptical interpolation mode 0: Complete ellipse 1: Specified arc length	Yes	0	0 to 2	INT
S4	NumOfTurns	Number of turns	Yes	1	1 to 100	INT
S5	AuxPoint	Center point	No	-	Positive number Negative number 0	REAL[0-3]
S6	EndPoint	End point coordinates	No	-	Positive number Negative number 0	REAL[0- 3] or _sGROUPPOS_INFO
S7	AddLength	Arc length during running when CircMode is 1	No	-	Negative number 0	REAL

					Positive number	
S8	Velocity	Target velocity	No	-	Positive number	REAL
S9	Acceleration	Acceleration	No	-	Positive number	REAL
S10	Deceleration	Deceleration	Yes	Acceleration	Positive number	REAL
S11	PathChoice	Path choice 0: CW 1: CCW	Yes	0	0 to 1	INT
S12	CurveType	Velocity curve type 0: T-shaped velocity curve 1: 5-segment S-curve Others: T-shaped velocity curve	Yes	0	0 to 1	INT
S13	BufferMode	Buffer mode 0: Aborting+No transition 1: Buffered+No transition 2: Previous velocity+No transition 3: Superimpose corners	Yes	0	0 to 3	INT
D1	Done	The target position is reached. TRUE after the target position is reached	Yes	FALSE	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	FALSE	ON/OFF	BOOL
D3	Active	Controlling TRUE when starting to execute the current curve segment	Yes	FALSE	ON/OFF	BOOL
D4	CommandAborted	Abortion of execution	Yes	FALSE	ON/OFF	BOOL
D5	Error	Error flag	Yes	FALSE	ON/OFF	BOOL
D6	ErrorID	Fault code	Yes	0	*1	INT

Function

Determination of Elliptic plane

By selecting the coordinate plane through CircAxes, you can draw an ellipse on the x-y plane, x-z plane, or y-z plane. After selecting the plane, the other two axes are used as auxiliary axes and linear interpolation is used.

Starting Point and Center Point of Ellipse

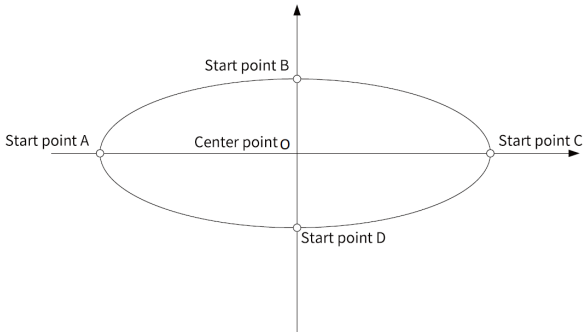
The starting point of the ellipse is the set position of the coordinate axis when the instruction is called, and the center point is (AuxPoint [0], AuxPoint [1]) specified by the instruction.

If you select the x-y plane, AuxPoint [0] is the X-axis coordinate and AuxPoint [1] is the Y-axis coordinate.

If you select the x-z plane, AuxPoint [0] is the X-axis coordinate and AuxPoint [1] is the Z-axis coordinate.

If you select the Y-z plane, AuxPoint [0] is the Y-axis coordinate and AuxPoint [1] is the Z-axis coordinate.

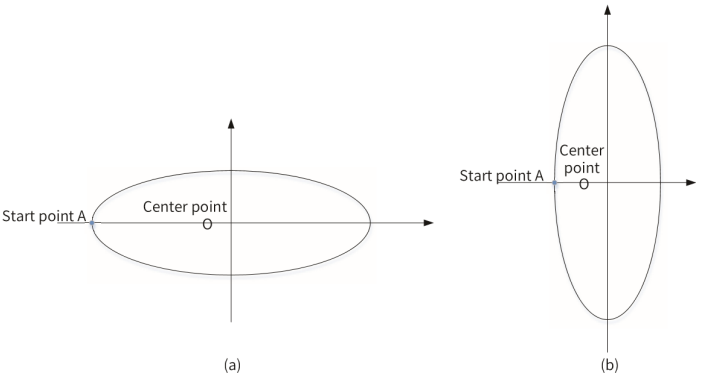
The line connecting the starting point and center point must be parallel to the coordinate axis. Taking the xy plane as an example, set the starting point coordinates to (x0, y0) and the center point coordinates to (x1, y1). When x0 = x1, the line connecting the two points is parallel to the Y-axis. If y0>y1, the starting point is point C in the figure below. If y0<y1, the starting point is point A in the figure below.



Determination of the major axis and minor axis

The parameters AuxPoint [2] and AuxPoint [3] in the instruction specify the lengths of the major and minor axes of the ellipse. When AuxPoint [2] > AuxPoint [3], AuxPoint [2] is the major axis, otherwise AuxPoint [3] is the major axis.

Assume that AuxPoint [2] = 12 and AuxPoint [3] = 5. If the distance of |AO| is equal to 12, the ellipse on the left is finally selected. If the distance of |AO| is equal to 5, the ellipse on the right is finally selected.



When the major axis equals the minor axis, the ellipse will become a circle.

Running Direction of the Ellipse

PathChoice specifies the running direction of the ellipse. When PathChoice is set to 0, it indicates clockwise running; When PathChoice is set to 1, it indicates counterclockwise running.

Determination of the End Point

CircMode determines the position of the end point.

When CircMode is 0, the running trajectory of the axis is a complete ellipse, where the end point is equal to the starting point.

When CircMode is 1, it is necessary to specify the arc length that is run from the starting point. When NumOfTurns is 0, PathChoice specifies the direction as CW (CCW). If AddLength is positive, the arc length specified by AddLength is run clockwise (counterclockwise). If AddLength is negative, the arc length specified by AddLength is run counterclockwise (clockwise). If NumOfTurns is greater than 0, run for (NumOfTurns – 1) turns in the direction specified by PathChoice, and then run at the arc length specified by AddLength on the last turn.

Curve shape

When Curvetype is set to 0, the T-shaped velocity curve is used for acceleration and deceleration planning in interpolation motion of axis groups.

When Curvetype is set to 1, the S-shaped velocity curve is used for acceleration and deceleration planning in interpolation motion of axis groups.

In other applications, the T-shaped velocity curve is used for acceleration and deceleration planning in interpolation motion of axis groups.

Processing of the auxiliary axis

When you select the x-y plane, the z-axis and a-axis are auxiliary axes. When you select the y-z plane, the x-axis and a-axis are auxiliary axes. When you select the x-z plane, the y-axis and a-axis are auxiliary axes.

In elliptical interpolation, the auxiliary axis will start running together with the elliptic coordinate axis and end the action together. When the elliptic coordinate axis is running, the auxiliary axis performs linear interpolation from the starting point to the end point, and the coordinates of the auxiliary axis at the end point are specified by EndPoint. When you select the x-y plane, the end point coordinate of the Z-axis is EndPoint[2], and the end point coordinate of the A-axis is EndPoint[3].

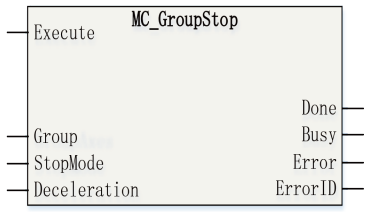
Fault code	Cause	Solution	Stop
9466	NumOfTurns in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9467	AddLength in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9468	The axis stops due to an error of the MC_MoveEllipse instruction.	Check the fault code of the MC_MoveEllipse instruction with an exception to locate the fault.	Yes
9469	CircAxes in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9470	CircMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9471	PathChoice in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes

Fault code	Cause	Solution	Stop
9472	Velocity in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9473	Acceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9474	Deceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9475	BufferMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9476	Failed to draw an ellipse because the center and the lengths of the major and minor axes are set incorrectly.	Ensure that the configured values of these parameters can form an ellipse.	Yes

5.13.3.5 MC_GroupStop

MC_GroupStop -- Stop axis group operation

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_GroupStop	Stop axis group operation		MC_GroupStop(Execute := ???, Group := ???, StopMode := , Deceleration := , Done => , Busy => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_MoveStop: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Group	Axis ID	No	-	0 to 32,767	INT
S2		Stop mode	Yes	1	0 to 1	INT

	StopMode	0: Decelerate to stop 1: Stop immediately				
S3	Deceleration	Deceleration rate	Yes	1000	Positive number, less than the maximum acceleration rate	REAL
D1	Done	Stop completed	Yes	OFF	OFF ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF ON	BOOL
D3	Error	Error flag	Yes	OFF	OFF ON	BOOL
D4	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Error Codes.

Function and Instruction Description

The MC_GroupStop instruction stops all axes in an axis group. It is active on the rising edge.

- Specifying the axis ID
  Group is latched on the rising edge of the Execute input.

  Modification on Group is invalid when Execute is ON.

  Modification on Group is invalid when Execute is OFF.
- Effective range
  The MC_GroupStop instruction can only stop interpolation instructions (such as MC_MoveLinear) but not single-axis motion instructions (such as MC_MoveAbsolute).

  The MC_Stop instruction can only stop single-axis motion instructions but not interpolation instructions.
- State transition
  On the rising edge of Execute, the interpolator performs the stop operation according to the stop mode specified by StopMode and aborts all buffered interpolation instructions. After the stop is completed, the Done signal output is active, and the single-axis PLCopen state machine is still in the SynchronizedMotion state.

  During the period when Execute is ON, the interpolator is always in the stop state, and a new interpolation instruction is inactive if it is triggered at this time.

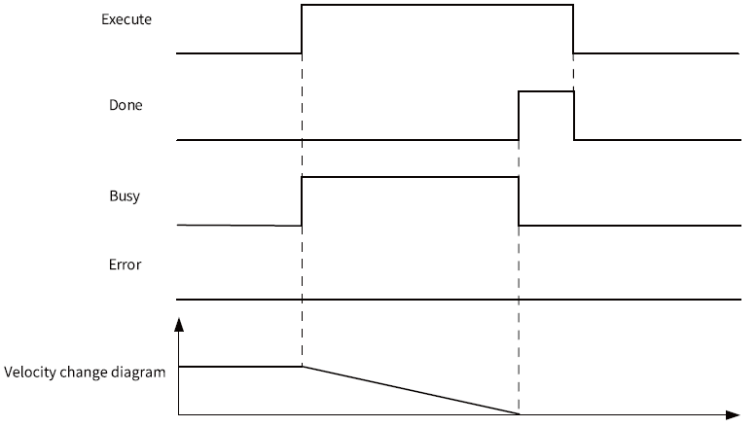
On the falling edge of Execute, the interpolator switches to a non-stop state, and each axis enters the StandStill state. At this time, a new interpolation instruction can be triggered.

- Stop mode
  When StopMode is set to 0, if CurveType in the running interpolation instruction is set to 0, the axes decelerate to stop based on the T-shaped curve with the deceleration rate specified by Deceleration. If CurveType is set to 1, the axes decelerate to stop based on the S-shaped curve with the deceleration rate specified by Deceleration.

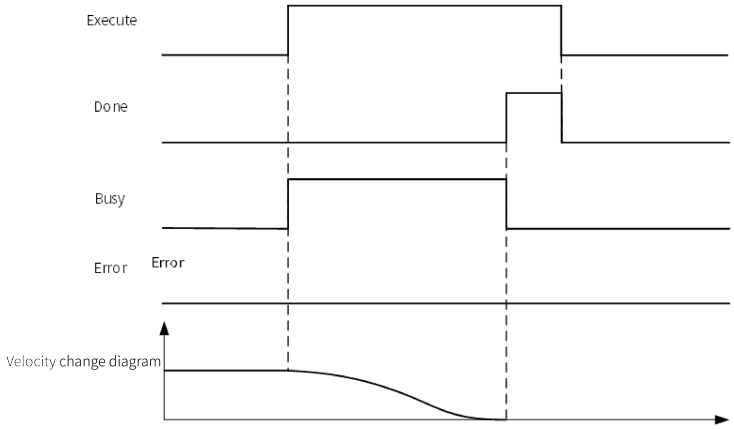
  When StopMode is set to 1, the axes stop immediately without deceleration.
- Re-execution
  When this instruction is re-triggered during axis deceleration, the axes in the axis group decelerate according to the new deceleration rate.
- Multi-execution
  This instruction does not support multi-execution. When an MC_GroupStop instruction is being executed and the Execute input is ON, if another MC_GroupStop instruction is triggered, the newly triggered instruction reports error 9441 (Axis in Stop state).

Timing Diagram

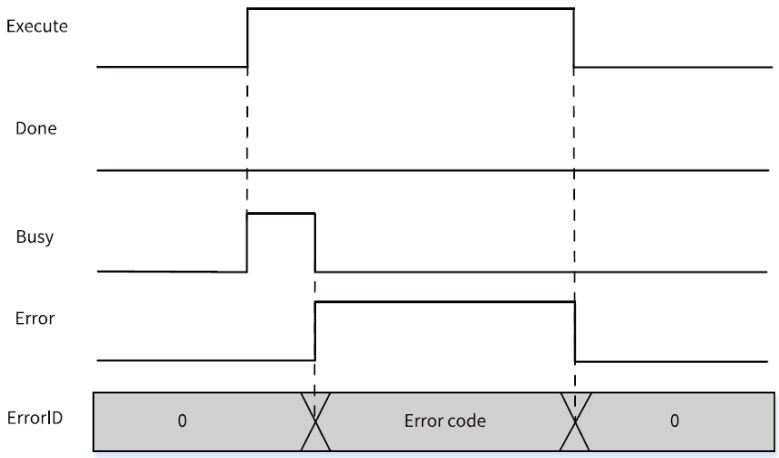
- The axes decelerate to stop normally based on the T-shaped curve.



- The axes decelerate to stop normally based on the S-shaped curve.



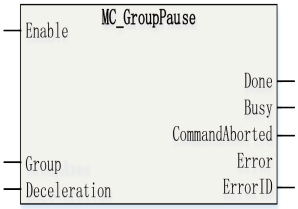
- An axis fails during deceleration.



5.13.3.6 MC_GroupPause

MC_GroupPause -- Pause axis group operation

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MC_GroupPause	Pause axis group operation		MC_GroupPause(Enable := ???, Group := ???, Deceleration := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_MoveCircular: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Group	Axis group	No	-	0 to 32,767	INT
S2	Deceleration	Deceleration rate	Yes	1000	Positive number, less than the maximum acceleration rate	REAL
D1	Done	Pause completed	Yes	OFF	OFF ON	BOOL

D2	Busy	Busy flag	Yes	OFF	OFF ON	BOOL
D3	CommandAborted	Abortion flag	Yes	OFF	OFF ON	BOOL
D3	Error	Error flag	Yes	OFF	OFF ON	BOOL
D4	ErrorID	Error code	Yes	0	*1	INT



*1: For details, see Error Codes.

Function and Instruction Description

The MC_GroupPause pauses all axes in an axis groups. It is active on the rising edge.

- Specifying the axis ID
  Group is latched on the rising edge of the Execute input.

  Modification on Group is invalid when Execute is ON.

  Modification on Group is invalid when Execute is OFF.
- Effective range
  The MC_GroupPause instruction can only pause interpolation instructions (such as MC_MoveLinear) but not single-axis motion instructions (such as MC_MoveAbsolute).
- Selecting the pause curve type

  If CurveType in the running interpolation instruction is set to 0, the axis decelerates to pause based on the T-shaped curve with the deceleration rate specified by Deceleration. If CurveType is set to 1, the axis decelerates to pause based on the S-shaped curve with the deceleration rate specified by Deceleration.
- State transition
  - When all axes in an axis group are in StandStill state:

    When Enable is set to ON, the axes in the axis group are still in the StandStill state. If a linear or circular interpolation instruction is triggered at this time, all axes in the axis group switch to the SynchronizedMotion state but remain paused and do not perform the interpolation algorithm until the Enable signal of the MC_GroupPause instruction becomes OFF.

  - When all axes in an axis group are in SynchronizedMotion state:

  - ▪ On the rising edge of Enable, the interpolator performs deceleration according to the deceleration rate specified by Deceleration. After deceleration is completed, the Done signal output becomes active, the single-axis PLCopen state machine is still in the SynchronizedMotion state, and the Busy

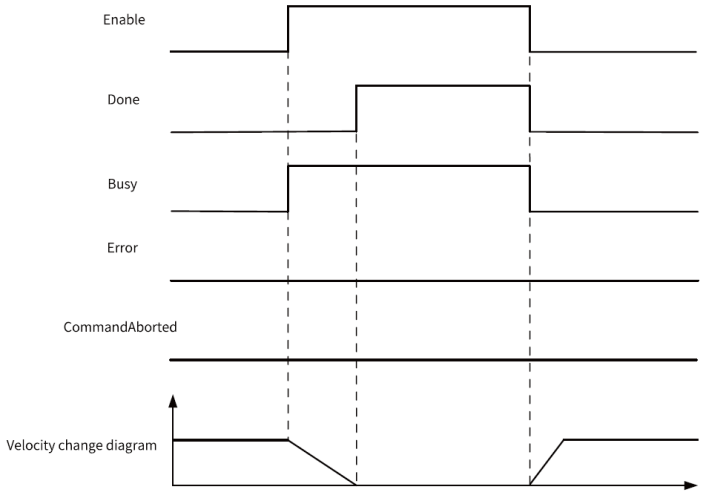
signal and Valid signal of the interpolation instruction being executed remain active during the pause period.

- During the period when the Enable signal is ON, the interpolator remains in paused state, and a new interpolation instruction is buffered if it is triggered at this time.
- On the falling edge of Enable, the interpolator resumes execution of the previously paused interpolation instruction.

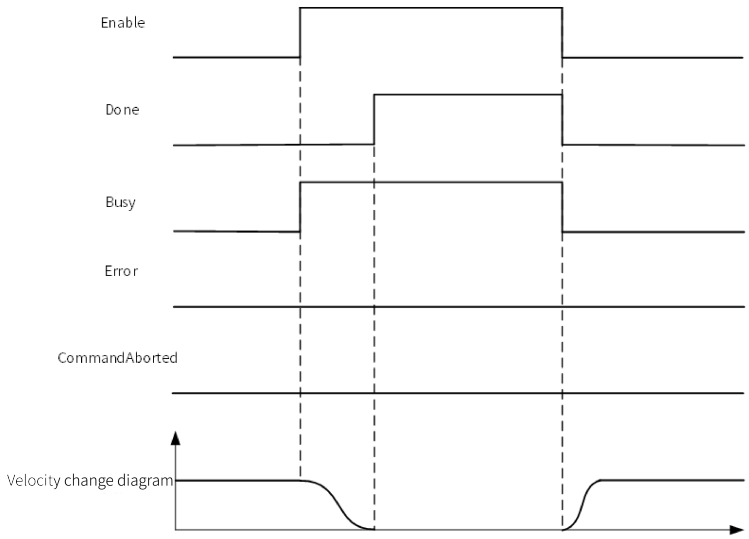
- Re-execution
  When this instruction is re-triggered during axis deceleration, the axes in the axis group decelerate according to the new deceleration rate.
- Multi-execution
  If an MC_GroupPause instruction is triggered during execution of another MC_GroupPause instruction, the first pause instruction is aborted, and the interpolator starts to decelerate according to the deceleration rate of the instruction triggered later.

Timing Diagram

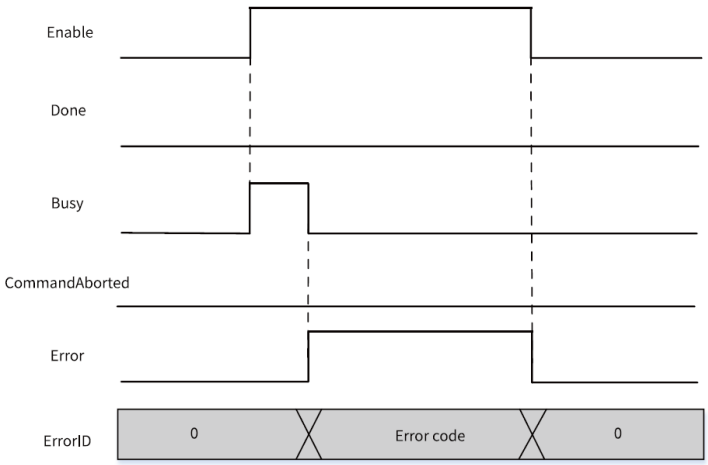
- When the acceleration/deceleration type specified in the interpolation instruction is T-shaped velocity curve during deceleration, axes can pause normally.



- When the acceleration/deceleration type specified in the interpolation instruction is S-shaped velocity curve during deceleration, axes can pause normally.



- An axis fails during deceleration.



5.13.3.7 Fault Codes

The following table lists the fault codes, fault information, and solutions when a fault occurs during use of the interpolation functions.

Fault code	Fault information	Solution
9400	Maximum axis group quantity exceeded	Check whether the number of axis groups is greater than 8.
9401	An axis in the axis group is faulty.	Check whether an axis in the axis group has entered the ErrorStop state. Locate the fault based on the fault code of each axis.
9402	The number of buffered interpolation instructions is greater than 8.	Check whether the number of buffered interpolation instructions is greater than 8.
9403	Axis reused	Locate the reused axis and replace it with an unused axis.
9404	Failed to create axis group	The x-axis and y-axis cannot be empty. Check whether the x-axis or y-axis does not exist or is not specified.
9405	The specified z-axis does not exist.	Check whether the axis specified by AxisID_z exists.
9406	Specified auxiliary axis non-existent	Check whether the axis specified by AxisID_a exists.
9407	Axis group ID duplicated	Check whether GroupID is duplicate.
9408	Failed to configure the axis.	Check whether any axis in the axis group fails to be configured. If yes, check whether the board software and the software tool match.
9409	The axis ID is less than 0.	Check whether the ID of an axis in the axis group is less than 0.

Fault code	Fault information	Solution
9410	The axis group is not released because the same MC_SetAxesGroup instruction is triggered repeatedly in a short time period.	Do not re-trigger the MC_SetAxesGroup instruction while its Busy signal output is still active.
9411	The MC_GroupStop instruction is aborted.	Check whether an instruction with higher priority is called while the MC_GroupStop instruction is still active.
9412	The value of CircAxes of the circular interpolation instruction is out of range.	Check whether the value of CircAxes of the circular interpolation instruction is out of range.
9413	The value of CircMode of the circular interpolation instruction is out of range.	Check whether the value of CircMode of the circular interpolation instruction is out of range.
9414	The value of PathChoice of the circular interpolation instruction is out of range.	Check whether the value of PathChoice of the circular interpolation instruction is out of range.
9415	The value of StopMode of the stop instruction is out of range.	Check whether the value of StopMode of the stop instruction is out of range.
9416	The x-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9417	The y-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9418	The z-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9419	The auxiliary axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9420	The circular interpolation instruction is triggered repeatedly.	Do not re-trigger the same circular interpolation instruction while its Busy signal output is still active.
9421	The linear interpolation instruction is triggered repeatedly.	Do not re-trigger the same linear interpolation instruction while its Busy signal output is still active.
9422	Failed to obtain the axis group.	Check whether the axis group specified by GroupID has been created by calling MC_SetAxesGroup.

Fault code	Fault information	Solution
9423	Failed to configure the axis.	Check whether an instruction is triggered when axis configuration is not completed. Check whether the communication state of all axes in the axis group is Axis ready.
9424	An axis is disabled.	Do not call the interpolation instruction when any axis is in Disabled state.
9425	Axis in execution of single-axis motion instruction	Do not call the interpolation instruction when any axis is executing single-axis motion instructions and not in StandStill state.
9426	An axis is in Stopping state.	Do not call the interpolation instruction when any axis is in Stopping state after executing the MC_Stop instruction.
9427	The axis group is in Stopping state.	Do not call the interpolation instruction while the MC_GroupStop instruction is still active.
9428	An axis is in Homing state.	Do not call the interpolation instruction when any axis is in Homing state after executing the MC_Home instruction.
9429	An axis is executing the position setting instruction.	Do not call the interpolation instruction when any axis is setting the current position by executing the MC_SetPosition instruction.
9430	An axis is in commissioning state.	Do not call the interpolation instruction when any axis is in commissioning state.
9431	An axis enters the commissioning state during interpolation, which aborts instruction execution of other axes.	Check whether any axis enters the commissioning state during interpolation.
9432	Failed to request the memory.	• Check whether the memory runs out. • Contact the manufacturer.
9433	The target velocity is 0 or less than 0.	Ensure that the target velocity of the instruction is greater than 0.
9434	The target acceleration is 0 or less than 0.	Ensure that the target acceleration of the instruction is greater than 0.
9435	The target deceleration is 0 or less than 0.	Ensure that the target deceleration of the instruction is greater than 0.
9436	The curve type setting is out of range.	Check whether the curve type is set to a value other than the T-shaped curve for the interpolation instruction.

Fault code	Fault information	Solution
9437	AbsRelMode is set incorrectly.	Check whether the parameter is set to a value other than the absolute positioning and relative positioning modes.
9438	BufferMode is set incorrectly.	Check whether the value of BufferMode is out of range.
9439	InsertMode is set incorrectly.	Check whether the value of InsertMode is set properly.
9440	An axis stops due to a fault.	Locate the faulty axis and rectify the fault based on the fault code.
9441	The MC_GroupStop instruction is called repeatedly.	Do not re-trigger an MC_GroupStop instruction or call other MC_GroupStop instructions while an MC_GroupStop instruction is still active.
9442	Data buffer area not empty	Contact Inovance for technical support.
9443	Not a circle	-
9444	The start, end, and border points in the circular interpolation instruction are the same point, and no circle can be drawn.	Check the input parameters of the circular interpolation instruction and ensure that the start, end, and border points can form a circle.
9445	The instruction buffer area is full.	Contact Inovance for technical support.
9446	The velocity of the x-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the x-axis is not greater than the maximum allowable velocity.
9447	The velocity of the y-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the y-axis is not greater than the maximum allowable velocity.
9448	The velocity of the z-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the z-axis is not greater than the maximum allowable velocity.
9449	The velocity of the auxiliary axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the auxiliary axis is not greater than the maximum allowable velocity.
9450	Failed to obtain the number of axis groups.	Update the software tool to the latest version.
9451	Internal fault	Contact the manufacturer.
9452	The instruction is called when the axis is in StandStill state.	Do not call this instruction when the axis is StandStill state.
9453	Maximum velocity exceeded	Ensure that the target velocity of the instruction is not greater than the maximum velocity specified on the axis group configuration interface.

Fault code	Fault information	Solution
9454	The maximum allowable acceleration (deceleration) is exceeded.	Ensure that the target acceleration (deceleration) of the instruction is not greater than the maximum acceleration (deceleration) specified on the axis group configuration interface.
9455	The axis group becomes faulty due to an error reported by the linear interpolation instruction.	Identify the first linear interpolation instruction that reports the error and troubleshoot the fault based on the fault code.
9456	The axis group becomes faulty due to an error reported by the circular interpolation instruction.	Identify the first circular interpolation instruction that reports the error and troubleshoot the fault based on the fault code.
9457	The axis group becomes faulty due to an error reported by the axis group stop instruction.	Identify the first axis group stop instruction that reports the error and troubleshoot the fault based on the fault code.
9458	The axis group becomes faulty due to an error reported by the axis group pause instruction.	Identify the first axis group pause instruction that reports the error and troubleshoot the fault based on the fault code.
9459	The x-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9460	The y-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9461	The z-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9462	The auxiliary axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9463	When the MC_GroupStop instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the MC_GroupStop instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the MC_GroupStop instruction when the axes enter the synchronous mode due to other instructions.

Fault code	Fault information	Solution
9464	When the linear or circular interpolation instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the linear or circular interpolation instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the linear or circular interpolation instruction when the axes enter the synchronous mode due to other non-axis-group instructions.
9465	When the MC_GroupPause instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the MC_GroupPause instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the MC_GroupPause instruction when the axes enter the synchronous mode due to other instructions.
9466	NumOfTurns in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9467	AddLength in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9468	The axis stops due to an error of the MC_MoveEllipse instruction.	Check the fault code of the MC_MoveEllipse instruction with an exception to locate the fault.
9469	CircAxes in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9470	CircMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9471	PathChoice in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9472	Velocity in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9473	Acceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9474	Deceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9475	BufferMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9476	Failed to draw an ellipse because the center and the lengths of the major and minor axes are set incorrectly.	Ensure that the parameter value is within the allowable range.

Fault code	Fault information	Solution
9477	The property of the x-axis in the axis group instruction does not support the interpolation motion.	Ensure that the x-axis is not in single-axis mode.
9478	The property of the y-axis in the axis group instruction does not support the interpolation motion.	Ensure that the y-axis is not in single-axis mode.
9479	The property of the z-axis in the axis group instruction does not support the interpolation motion.	Ensure that the z-axis is not in single-axis mode.
9480	The property of the auxiliary axis in the axis group instruction does not support the interpolation motion.	Ensure that the auxiliary axis is not in single-axis mode.

5.14 MC Axis Control (CANopen)

5.14.1 List of Instructions

The following table lists the CANopen axis control instructions.

Category	Instruction	Function
CANopen axis control instruction	MC_Power_CO	Enable servo axis through communication
	MC_Reset_CO	Reset servo axis fault through communication
	MC_ReadActualPosition_CO	Read current position of axis through communication
	MC_ReadActualVelocity_CO	Read current velocity of axis through communication
	MC_Halt_CO	Halt servo axis through communication
	MC_Stop_CO	Stop servo axis through communication
	MC_MoveAbsolute_CO	Control absolute positioning of axis through communication
	MC_MoveRelative_CO	Control relative positioning of axis through communication

Category	Instruction	Function
	MC_MoveVelocity_CO	Control axis velocity operation mode through communication
	MC_Jog_CO	Control axis jogging through communication
	MC_Home_CO	Control axis homing through communication
	MC_WriteParameter_CO	Write axis parameters through communication
	MC_ReadParameter_CO	Read axis parameters through communication

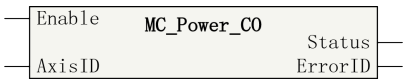
5.14.2 MC_Power_CO

Description

Control servo axis enable or disable.

MC_Power_CO: Enable servo axis through communication

Graphic Block



The instruction formats are as follows.

16-bit instruction	MC_Power_CO: Continuous execution					
32-bit instruction	-					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
D1	Status	Axis status	Yes	OFF	ON/OFF	BOOL
D2	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Error Codes of CANopen Axis Control Instructions.

See the following table for soft elements.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S	-	-	-	-	-	-	√	-	-
D1	√[1]	-	√	-	√	√	-	-	-
D2	-	-	√	√	√	√	-	-	-

NOTE

- [1] The X element is not supported.
- The MC_Power_CO instruction can be executed only once for each axis.

Function and Instruction Description

“AxisID” specifies the ID of the controlled axis, which ranges from K1 to K16.

“Status” specifies the actual state output of the axis. “ON” indicates that the axis is enabled, and “OFF” indicates that the axis is disabled.

“ErrorID” specifies the error code. For details, see “Description of Instruction Error Codes”.

The MC_Power_CO instruction writes to the corresponding control word (6040h) based on the read status word (6041h) to enable the axis. The following table describes the correspondence between the status word (6041h) and the control word (6040h).

Energy flow state	Status word (6041h)		Control word (6040h)	
ON	Not ready to switch on	xxxx xxxx x0xx 0000 b	Shutdown	0000 0000 0000 0110 b
	Switch on disabled	xxxx xxxx x1xx 0000 b		
	Ready to switch on	xxxx xxxx x01x 0001 b	Switch on	0000 0000 0000 0111 b
	Switched on	xxxx xxxx x01x 0011 b	Switch on + enable operation	0000 0000 0000 1111 b
	Fault reaction active		-	

Energy flow state	Status word (6041h)		Control word (6040h)	
		xxxx xxxx x0xx 1111 b		xxxx xx00 xx00 xxxx b
	Fault	xxxx xxxx x0xx 1000 b		
	Others		-	xxxx xxxx xxxx xxxx b
OFF	Ready to switch on	xxxx xxxx x01x 0001 b	Disable voltage	0000 0000 0000 0000 b
	Switched on	xxxx xxxx x01x 0011 b		
	Operation enabled	xxxx xxxx x01x 0111 b		
	Others		-	xxxx xx00 xx00 xxxx b

In this table, x indicates any value (status word) or remains unchanged (control word).

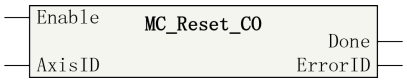
5.14.3 MC_Reset_CO

Description

This instruction resets errors of an axis and makes the axis enter the StandStill or Disabled state.

MC_Reset_CO: Reset servo axis fault through communication

Graphic Block



The instruction formats are as follows.

16-bit instruction	MC_Reset_CO: Continuous execution
32-bit instruction	-

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL

D2	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see Error Codes of CANopen Axis Control Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Function and Instruction Description

This instruction resets faults of a CANopen bus axis and makes the axis enter the StandStill or Disabled state.

“AxisID” specifies the ID of the controlled axis, which ranges from K1 to K16.

"Done" indicates the output after the reset operation is completed.

"ErrorID" specifies the error code. For details, see “Description of Instruction Error Codes”.

The MC_Reset_CO instruction writes to the corresponding control word (6040h) based on the read status word (6041h) to reset the axis fault. The following table describes the correspondence between the status word (6041h) and the control word (6040h).

Energy flow state	Status word (6041h)		Control word error reset (6040h.bit7)
ON	Switch on disabled	xxxx xxxx x1xx 0000b	0
	Operation enabled	xxxx xxxx x01x 0111b	-
	Fault	xxxx xxxx x0xx 1000b	1
	-	Others	x
↑	-	xxxx xxxx xxxx xxxxb	0

Energy flow state	Status word (6041h)		Control word error reset (6040h.bit7)
OFF	-	xxxx xxxx xxxx xxxxb	x

In this table, x indicates any value (status word) or remains unchanged (control word).

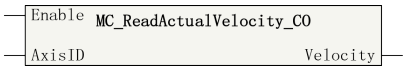
5.14.4 MC_ReadActualVelocity_CO

Introduction

The MC_ReadActualVelocity_CO instruction reads the current velocity of an axis.

MC_ReadActualVelocity_CO -- Read current velocity of axis through communication

Graphic Block



The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_ReadActualVelocity_CO: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANopen axis to be operated	No	-	1 to 16	INT
D1	Velocity	Current velocity	Yes	0	-	REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The MC_ReadActualVelocity_CO instruction reads the actual velocity of a CANopen bus axis.

AxisID specifies the ID of the axis to be read, which ranges from 1 to 16.

Position specifies the current position of the axis, which is a 32-bit floating-point number.

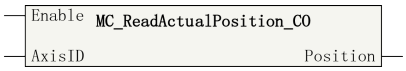
5.14.5 MC_ReadActualPosition_CO

Introduction

The MC_ReadActualPosition_CO instruction reads the current position of an axis.

MC_ReadActualPosition_CO -- Read current position of axis through communication

Graphic Block



The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_ReadActualPosition_CO: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANopen axis to be operated	No	-	1 to 16	INT
D1	Position	Current position	Yes	0	-	REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The MC_ReadActualPosition_CO instruction reads the actual position of a CANopen bus axis.

AxisID specifies the ID of the axis to be read, which ranges from 1 to 16.

Position specifies the current position of the axis, which is a 32-bit floating-point number.

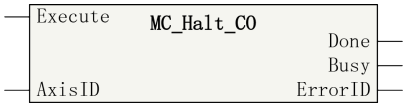
5.14.6 MC_Halt_CO

Introduction

The MC_Halt_CO instruction halts the current motion of an axis so that the axis can respond to other motion instructions.

MC_Halt_CO -- Halt servo axis through communication

Graphic Block



The following table describes the instruction format.

16-bit Instruction	MC_Halt_CO: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis ID	No	-	-	INT
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	ErrorID	Error code	Yes	-	*1	INT

NOTE

*1: For details, see Error Codes of CANopen Axis Control Instructions.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D3	-	-	-	√	√	√	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description

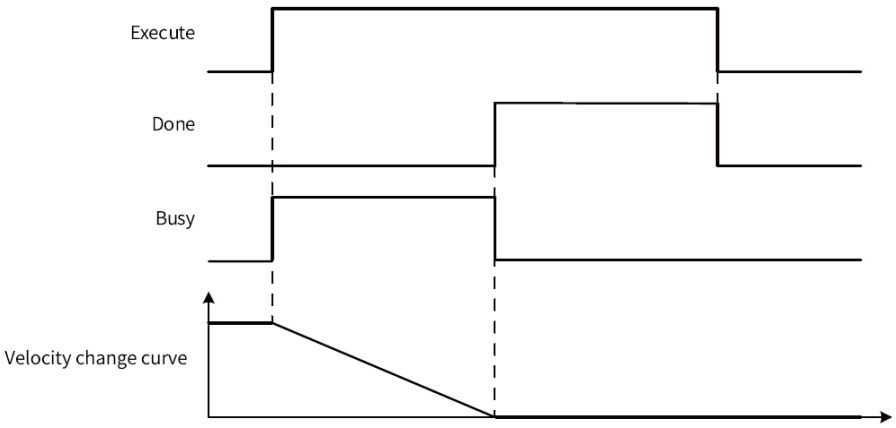
The MC_Halt_CO instruction halts the current motion of a CANopen bus axis so that the axis can respond to other motion instructions.

The MC_Halt_CO instruction can be aborted by instructions including MC_MoveAbsolute_CO, MC_MoveRelative_CO, MC_MoveVelocity_CO, and MC_Jog_CO.

The following table describes the operation procedure of the MC_Halt_CO instruction on a CANopen object.

Step	Operation/Condition	Description
1	6040h.bit4 = 0	Trigger the motion halt operation using the control word. Write 0 to the target velocity.
	6040h.bit5 = 0	
	6040h.bit6 = 0	
	6040h.bit8 = 1	
	60FFh = 0	
2	606Ch = 0	Wait until the halt operation is completed.
	6061h = 3 and 6041h.bit13 = 1	
	6061h != 3 and 6041h.bit10 = 1	
3	6060h = 1	Switch to the position mode.

Timing Diagram



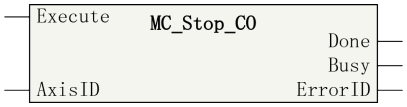
5.14.7 MC_Stop_CO

Introduction

The MC_Stop_CO instruction stops an axis and makes it enter the Stopping state so that the axis no longer responds to any motion instruction.

MC_Stop_CO -- Stop servo axis through communication

Graphic Block



The following table describes the instruction format.

16-bit Instruction	MC_Stop_CO: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANopen axis to be operated	No	-	1 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	ErrorID	Error code	Yes	0	*1	INT

> **NOTE**
>
> *1: For details, see Error Codes of CANopen Axis Control Instructions.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

> **NOTE**
>
> [1]: The X element is not supported.

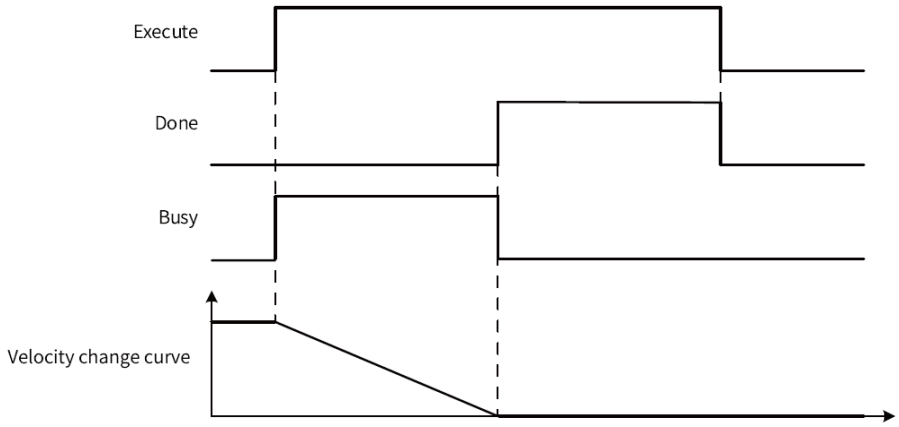
Function and Instruction Description

The MC_Stop_CO instruction stops a CANopen bus axis and makes it enter the Stopping state so that the axis no longer responds to any motion instruction.

The following table describes the operation procedure of the MC_Stop_CO instruction on a CANopen object.

Step	Operation/Condition	Description
1	6040h.bit4 = 0	Trigger the motion halt operation using the control word. Write 0 to the target velocity.
	6040h.bit5 = 0	
	6040h.bit6 = 0	
	6040h.bit8 = 1	
	60FFh = 0	
2	606Ch = 0	Wait until the halt operation is completed.
	6061h = 3 and 6041h.bit13 = 1	
	6061h != 3 and 6041h.bit10 = 1	
3	6060h = 1	Switch to the position mode.

Timing Diagram

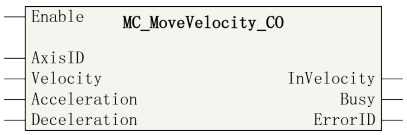


5.14.8 MC_MoveVelocity_CO

Description

MC_MoveVelocity_CO: Control axis velocity through communication

Graphic Block



The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	C_MoveVelocity_CO: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	AxisID	ID of the CANopen axis to be operated	No	-	1 to 16	INT
S2	Velocity	Velocity	No	-	-	REAL
S3	Acceleration	Acceleration	No	-	-	REAL
S4	Deceleration	Deceleration	Yes	Acceleration	-	REAL
D1	InVelocity	Velocity reached	Yes	OFF	ON/OFF	BOOL

D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see Error Codes of CANopen Axis Control Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Function and Instruction Description

This instruction controls a CANopen bus axis to move at the specified velocity. When the specified velocity ("Velocity") is greater than 0, the axis moves in the forward direction; when it is less than 0, the axis moves in the reverse direction. The velocity can be modified during execution of this instruction, and the modification takes effect in real time.

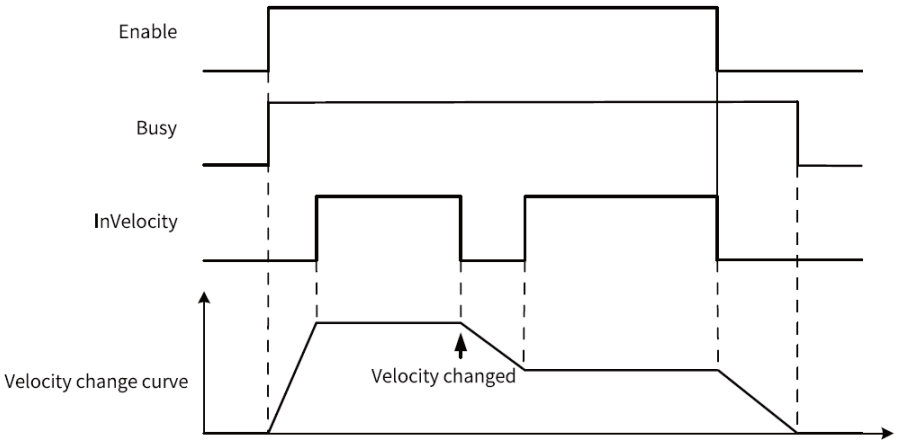
If "Deceleration" is not specified, that is, it is left empty, the specified acceleration ("Acceleration") is used as the deceleration by default.

Operation procedure of the MC_MoveVelocity_CO instruction on a CANopen object

Procedure	Operation/Condition	Description
1	6040h.bit8 = 0	Reset control word halt bit

Procedure	Operation/Condition	Description
2	6083h = Acceleration	Write acceleration
3	6084h = Deceleration	Write deceleration
4	6060h = 3	Switch to speed mode
5	6061h = 3	Wait for the completion of speed mode switching
6	60FFh = Target speed	Sets target speed
	6041h.bit10 = 1	Target speed reach
	60FFh < 0 and 6041h.bit11 = 1 and 60FDh.bit0 = 1: 60FFh = 0	The negative limit is reached, and motion ends.
	607Ah > 0 and 6041h.bit11 = 1 and 60FDh.bit1 = 1: 60FFh = 0	The positive limit is reached, and motion ends.
	60FFh = 0	The instruction flow becomes inactive, and motion ends.

Sequence Diagram

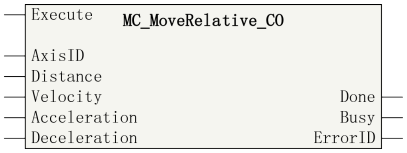


5.14.9 MC_MoveRelative_CO

Description

MC_MoveRelative_CO: Control relative positioning of axis through communication

Graphic Block



The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	MC_MoveRelative_CO: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	AxisID	ID of the CANopen axis to be operated	No	-	1 to 16	INT
S2	Distance	Target distance	No	-	-	REAL
S3	Velocity	Maximum velocity	No	-	-	REAL
S4	Acceleration	Acceleration	No	-	-	REAL
S5	Deceleration	Deceleration	Yes	Acceleration	-	REAL
D1	Done	Completion flag, indicating that the target position is reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see Error Codes of CANopen Axis Control Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S5	-	-	-	√	√	√	-	√	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Function

This instruction implements the relative positioning function of a CANopen bus axis. It controls an axis to move from the current position for a specified distance.

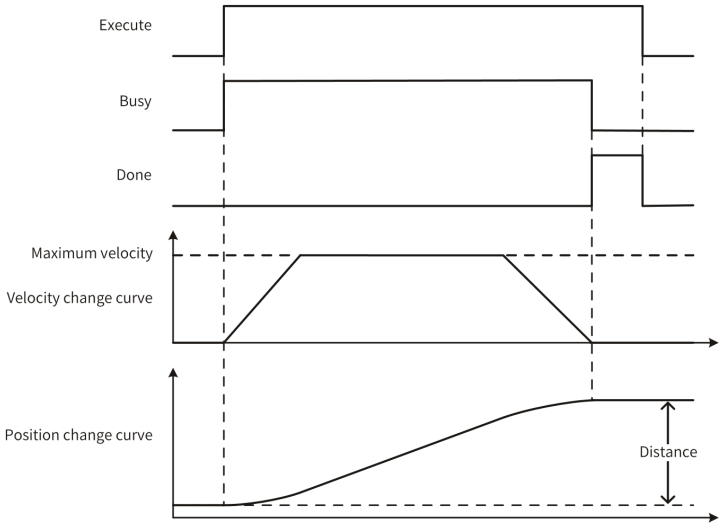
If "Deceleration" is not specified, that is, it is left empty, the specified acceleration ("Acceleration") is used as the deceleration by default.

Operation procedure of the MC_MoveRelative_CO instruction on a CANopen object

Procedure	Operation/Condition	Description
1	6060h = 1	Switch to position mode
2	6061h = 1	Wait for the completion of position mode switching
3	6040h.bit5 = m	Control word write related mode. Cache mode (parameter number: K1000) = 0, then m = 1; otherwise, m = 0.
	6040h.bit6 = 1	
	6040h.bit8 = 0	
	6040h.bit9 = 0	
4	607Ah = Position	Write (relative) target position, positioning speed
	6081H = Speed	
5	6083h = Acceleration	Write acceleration
6	6084h = Deceleration	Write deceleration
7	6040h.bit4 = 1	Trigger positioning

Procedure	Operation/Condition	Description
8	6041h.bit12 = 1	Wait for positioning to start
9	6040h.bit4 = 0	Reset positioning trigger
10	607Ah < 0 and 6041h.bit11 = 1 and 60FDh.bit0 = 1	When the negative movement meets the negative limit, the positioning ends
	607Ah > 0 and 6041h.bit11 = 1 and 60FDh.bit1 = 1	When the forward movement meets the positive limit, the positioning ends.
	6041h.bit10 = 1 and 6041h.bit12 = 0	Target position reached and positioning completed

Sequence Diagram

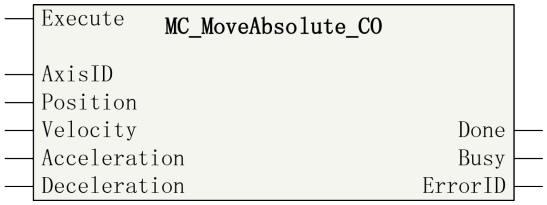


5.14.10 MC_MoveAbsolute_CO

Description

MC_MoveAbsolute_CO: Control absolute positioning of axis through communication

Graphic Block



The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	MC_MoveAbsolute_CO: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	AxisID	ID of the CANopen axis to be operated	No	-	1 to 16	INT
S2	Position	Target position	No	-	-	REAL
S3	Velocity	Maximum velocity	No	-	-	REAL
S4	Acceleration	Acceleration	No	-	-	REAL
S5	Deceleration	Deceleration	Yes	Acceleration	-	REAL
D1	Done	Completion flag, indicating that the target position is reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1. For details, see Error Codes of CANopen Axis Control Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	-	√	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Function

This instruction implements the absolute positioning function of a CANopen bus axis. It controls an axis to move to the specified position.

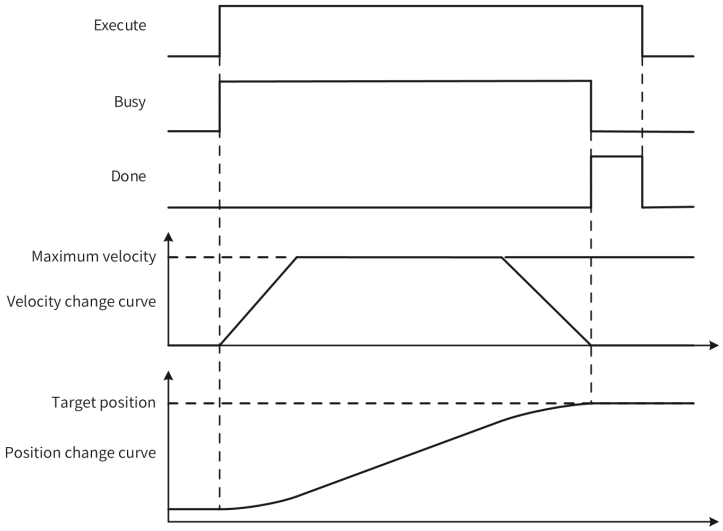
If "Deceleration" is not specified, that is, it is left empty, the specified acceleration ("Acceleration") is used as the deceleration by default.

For MC_MoveAbsolute_CO instruction CANOpen object operation steps, see the following table.

Method	Operation/Condition	Description
1	6060h = 1	Switch to position mode
2	6061h = 1	Wait for the completion of position mode switching
3	6040h.bit5 = m	Control word write related mode. Cache mode (parameter number: K1000) = 0, then m = 1; otherwise, m = 0.
	6040h.bit6 = 0	
	6040h.bit8 = 0	
	6040h.bit9 = 0	
4	607Ah = Position	Write (absolute) target position, positioning speed
	6081H = Speed	
5	6083h = Acceleration	Write acceleration
6	6084h = Deceleration	Write deceleration

Method	Operation/Condition	Description
7	6040h.bit4 = 1	Trigger positioning
8	6041h.bit12 = 1	Wait for positioning to start
9	6040h.bit4 = 0	Reset positioning trigger
10	607Ah < 6064h and 6041h.bit11 = 1 and 60FDh.bit0 = 1:	When the negative movement meets the negative limit, the positioning ends
	607Ah > 6064h and 6041h.bit11=1 and 60FDh.bit1=1	When the forward movement meets the positive limit, the positioning ends.
	6041h.bit10 = 1 and 6041h.bit12 = 0	Target position reached and positioning completed

Sequence Diagram

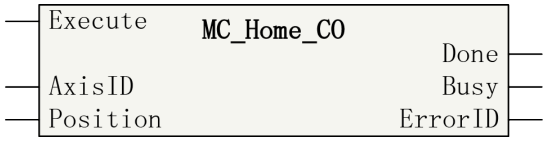


5.14.11 MC_Home_CO

Description

MC_Home_CO: Control axis homing through communication

Graphic Block



The instruction formats are as follows.

16-bit instruction	-

32-bit instruction	MC_Home_CO: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	AxisID	ID of the CANopen axis to be operated	No	-	1 to 16	INT
S2	Position	Target position after homing	Yes	0	-	REAL
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see Error Codes of CANopen Axis Control Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Function and Instruction Description

This function implements homing of a CANopen bus axis.

You need to set the homing mode and velocity on the CANopen configuration interface. See the relevant manuals of servo/motor drives for the description of each homing mode.

Operation procedure of the MC_Home_CO instruction on a CANopen object

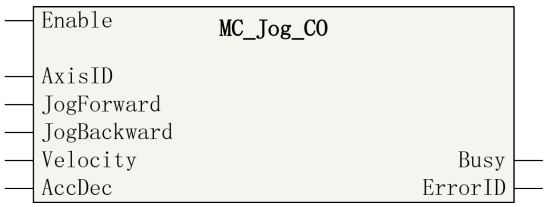
Method	Operation/Condition	Description
1	6060h = 6	Switch to homing mode
2	6061h = 6	Wait for the completion of homing mode switching
3	607Ch= origin offset	Sets origin offset
4	6040h.bit4 = 1	Start homing
5	6041h.bit10 = 1 and 6041h.bit13 = 1	Homing failed
	6041h.bit10 = 1 and 6041h.bit12 = 1	Homing successful

5.14.12 MC_Jog_CO

Description

MC_Jog_CO: Control axis jogging through communication

Graphic Block



The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	MC_Jog_CO: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	AxisID	ID of the CANopen axis to be operated	No	-	1 to 16	INT
S2	JogForward		No	-	ON/OFF	BOOL

		Jogging in forward direction, triggered on the rising edge				
S3	JogBackward	Jogging in reverse direction, triggered on the rising edge	No	-	ON/OFF	BOOL
S4	Velocity	Target speed	No	-	-	REAL
S5	AccDec	Acceleration/ deceleration	No	-	-	REAL
D1	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D2	ErrorID	Fault code	Yes	0	*1	INT

NOTE

*1: For details, see Error Codes of CANopen Axis Control Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	-	√	-	-	√	-	-	-
S3	√	-	√	-	-	√	-	-	-
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	-	√	-
D1	√[1]	-	√	-	-	√	-	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D2	-	-	-	√	√	√	-	-	-

NOTE

1: No

Function

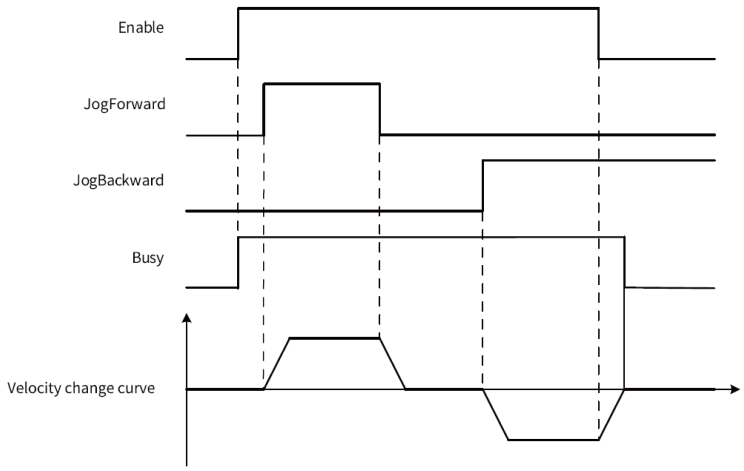
This instruction implements the jogging function of a CANopen bus axis. When JogForward is active, the axis moves forward at the velocity specified by "Velocity"; when JogBackward is active,

the axis moves in reverse direction at the velocity specified by "Velocity". When both JogForward and JogBackward are active, the axis stops motion.

The following table shows the operation procedure of the MC_Jog_CO instruction on a CANopen object.

Procedure	Operation/Condition	Description
1	6040h.bit8 = 0	Reset control word halt bit
2	6083h = acceleration/deceleration	Write acceleration
3	6084h = acceleration/deceleration	Write deceleration
4	6060h = 3	Switch to speed mode
5	6061h = 3	Wait for the completion of speed mode switching
6	Forward jog: 60FFh = Target speed Reverse jog: 60FFh = - Target speed Other: 60Ffh = 0	FWD/REV jog
	60FFh < 0 and 6041h.bit11 = 1 and 60FDh.bit0 = 1: 60FFh = 0	When the negative movement meets the negative limit, the Jog ends
	607Ah > 6040h and 6041h.bit11 = 1 and 60FDh.bit1 = 1: 60FFh = 0	When the positive movement meets the forward limit, the Jog ends
	60FFh = 0	Inactive instruction energy flow, end of Jog

Sequence Diagram

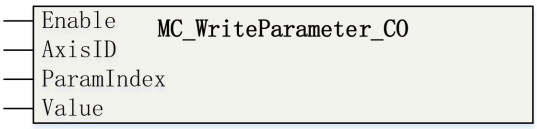


5.14.13 MC_WriteParameter_CO

Introduction

MC_WriteParameter_CO -- Write axis parameters through communication

Graphic Block



The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_WriteParameter_CO					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANopen axis to be operated	No	-	1 to 16	INT
S2	ParamIndex	Parameter number	No	-	*1	INT
S3	Value	Parameter value	No	-	-	DINT

NOTE

*1: For details, see the parameter number list below.

The following table lists soft elements in the instruction.

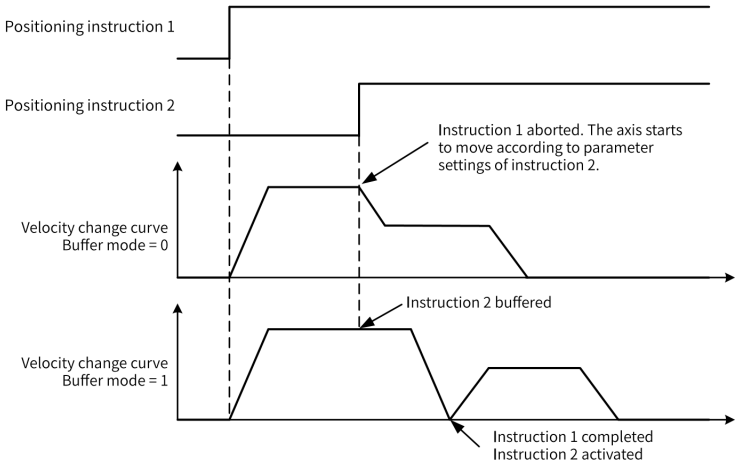
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The MC_WriteParameter_CO instruction sets parameters for a CANopen bus axis. The following table lists the parameter numbers.

Parameter Number	Name	Data Type	Read/ Write	Description
K1000	Buffer mode	UINT32	Read/write	Positioning buffer mode 0 (default): Trigger immediately without buffering 1: Wait for the current positioning to complete For details, see the timing diagram in buffer mode.
K1001	DI state	UINT32	Read	DI state [31:16]: Customized by the manufacturer [15:3]: Reserved [1]: Positive limit 0: Inactive; 1: Active [0]: Negative limit 0: Inactive; 1: Active
K1010	Axis state	INT32	Read	Current state of the axis -1: Not configured 0: Disabled 1: StandStill 2: Stopping 3: Homing 4: ContinuousMotion 5: DiscreteMotion 15: ErrorStop

Timing Diagram

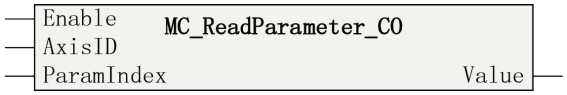


5.14.14 MC_ReadParameter_CO

Introduction

MC_ReadParameter_CO -- Read axis parameters through communication

Graphic Block



The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	MC_ReadParameter_CO					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANopen axis to be operated	No	-	1 to 16	INT
S2	ParamIndex	Parameter number	No	-	*1	INT
D1	Value	Parameter value	Yes	-	-	DINT

NOTE

*1: For details, see the parameter number list below.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D1	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The MC_ReadParameter_CO instruction reads parameters of a CANopen bus axis. The following table lists the parameter numbers.

Parameter No.	Name	Data Type	Read/Write	Description
K1000	Buffer mode	UINT32	Read/write	Positioning buffer mode 0 (default): Trigger immediately without buffering 1: Wait for the current positioning to complete For details, see the timing diagram in buffer mode.

Parameter No.	Name	Data Type	Read/Write	Description
K1001	DI state	UINT32	Read	DI state [31:16]: Customized by the manufacturer [15:3]: Reserved [1]: Positive limit 0: Inactive; 1: Active [0]: Negative limit 0: Inactive; 1: Active
K1010	Axis state	INT32	Read	Current state of the axis -1: Not configured 0: Disabled 1: StandStill 2: Stopping 3: Homing 4: ContinuousMotion 5: DiscreteMotion 15: ErrorStop

5.14.15 Error Codes

When an error occurs during use of CANopen axis control instructions, refer to the following error codes.

The following table lists the error codes of CANopen axis control instructions.

Code	Description
0	No error occurs.
1	The axis ID is incorrect. • The axis ID is out of range (1 to 16). • The axis ID does not exist in the CANopen configuration or a PDO configuration error occurs.
2	The parameters of the instruction are incorrect. • The acceleration/deceleration in the MC_MoveAbsolute_CO, MC_MoveRelative_CO, MC_MoveVelocity_CO, or MC_Jog_CO instruction is less than or equal to 0.

Code	Description
	• The velocity in the MC_MoveAbsolute_CO or MC_MoveRelative_CO instruction is less than or equal to 0.
3	The value of the instruction parameter (position or home offset) is out of range. ※ 1
4	The value of instruction parameter (speed) is out of range. ※ 1
5	The value of instruction parameter (acceleration rate) is out of range. ※ 1
6	The value of instruction parameter (deceleration rate) is out of range. ※ 1
8	Execution of the current instruction is stopped due to abortion by another instruction, Enable loss, or disconnection.
9	Execution of the current instruction is stopped due to forward overtravel. ※ 2
10	Execution of the current instruction is stopped due to reverse overtravel. ※ 2
11	Homing failed.
16	The axis is not enabled and the current instruction cannot be executed.
17	Failed to execute the MC_Reset_CO instruction because the axis is not in ErrorStop state.
18	Failed to execute the current instruction because the axis is in Stopping state.
19	Failed to execute the current instruction because the axis is homing.
20	Failed to execute the current instruction because the axis is in ContinuousMotion state.
21	Failed to execute the current instruction because the axis is positioning.
31	Failed to execute the current instruction because the axis is in ErrorStop state.
250	Axis enable timeout.
251	Error in servo/motor drive. ※ 3
255	Disconnection of servo/motor drive. ※ 3

NOTE

- ※ 1. The value exceeds the 32-bit integer range after converting the pulse unit.
- ※ 2. The axis enters the ErrorStop state when overtravel occurs during motion. You need to reset the fault by executing MC_Reset_CO before triggering the axis to move in the reverse direction.
- ※ 3. This error code applies only to the MC_Power_CO instruction. If this fault occurs during the execution of other instructions, an instruction abortion error is reported; if another instruction is triggered after this fault occurs, an error indicating that the axis is disabled is reported.

5.15 HC Axis Control (Pulse Input)

5.15.1 List of Instructions

The following table lists the high-speed counter instructions.

Instruction category	Name	Function
Bus encoder axis (H5U)	ENC_Counter	Encoder enable
	ENC_Reset	Encoder reset
	ENC_Preset	Encoder preset
	ENC_TouchProbe	Encoder probe
	ENC_ArrayCompare	Encoder one-dimensional array comparison
	ENC_StepCompare	Encoder one-dimensional step comparison
	ENC_GroupArrayCompare	Encoder two-dimensional array comparison
	ENC_ReadStatus	Encoder state read
	ENC_DigitalOutput	Encoder DO control
	ENC_ResetCompare	Encoder comparison output reset
	HC_Counter	High-speed counter enable
	HC_Preset	High-speed counter preset
	HC_TouchProbe	High-speed counter probe
	HC_Compare	High-speed counter comparison
	HC_ArrayCompare	High-speed counter array comparison
	HC_StepCompare	High-speed counter step comparison
High-speed counter instruction (H5U)	HC_Counter	High-speed counter enable
	HC_Preset	High-speed counter preset
	HC_TouchProbe	High-speed counter probe
	HC_Compare	High-speed counter comparison

Instruction category	Name	Function
	HC_ArrayCompare	High-speed counter array comparison
	HC_StepCompare	High-speed counter step comparison

Instruction category	Name	Function
Bus encoder axis & High-speed counter (Easy)	ENC_Counter	Encoder enable
	ENC_Reset	Encoder reset
	ENC_Preset	Encoder preset
	ENC_TouchProbe	Encoder probe
	ENC_ArrayCompare	Encoder one-dimensional array comparison
	ENC_StepCompare	Encoder one-dimensional step comparison
	ENC_Compare	Single-point comparison output
	ENC_GroupArrayCompare	Encoder two-dimensional array comparison
	ENC_ReadStatus	Encoder state read
	ENC_DigitalOutput	Encoder DO control
	ENC_ResetCompare	Encoder comparison output reset
	ENC_SetUnit	Gear ratio setting
	ENC_SetLineRotationMode	Rotation mode setting
	HC_Counter	High-speed counter enable
	HC_Preset	High-speed counter preset
	HC_TouchProbe	High-speed counter probe
	HC_Compare	High-speed counter comparison
	HC_ArrayCompare	High-speed counter array comparison
	HC_StepCompare	High-speed counter step comparison

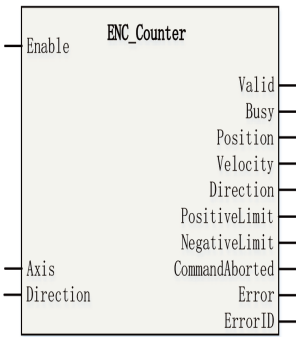
5.15.2 ENC_Counter

Introduction

The ENC_Counter instruction enables counting of the encoder axis.

ENC_Counter -- Encoder enable

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_Counter	Encoder enable		ENC_Counter(Enable := ???, Axis := ???, Invert := , Valid => , Busy => , Position => , Velocity => , ActDirection => , PositiveLimit => , NegativeLimit => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

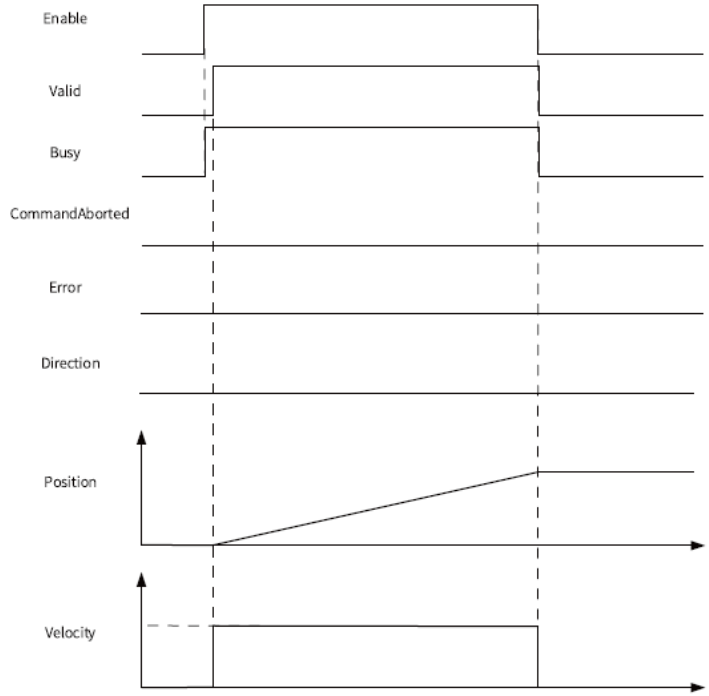
16-bit Instruction	-					
32-bit Instruction	ENC_Counter: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Encoder axis	No	-	-	_sENC_EXT_AXIS
S2	Direction(Invert)	Counting direction reversal (local encoder axis) 0: Forward 1: Reverse	Yes	0	0 to 1	INT

16-bit Instruction	-					
32-bit Instruction	ENC_Counter: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
D1	Valid	Active state	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	Position	Current position	Yes	0	Positive number 0 Negative number	REAL
D4	Velocity	Current velocity	Yes	0	Positive number 0 Negative number	REAL
D5	Direction	Counting direction	Yes	0	ON OFF	BOOL
D6	PositiveLimit	Positive limit in linear mode	Yes	OFF	ON OFF	BOOL
D7	NegativeLimit	Negative limit in linear mode	Yes	OFF	ON OFF	BOOL
D8	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D9	Error	Error	Yes	OFF	ON OFF	BOOL
D10	ErrorID	Error code	Yes	0	*1	INT

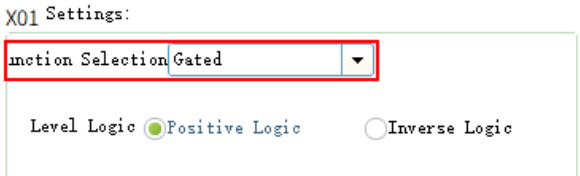
> **NOTE**
>
> *1: For details, see 5.15.22 Error Codes.

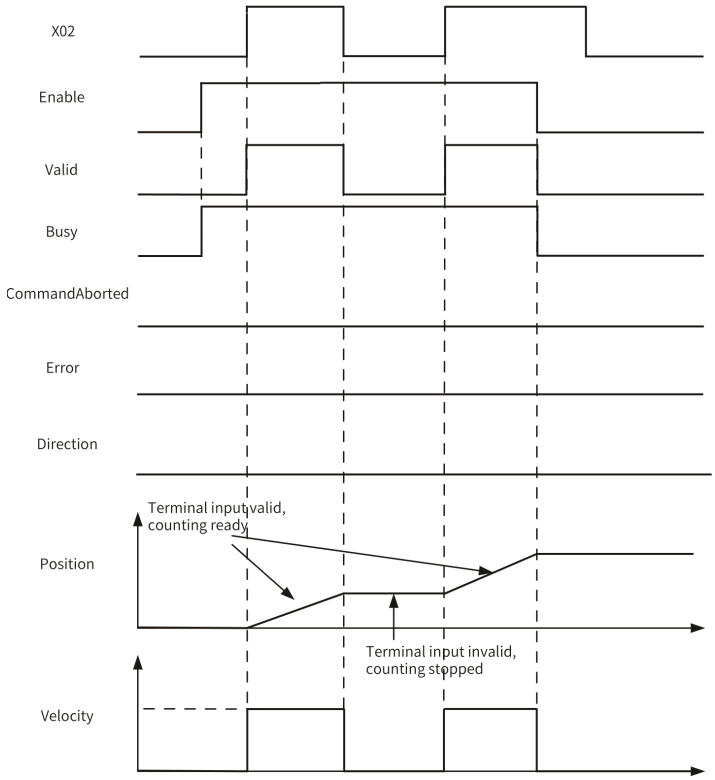
Function Description (Bus Encoder Axis)

- If the gating function of the DI terminal is not used, when the instruction input Enable is ON, the Busy signal and Valid signal are ON and the encoder axis starts counting; when the instruction input Enable is OFF, the Busy signal and Valid signal are OFF and the encoder axis stops counting. For example, when the GR10-2HCE module inputs 10 kHz pulses at constant speed:



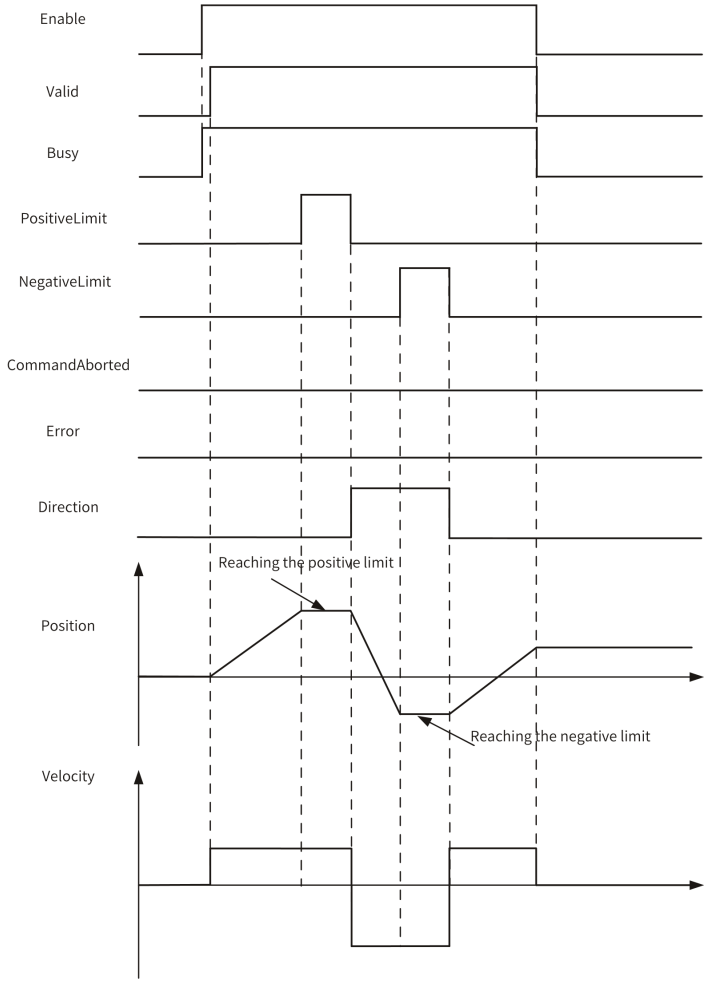
- If the gating function of the DI terminal is used (for example, X02 is assigned with the gating function), when the instruction input Enable is ON and X02 is OFF, the Busy signal is ON, the Valid signal is OFF, and the bus encoder axis suspends counting; when the instruction input Enable is ON and X02 is ON, the Busy signal and Valid signal are ON and the bus encoder axis starts counting; when the instruction input Enable is OFF, the encoder axis stops counting.





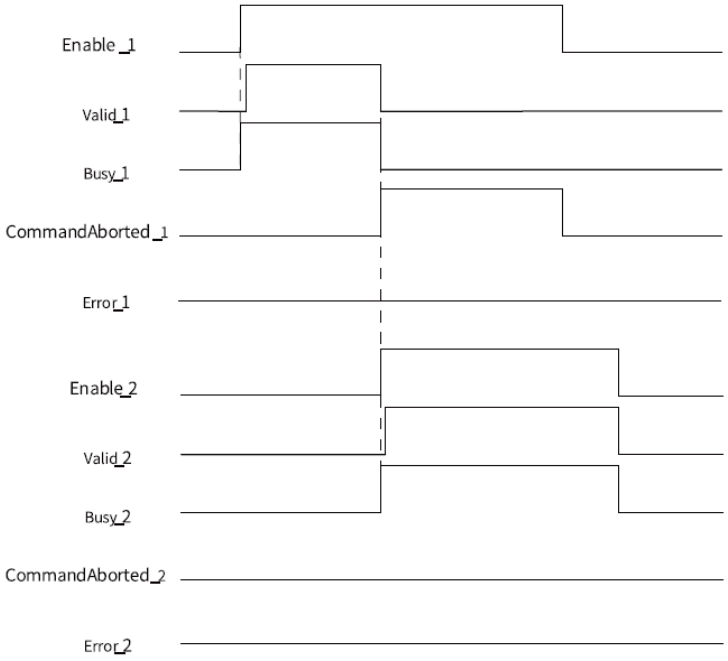
The timing diagram is as follows:

- In linear mode, if software limiting is enabled, when the count value reaches the limit, the counter stops counting and the limit signal output is active, when the pulse input is reversed, the limit signal is reset and the counter counts backwards.



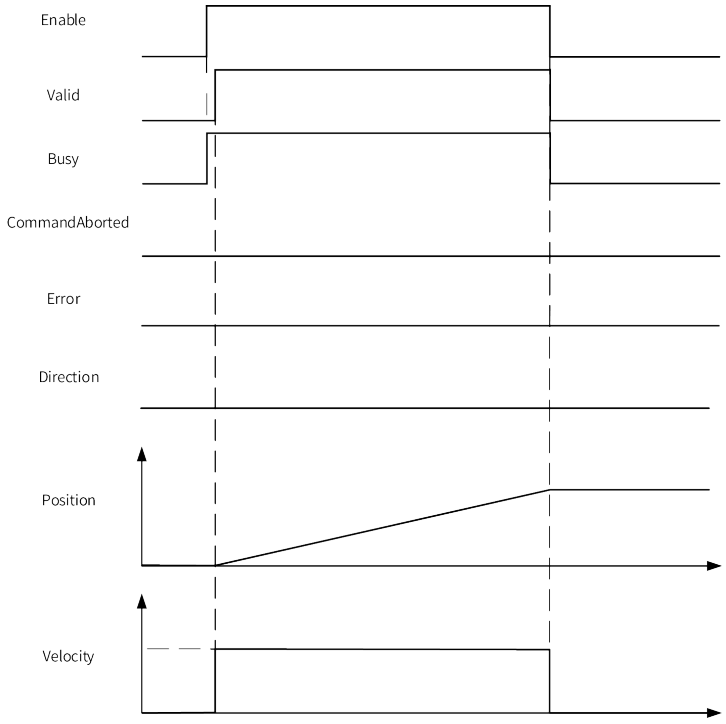
Multi-execution

When multiple ENC_Counter instructions are called to control the same axis, the instruction triggered first will be aborted by the instruction triggered later.

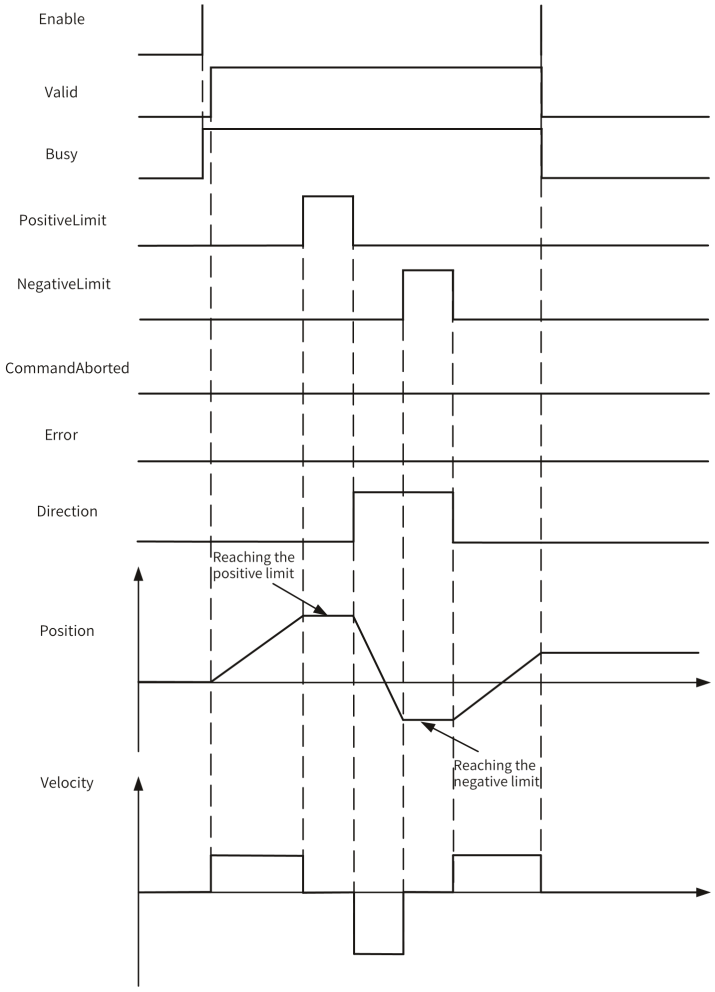


Function Description (Local Encoder Axis)

When the instruction input Enable is ON, the Busy signal and Valid signal are ON and the encoder axis starts counting; when the instruction input Enable is OFF, the Busy signal and Valid signal are OFF and the encoder axis stops counting.

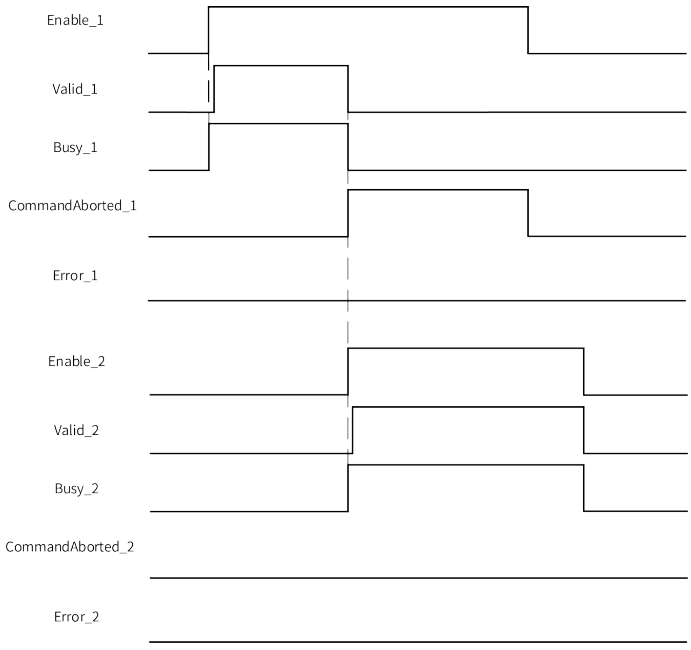


In linear mode, if software limiting is enabled, when the count value reaches the limit, the counter stops counting and the limit signal output is active; when the pulse input is reversed, the limit signal is reset and the counter counts backwards.



Multi-execution

When multiple ENC_Counter instructions are called to control the same axis, the instruction triggered first will be aborted by the instruction triggered later.



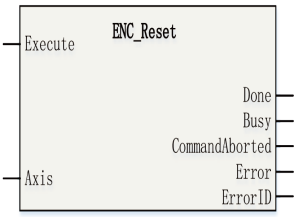
5.15.3 ENC_Reset

Introduction

The ENC_Reset instruction resets faults of a bus encoder axis.

ENC_Reset -- Reset encoder

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_Reset	Reset encoder		ENC_Reset(Execute := ???, Axis := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	ENC_Reset: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Encoder axis	No	-	-	_sENC_EXT_AXIS
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see 5.15.22 Error Codes.

Function Description

When the bus encoder axis fails and enters the ErrorStop state, the ENC_Reset instruction can be used to reset the fault of the axis.

The local encoder axis does not support this instruction.

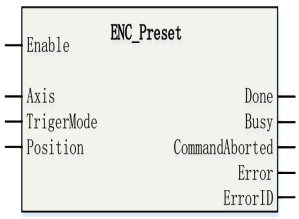
Timing Diagram

None

5.15.4 ENC_Preset

ENC_Preset: Encoder preset

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_Preset	Encoder preset		ENC_Preset(Enable := ???, Axis := ???, TrigerMode := , Position := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	ENC_Preset: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Encoder axis	No	-	-	_sENC_EXT_AXIS
S2	TrigerMode	0: Triggered on the rising edge of the instruction 1: Triggered on the rising edge of the external input X 2: Triggered on the falling edge of the external input X (local encoder axis) 3: Triggered on the rising or falling edge of the external input X (local encoder axis)	Yes	0	0 to 4	INT

		4: Triggered by the Z signal (bus encoder axis)				
S3	Position	Preset position	Yes	0	Positive number 0 Negative number	REAL
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

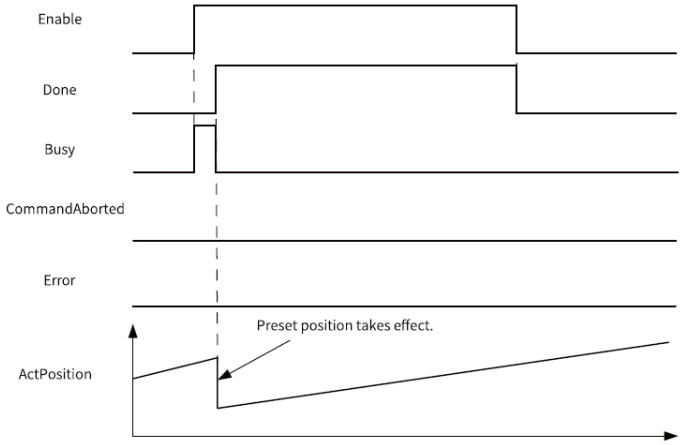
NOTE

*1: For details, see Instruction Error Codes.

Function Description (Bus Encoder Axis)

This instruction can be used to set the current position of the encoder axis as the value of the input parameter "Position".

- When "TrigerMode" is set to 0, the encoder position is set when the Enable input is active high.

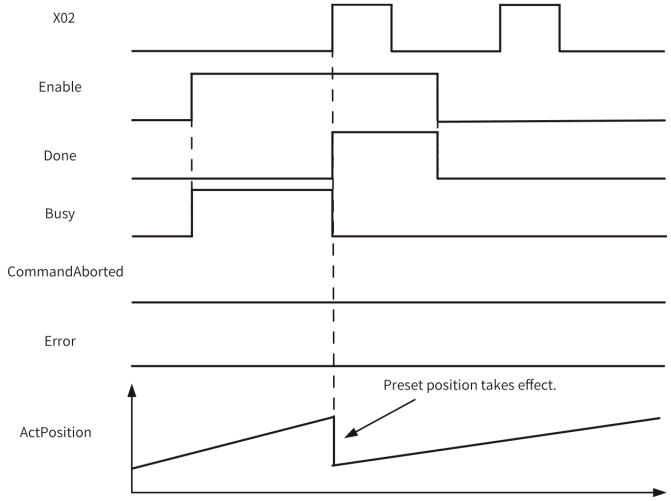


- When "TrigerMode" is set to 1, the encoder position is set on the rising edge of the DI terminal. To select this mode, you need to assign the DI terminal with the preset function. If multiple DI terminals are

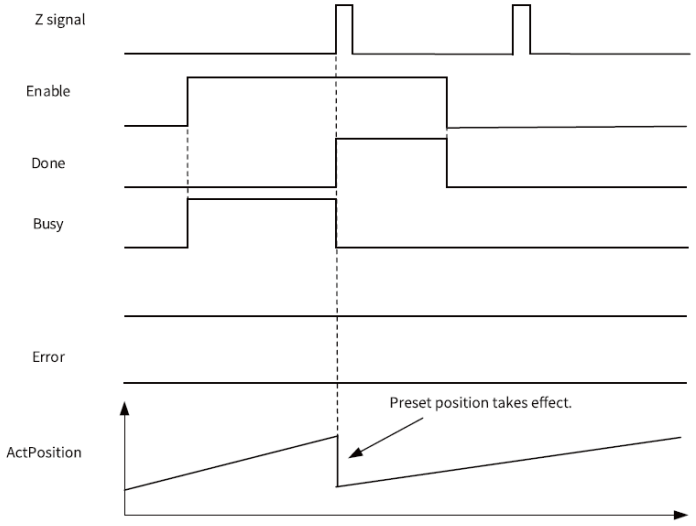
assigned with the preset function at the same time, the preset function can be implemented as long as one of the inputs is active.



The sequence diagram is as follows:

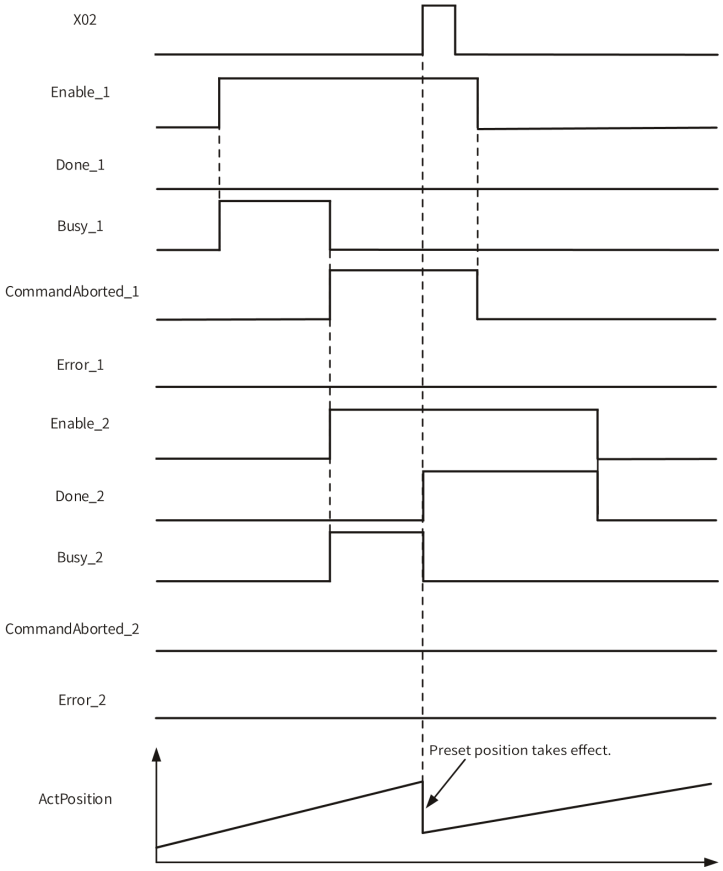


- When "TrigerMode" is set to 4, the preset function is completed when the rising edge of the Z signal is detected.



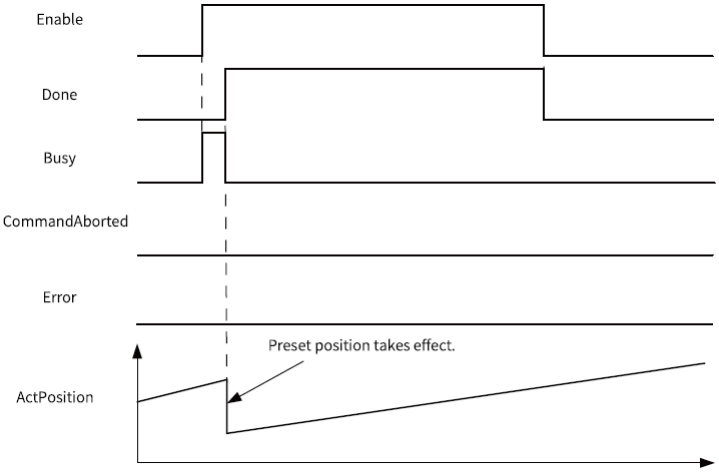
Multi-execution

When multiple preset instructions are called to set the position of the same axis, the instruction triggered first will be aborted by the instruction triggered later.



Function Description (Local Encoder Axis)

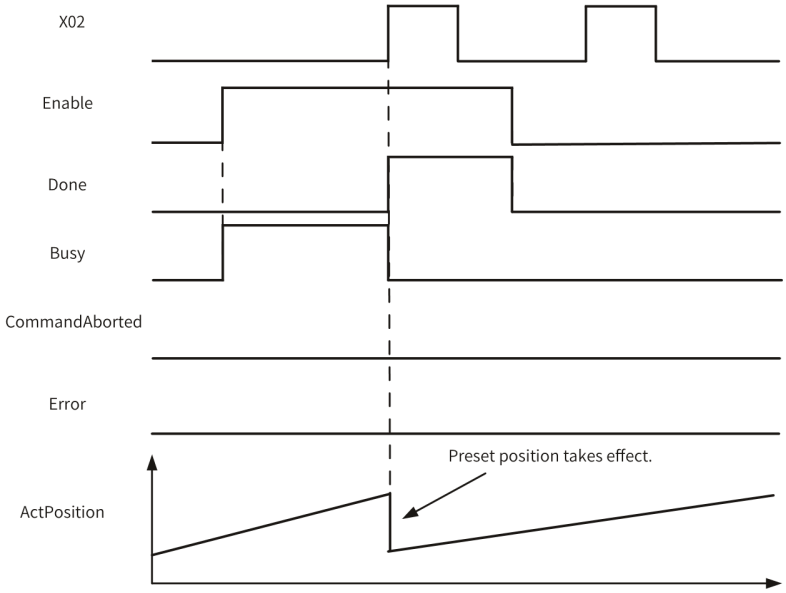
When “TrigerMode” is set to 0, the encoder position is set on the rising edge of the Enable input.



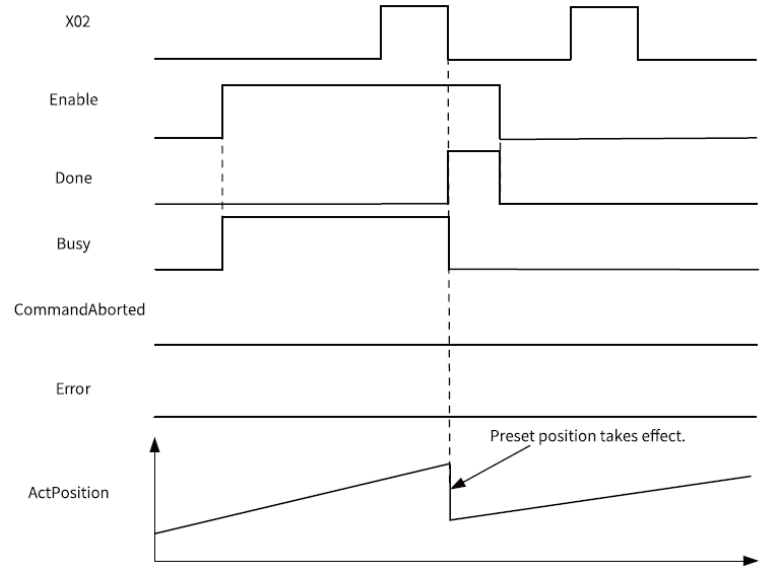
When “TrigerMode” is set to 1, the encoder position is set on the rising edge of the DI terminal. To select this mode, you need to assign the DI terminal with the preset function.



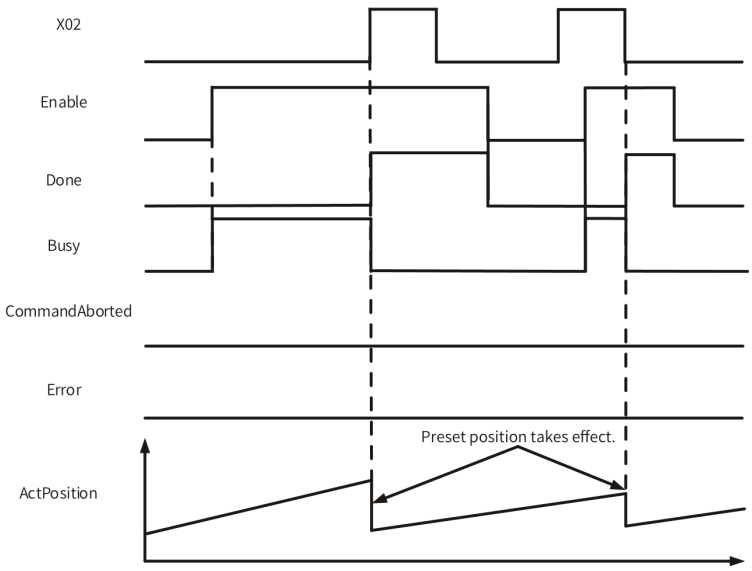
After the instruction is triggered, the position is preset on the rising edge of X02.



When "TrigerMode" is set to 2, the encoder position is set on the falling edge of the DI terminal.



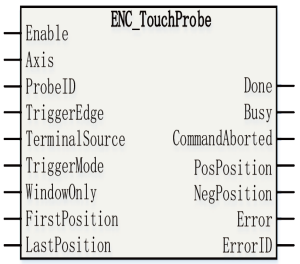
When "TrigerMode" is set to 3, the encoder position is set on the rising or falling edge of the DI terminal. As long as the instruction is active, the position is set on whichever edge that arrives first.



5.15.5 ENC_TouchProbe

ENC_TouchProbe -- Encoder probe

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_TouchProbe	Encoder probe		ENC_TouchProbe(Enable := ???, Axis := ???, ProbeID := ???, TriggerEdge := ???, TerminalSource := , TriggerMode := , WindowOnly := , FirstPosition := , LastPosition := , Done => , Busy => , CommandAborted => , PosPosition => , NegPosition => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	ENC_TouchProbe: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name	No	-		_sENC_EXT_AXIS
S2	ProbeID	Probe ID 0: Probe 1 1: Probe 2	No	-	0 to 1	INT
S3	TriggerEdge	Trigger edge 0: Only rising edge 1: Only falling edge 2: Both rising edge and falling edge	No	-	0 to 2	INT

S4	Terminal Source	Probe signal source (applicable to only the bus encoder axis) 0: DI terminal 1: Encoder Z signal	Yes	0	0 to 1	INT
S5	TriggerMode	Trigger mode 0: Single trigger 1: Continuous trigger	Yes	0	0 to 1	INT
S6	Window Only	Probe window enable 0: Disabled. Probe signals are detected at all positions. 1: Enabled. Probe signals are detected only when the current position is between FirstPosition (included) and LastPosition.	Yes	OFF	ON OFF	BOOL
S7	FirstPosition	Probe window start position	Yes	0	Positive number, negative number, or 0	REAL
S8	LastPosition	Probe window end position	Yes	0	Not equal to Position	REAL
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion	Yes	OFF	ON OFF	BOOL
D4	PosPosition	Position latched on the rising edge	Yes	0	Positive number, negative number, or 0	REAL
D5	NegPosition	Position latched on the falling edge	Yes	0	Positive number, negative number, or 0	REAL

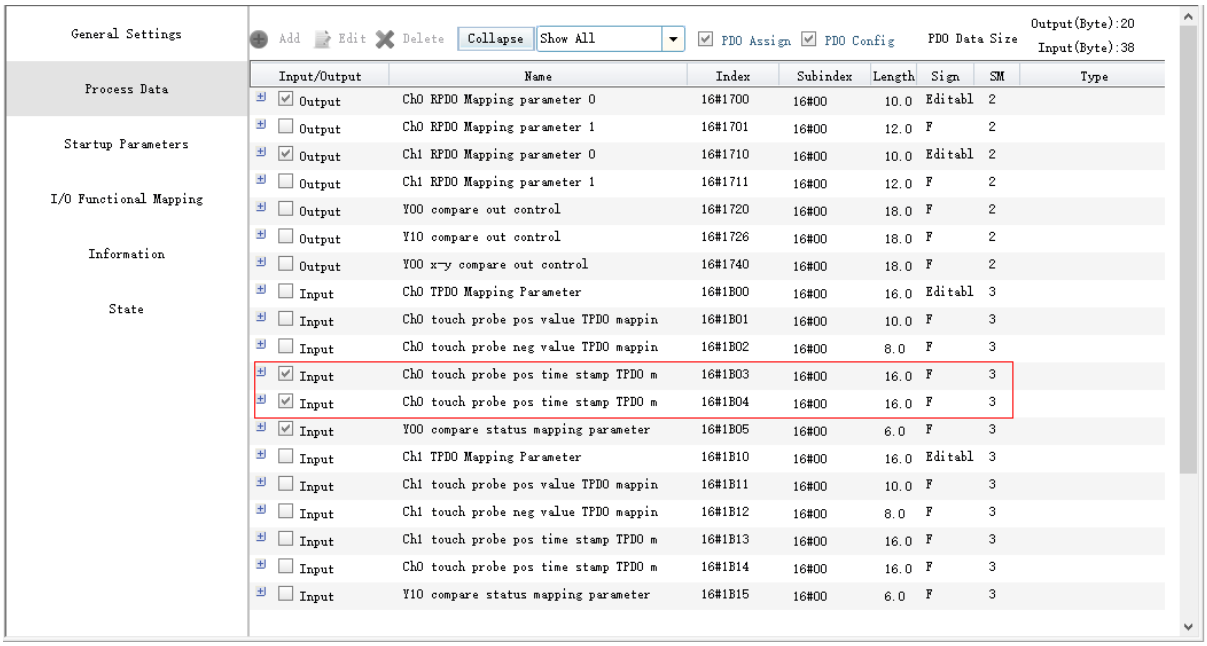
D6	Error	Error flag	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT

NOTE

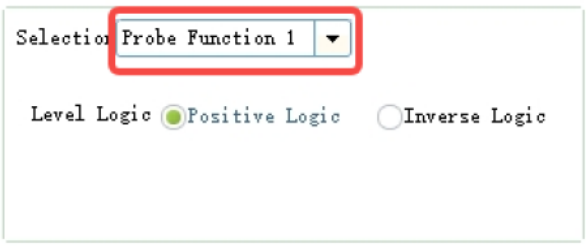
*1: For details, see 5.15.22 Error Codes.

Function Description (Bus Encoder Axis)

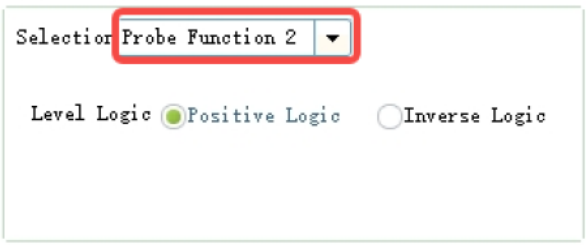
When the bus encoder axis is associated with CH0 of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, the terminals X00 and X01 of the GR10-2HCE module can be used as probe terminals.



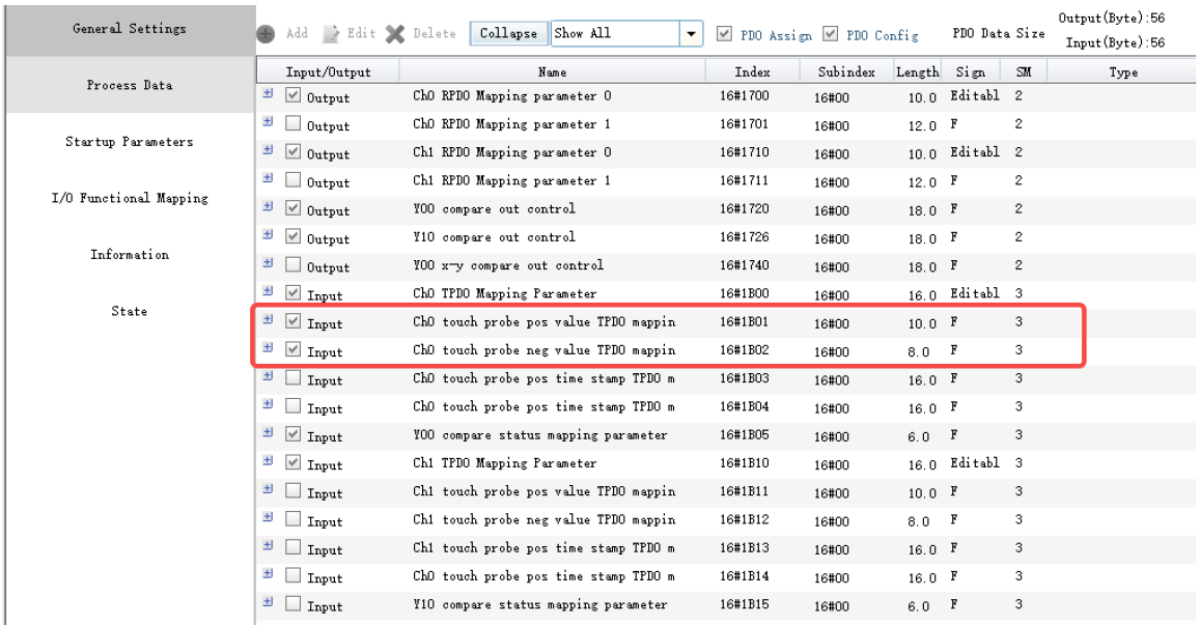
X00 Settings:



X01 Settings:

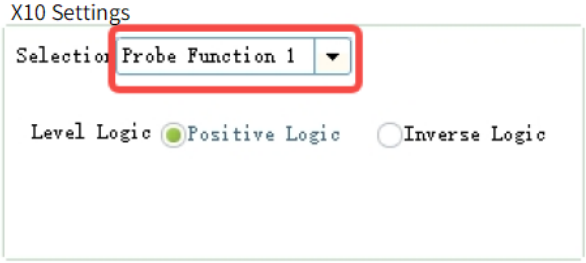


When the bus encoder axis is associated with CH1 of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, the terminals X10 and X11 of the GR10-2HCE module can be used as probe terminals.

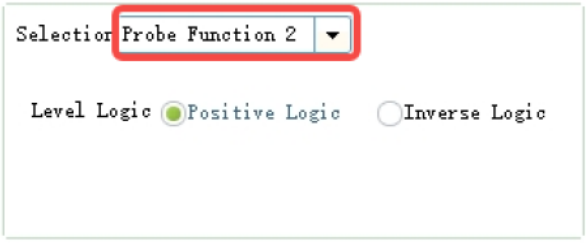


Input/Output	Name	Index	Subindex	Length	Sign	SM	Type
Output	Ch0 RPDO Mapping parameter 0	16#1700	16#00	10.0	Editabl	2	
Output	Ch0 RPDO Mapping parameter 1	16#1701	16#00	12.0	F	2	
Output	Ch1 RPDO Mapping parameter 0	16#1710	16#00	10.0	Editabl	2	
Output	Ch1 RPDO Mapping parameter 1	16#1711	16#00	12.0	F	2	
Output	Y00 compare out control	16#1720	16#00	18.0	F	2	
Output	Y10 compare out control	16#1726	16#00	18.0	F	2	
Output	Y00 x-y compare out control	16#1740	16#00	18.0	F	2	
Input	Ch0 TPDO Mapping Parameter	16#1B00	16#00	16.0	Editabl	3	
Input	Ch0 touch probe pos value TPDO mappin	16#1B01	16#00	10.0	F	3	
Input	Ch0 touch probe neg value TPDO mappin	16#1B02	16#00	8.0	F	3	
Input	Ch0 touch probe pos time stamp TPDO m	16#1B03	16#00	16.0	F	3	
Input	Ch0 touch probe pos time stamp TPDO m	16#1B04	16#00	16.0	F	3	
Input	Y00 compare status mapping parameter	16#1B05	16#00	6.0	F	3	
Input	Ch1 TPDO Mapping Parameter	16#1B10	16#00	16.0	Editabl	3	
Input	Ch1 touch probe pos value TPDO mappin	16#1B11	16#00	10.0	F	3	
Input	Ch1 touch probe neg value TPDO mappin	16#1B12	16#00	8.0	F	3	
Input	Ch1 touch probe pos time stamp TPDO m	16#1B13	16#00	16.0	F	3	
Input	Ch0 touch probe pos time stamp TPDO m	16#1B14	16#00	16.0	F	3	
Input	Y10 compare status mapping parameter	16#1B15	16#00	6.0	F	3	

X10 Settings



X11 Settings



On the rising edge, the instruction latches the input parameters on the left, such as ProbeID and TriggerEdge, and other state update parameters are invalid.

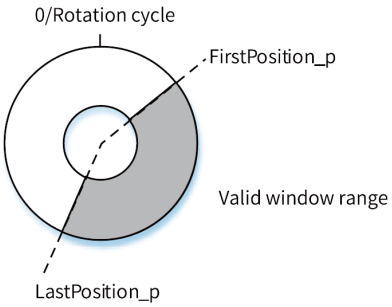
When Enable is ON, the function block latches the current position of the axis when the instruction detects that the input of the probe specified by ProbeID is active and meets the probe detection conditions.

- When WindowOnly is OFF, the window detection function is disabled. The axis position can be latched as long as the probe input signal is active.
- When WindowOnly is ON, the window detection function is enabled.

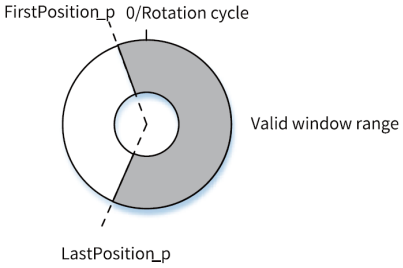
In linear mode, the instruction detects the probe signal only when the current position of the axis falls within the range specified by FirstPosition and LastPosition.

Valid window range in ring mode:

- When FirstPosition is less than LastPosition, the valid window range is as follows.



- When FirstPosition is greater than LastPosition, the valid window range is as follows.



- When FirstPosition is equal to LastPosition, the window is invalid and error 9107 is reported.

This instruction can detect the rising edge or falling edge of the probe signal separately or both the rising edge and the falling edge at the same time. When detecting only the rising edge (falling edge), the instruction writes the value detected on the rising edge (falling edge) into PosPosition (NegPosition). At this time, the Done signal is set to ON when a detection cycle is completed.

If the rising edge and falling edge are detected at the same time, after the Enable signal is active, the instruction immediately writes the position into PosPosition upon detecting the rising edge and writes the position into NegPosition upon detecting the falling edge. After that, the detection cycle is completed and the Done signal is output. There is no requirement on the input sequence of the rising edge and falling edge.

The input TerminalSource of this instruction can be used to set the terminal type to DI or the Z signal.

This instruction supports the single trigger and continuous trigger modes. If the single trigger mode is used, instruction execution ends when the Done signal output is active. If the continuous trigger mode is used, the Done output active signal is reset after one PLC scan cycle, and the instruction automatically starts to detect new probe input signals.

The following is an example:

In linear mode, the window range is 10 to 100, the EtherCAT cycle is set to 8 ms, and the velocity is 100. Then, the axis moves 0.8 per EtherCAT cycle. If the current position at the moment when an EtherCAT cycle starts is 9.9, the probe signal is not detected within this EtherCAT cycle. The current position changes to 10.7 upon start of the next EtherCAT cycle. Therefore, the probe signals between 10 and 10.7 are lost. If the current position at the moment when an EtherCAT cycle starts is 99.9, the probe signal is detected within this EtherCAT cycle. The current position changes to 100.7 upon start of the next EtherCAT cycle. Therefore, the probe signals between 100 and 100.7 are responded.

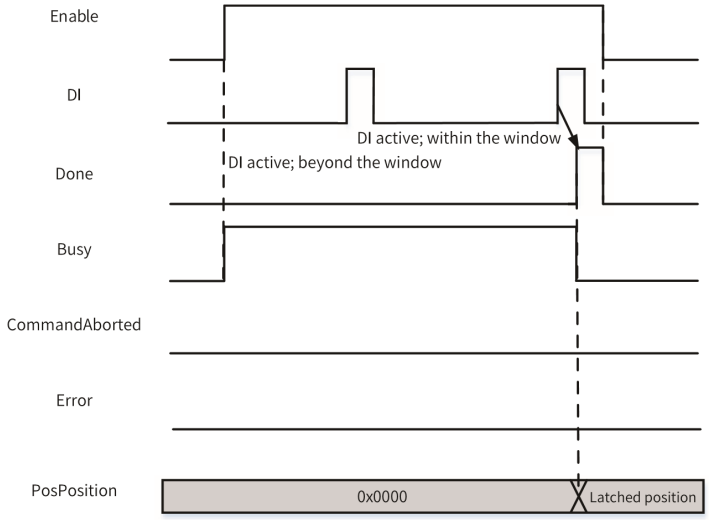
In continuous mode, if the input frequency of the probe signal is greater than the frequency of the PLC scan cycle, some probe signals are lost.

Abortion

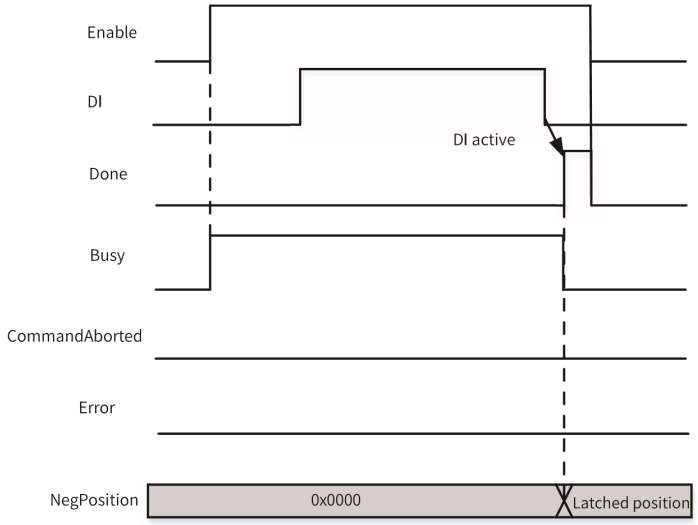
The ENC_TouchProbe instruction supports the detection probe 1 and probe 2. If two probe instructions are defined in the program and the probe IDs of the two instructions are different, the two probe instructions will work independently. If the probe IDs are the same, the probe instruction executed later will abort the previous probe instruction.

Timing Diagram

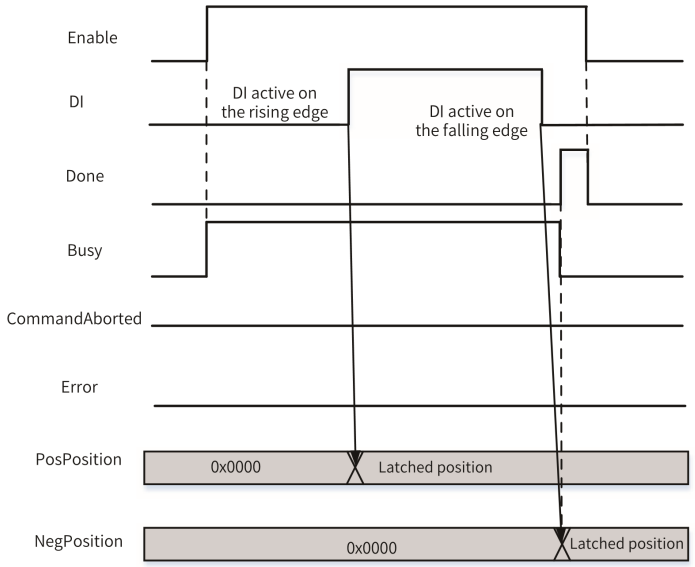
1. Probe 1, active on the rising edge, DI signal trigger source, single trigger mode, window function enabled



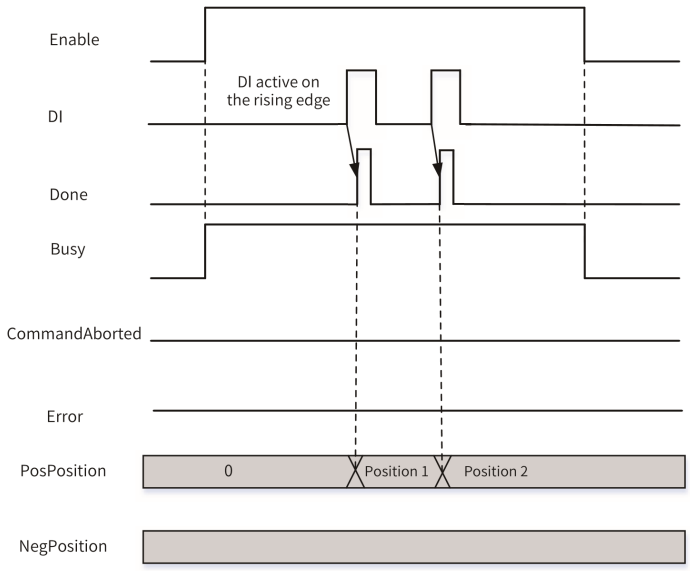
2. Probe 1, active on the falling edge, DI signal trigger source, single trigger mode, window function disabled



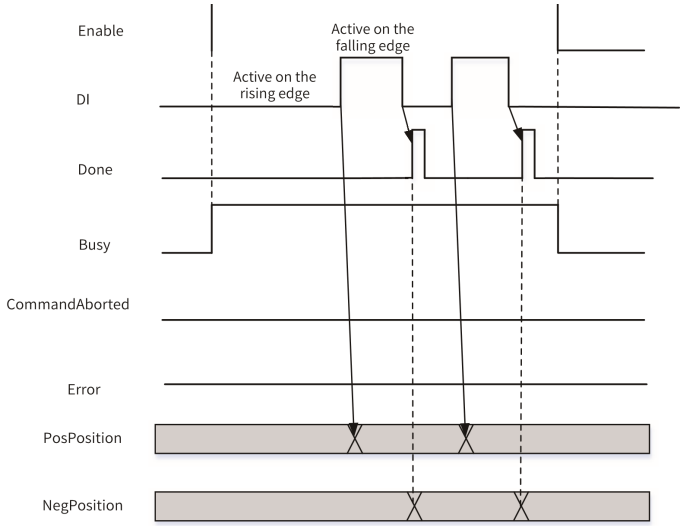
3. Probe 1, active on both the rising edge and falling edge, DI signal trigger source, single trigger mode, window function disabled



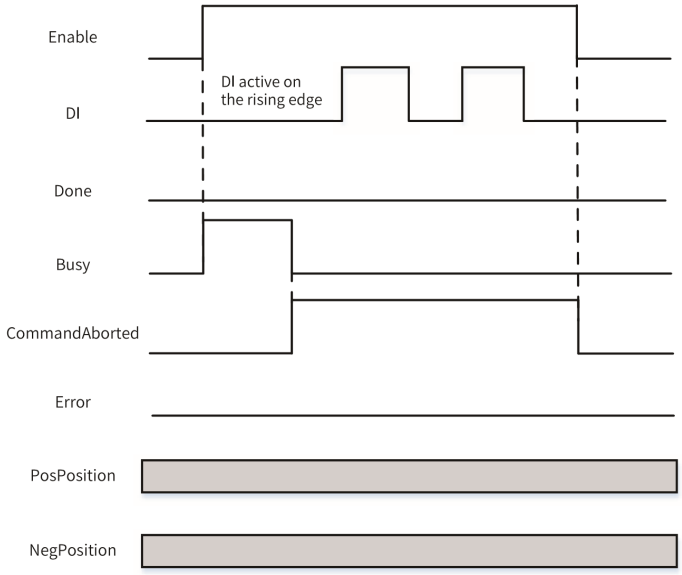
4. Probe 1, active on the rising edge, DI signal trigger source, continuous trigger mode, window function disabled



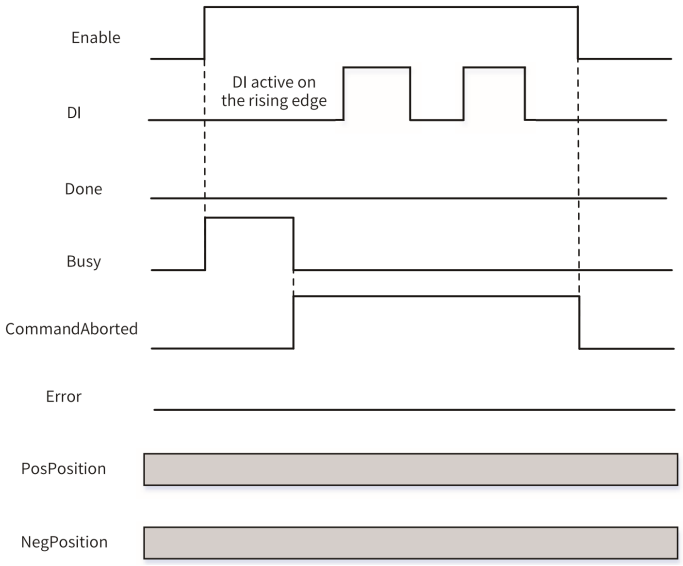
5. Probe 1, active on both the rising edge and falling edge, DI signal trigger source, continuous trigger mode, window function disabled (The Done signal is active for a cycle after the DI signal is active on both the rising and falling edges.)



6. Probe 1, aborted by another probe-related instruction, window function disabled



7. Probe 1, instruction error



Function Description (Local Encoder Axis)

When the bus encoder axis is associated with CH0 of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, the terminals X00 and X01 of the GR10-2HCE module can be used as probe terminals.

Select Probe 1 Enable and Probe 2 Enable on the configuration interface of the local encoder axis, and select the appropriate probe terminals as needed.



On the rising edge, the instruction latches the input parameters on the left, such as ProbeID and TriggerEdge, and other state update parameters are invalid.

When Enable is TRUE, the function block latches the current position of the axis when the instruction detects that the input of the probe specified by ProbeID is active and meets the probe detection conditions.

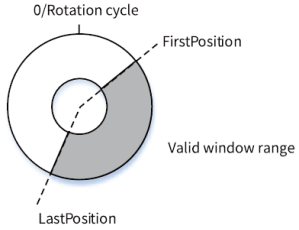
When WindowOnly is FALSE, the window detection function is disabled. The axis position can be latched as long as the probe input signal is active.

When WindowOnly is TRUE, the window detection function is enabled.

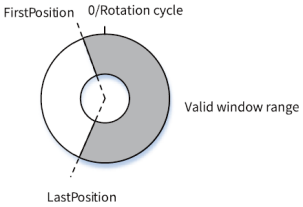
In linear mode, the instruction detects the probe signal only when the current position of the axis falls within the range specified by FirstPosition and LastPosition.

Valid window range in ring mode:

- When FirstPosition is less than LastPosition, the valid window range is as follows.



- When FirstPosition is greater than LastPosition, the valid window range is as follows.



- When FirstPosition is equal to LastPosition, the window is invalid and error 9107 is reported.

This instruction can detect the rising edge or falling edge of the probe signal separately or both the rising edge and the falling edge at the same time. When detecting only the rising edge (falling edge), the instruction writes the value detected on the rising edge (falling edge) into PosPosition (NegPosition). At this time, the Done signal is set to ON when a detection cycle is completed.

If the rising edge and falling edge are detected at the same time, after the Enable signal is active, the instruction immediately writes the position into PosPosition upon detecting the rising edge and writes the position into NegPosition upon detecting the falling edge. After that, the detection cycle is completed and the Done signal is output. There is no requirement on the input sequence of the rising edge and falling edge.

This instruction supports the single trigger and continuous trigger modes. If the single trigger mode is used, instruction execution ends when the Done signal output is active. If the continuous trigger mode is used, the Done output active signal is reset after one PLC scan cycle, and the instruction automatically starts to detect new probe input signals.

Note

When the window function is enabled, probe signal loss or detection out-of-range may occur near the window area. The following is an example:

In linear mode, the window range is 10 to 100, the main task cycle is set to 8 ms, and the velocity is 100 (Unit/s). Then, the axis moves 0.8 (Unit) per main task cycle. The Unit can be specified based on the axis configuration. If the current position at the moment when a main task cycle starts is 9.9, the probe signal is not detected within this main task cycle. The current position changes to 10.7 upon start of the next main task cycle. Therefore, the probe signals between 10 and 10.7 are lost. If the current position at the moment when a main task cycle starts is 99.9, the probe signal is detected within this main task cycle. The current position changes to 100.7 upon start of the next main task cycle. Therefore, the probe signals between 100 and 100.7 are responded.

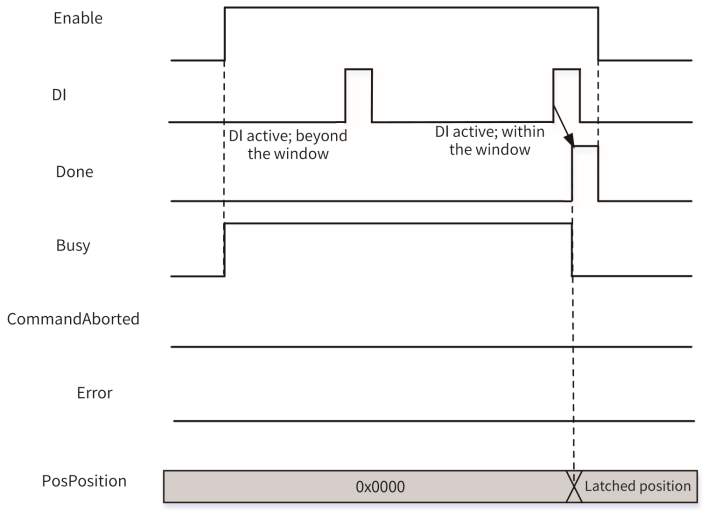
In continuous mode, if the input frequency of the probe signal is greater than the frequency of the PLC scan cycle, some probe signals are lost.

Abortion

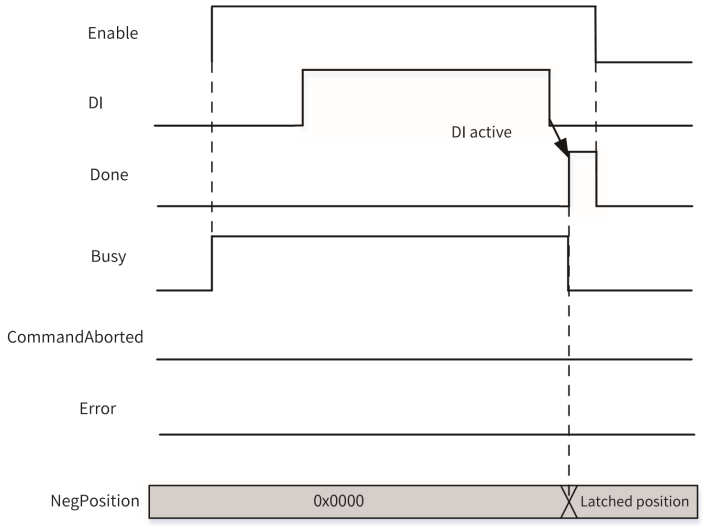
The ENC_TouchProbe instruction supports the detection probe 1 and probe 2. If two probe instructions are defined in the program and the probe IDs of the two instructions are different, the two probe instructions will work independently. If the probe IDs are the same, the probe instruction executed later will abort the previous probe instruction.

Timing Diagram

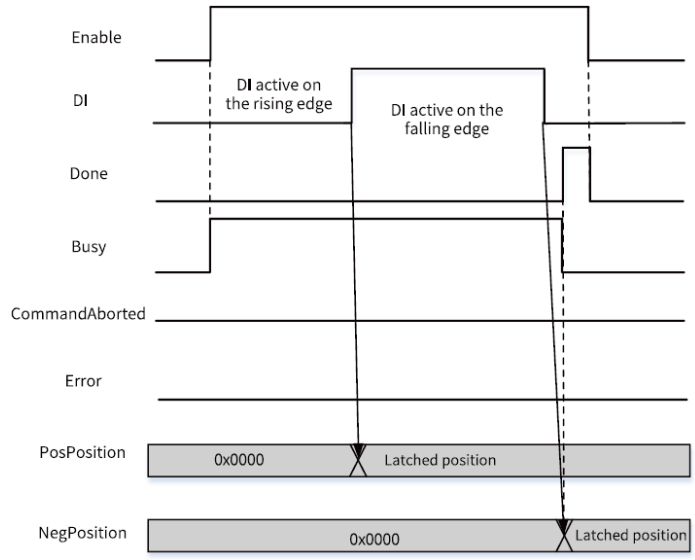
1. Probe 1, active on the rising edge, DI signal trigger source, single trigger mode, window function enabled



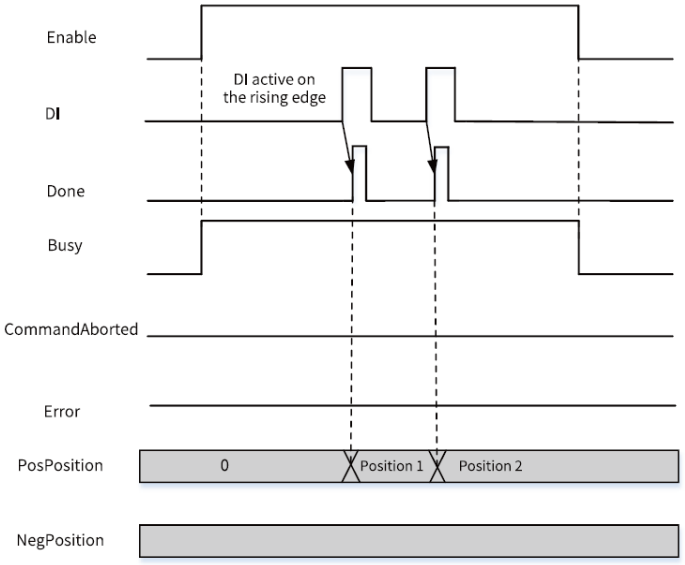
2. Probe 1, active on the falling edge, DI signal trigger source, single trigger mode, window function disabled



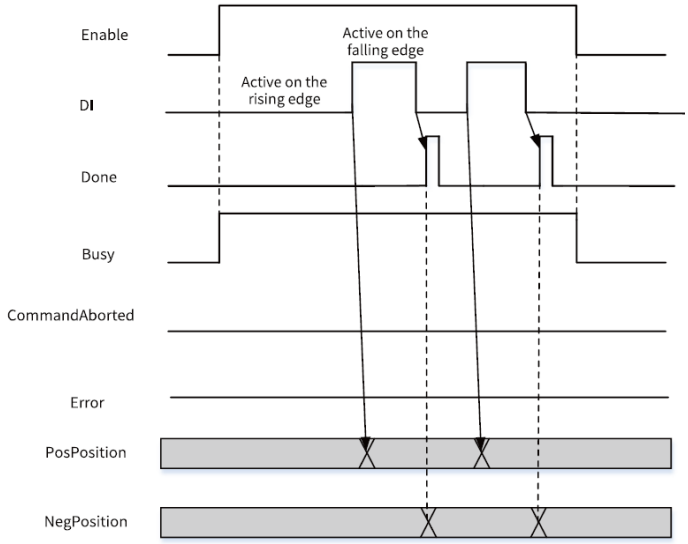
3. Probe 1, active on both the rising edge and falling edge, DI signal trigger source, single trigger mode, window function disabled



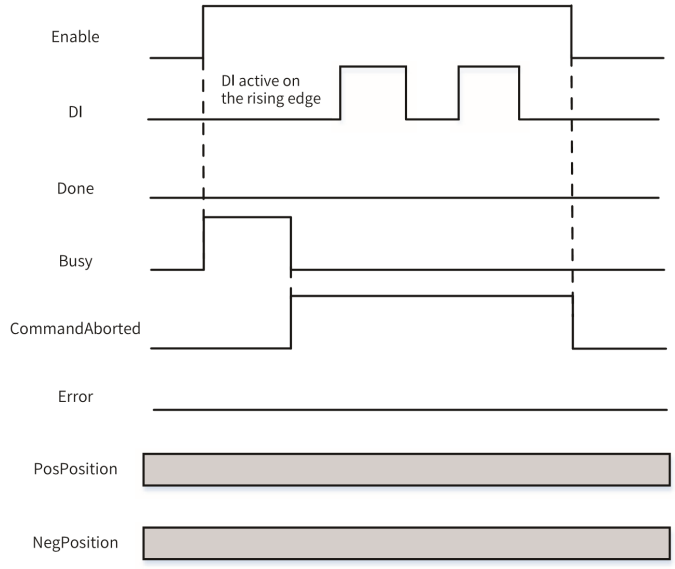
4. Probe 1, active on the rising edge, DI signal trigger source, continuous trigger mode, window function disabled



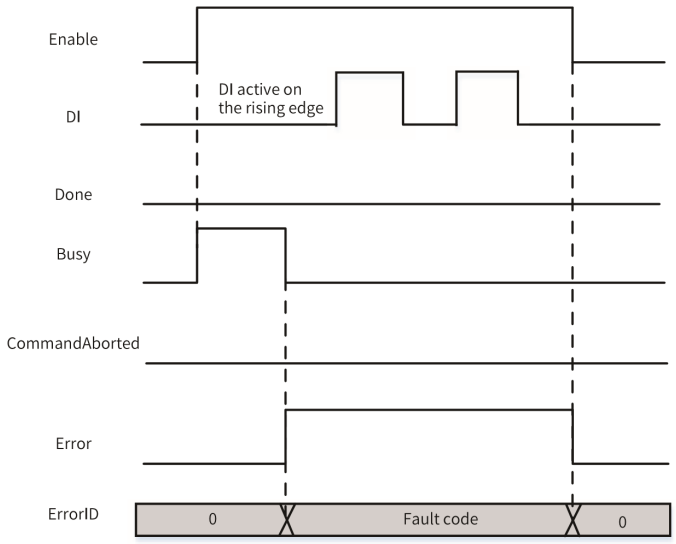
5. Probe 1, active on both the rising edge and falling edge, DI signal trigger source, continuous trigger mode, window function disabled (The Done signal is active for a cycle after the DI signal is active on both the rising and falling edges.)



6. Probe 1, aborted by another probe-related instruction, window function disabled



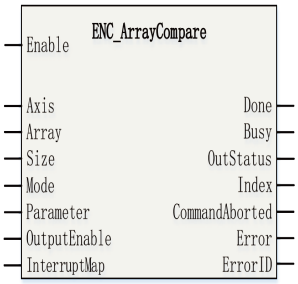
7. Probe 1, instruction error



5.15.6 ENC_ArrayCompare

ENC_ArrayCompare -- Encoder one-dimensional array comparison

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_ArrayCompare	One-dimensional array comparison		ENC_ArrayCompare(Enable := ???, Axis := ???, Array := ???, Size := ???, Mode := ???, Parameter := , OutputEnable := , InterruptMap := , Done => , Busy => , OutStatus => , Index => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	ENC_ArrayCompare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type

S1	Axis	Encoder axis	No	-	-	_sENC_EXT_AXIS
S2	Array	Comparand array	No	-		REAL[0-1000]
S3	Size	Number of comparands	No	-	1 to 1000	INT
S4	Mode	Comparison mode (applicable to only the bus encoder axis) 0: Reserved 1: Time control 2: Pulse control 3: Level control	Yes		0 to 3	INT
S5	Parameter	Control parameters (applicable to only the bus encoder axis) Time control: output active duration, in μs Pulse control: output active pulse count, in Unit Level control: initial level; 0 indicates low level and non-zero indicates high level.	Yes		Positive number 0 Negative number	REAL
S6	OutputEnable	Hardware output enable (applicable to only the local encoder axis) 0: Disabled 1: Enabled	Yes	1	0 to 1	INT
S7	InterruptMap	Interrupt ID (applicable to only the local encoder axis) 0: No interrupt is generated. 1: Associate with comparison interrupt 1. … 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3			Yes	OFF	ON	BOOL

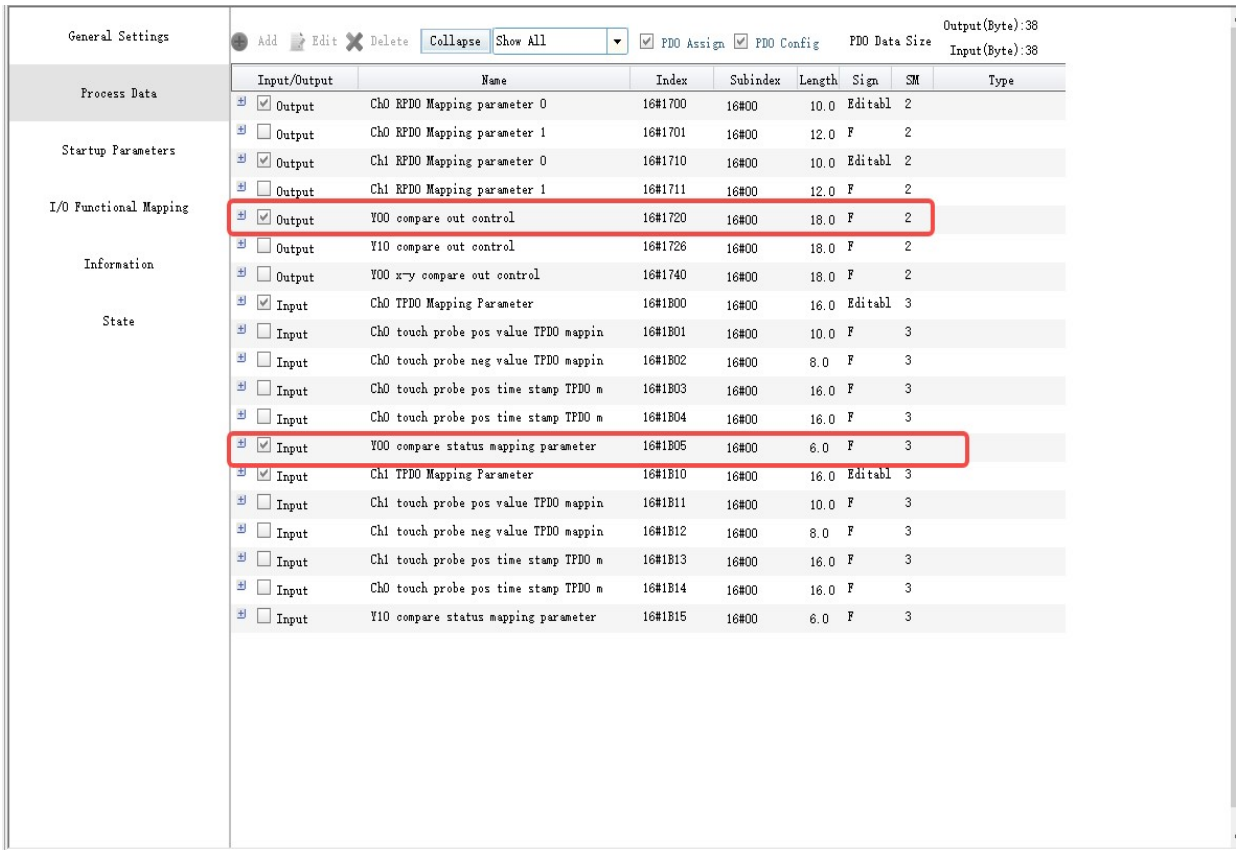
	OutStatus	Port output state (applicable to only the bus encoder axis)			OFF	
D4	Index	Index of the next comparand	Yes	0	0 to 999	INT
D5	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see 5.15.22 Error Codes.

Function Description (Bus Encoder Axis)

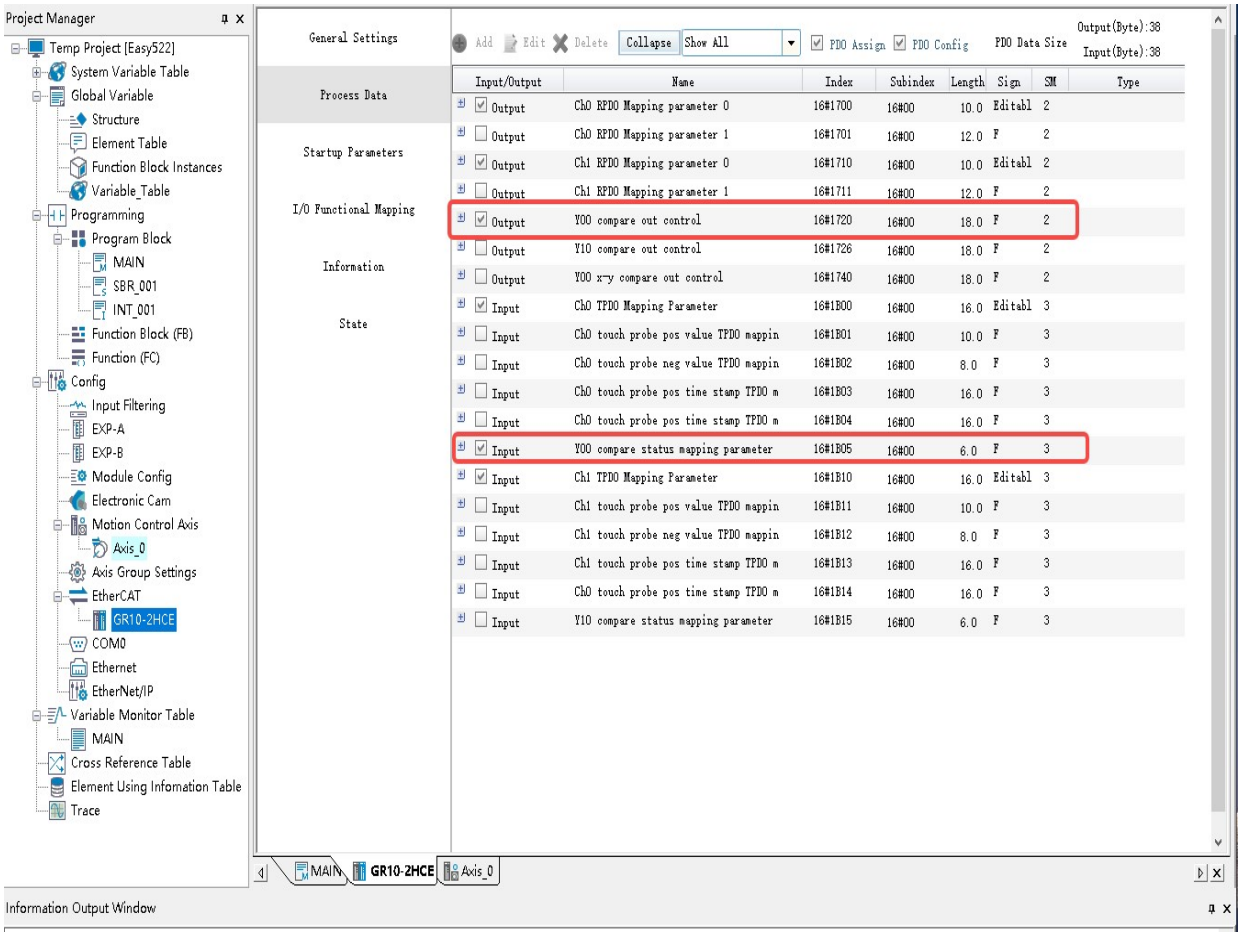
When the bus encoder axis is associated with CH0 of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, Y00 of the GR10-2HCE module can be used for comparison output.



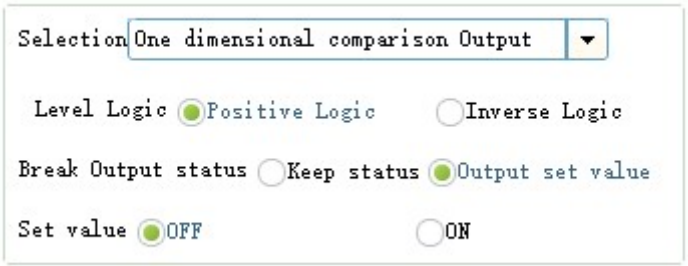
Y10 Settings:



When the bus encoder axis is associated with CH1 of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, Y10 of the GR10-2HCE module can be used for comparison output.



Y00 Settings:



Setting Comparison Points

Array in the instruction specifies the comparand array, and Size indicates the actual number of points to be compared. Ensure that the value of Size is less than or equal to the number of data entries specified by Array in the PLC program. If the value of Size is greater than the number of data entries specified by Array, the instruction will not report an error, but the array index out-of-bounds error and program execution error will occur inside the PLC.

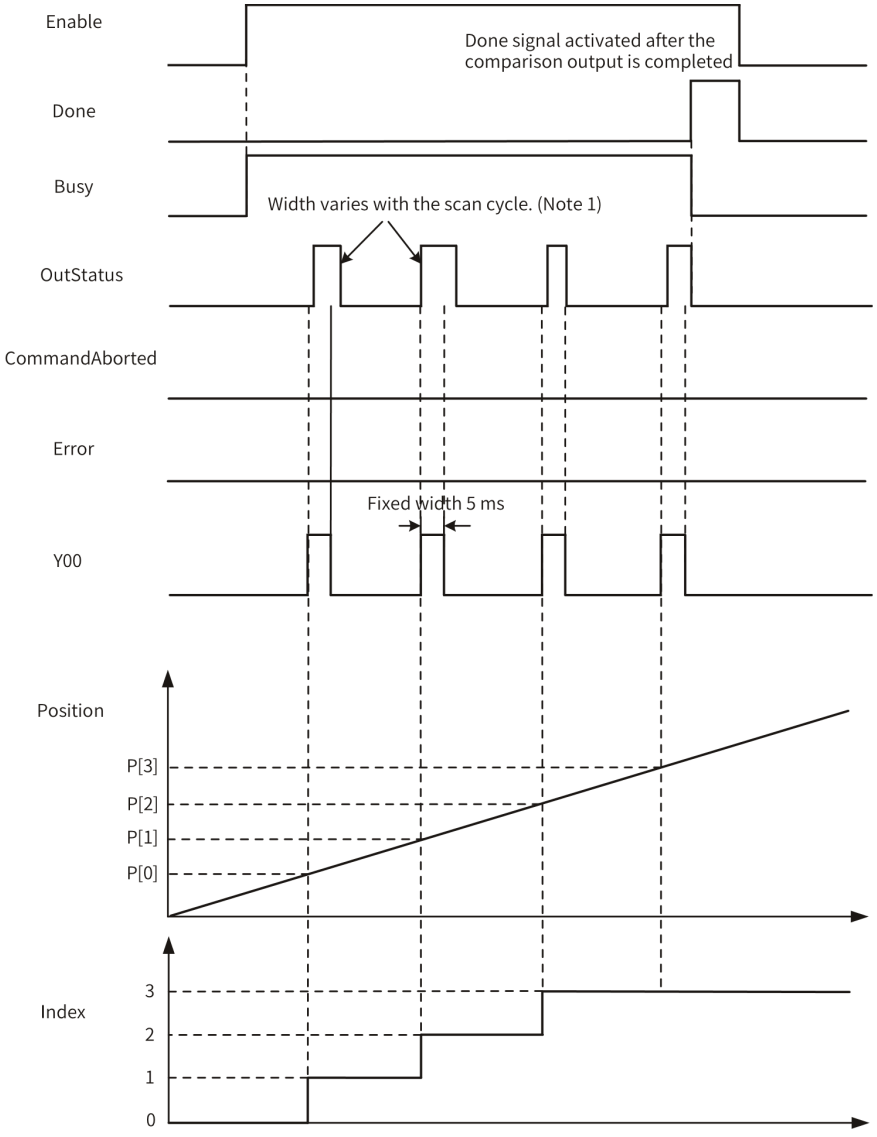
Setting the Comparison Output Mode

The basic principle of the comparison output is to set the DO terminal to high level or reverse the original level after the encoder runs to the specified position. When the output is set to high level, the time duration or the number of consecutive pulses during which the output remains high level can be specified.

1. Time mode

   When Mode is set to 1, the DO terminal outputs high level after the encoder axis reaches the comparison point. Parameter specifies the time duration during which the output remains high level. Size specifies the number of comparison points and Array specifies the comparison point array. Among the output parameters, OutStatus indicates the output state of the comparison output terminal, and Index indicates the index of the next array coordinate point to be compared.

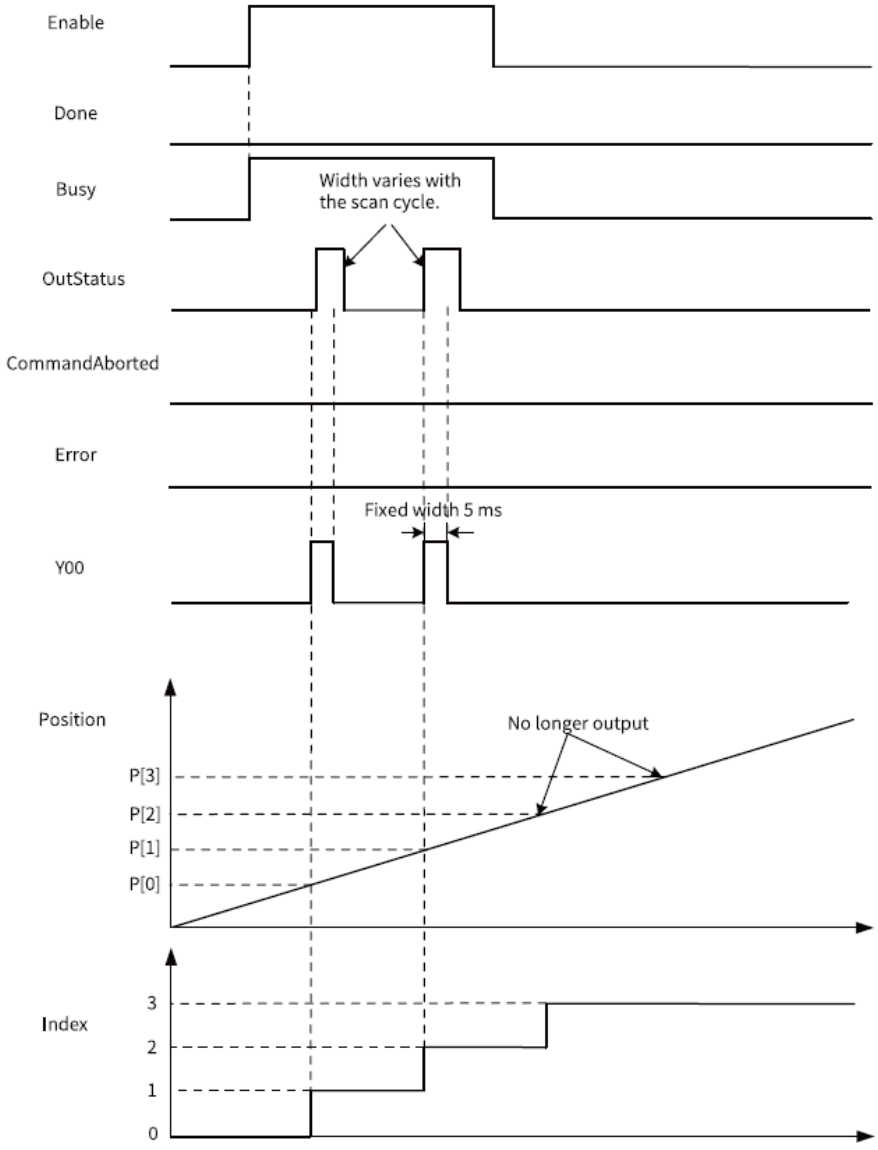
   Assume that the comparison point array is P[4], and the output time at each comparison point is 5 ms. The instruction starts to run after the Enable input becomes active. The timing diagram is as follows.



NOTE

If the PLC scan cycle is greater than the specified output time of point Y, for example, if the PLC scan cycle is 10 ms and the output time of point Y is 100 μs, the output change of OutStatus may not be detected in the PLC task, but point Y has normal output.

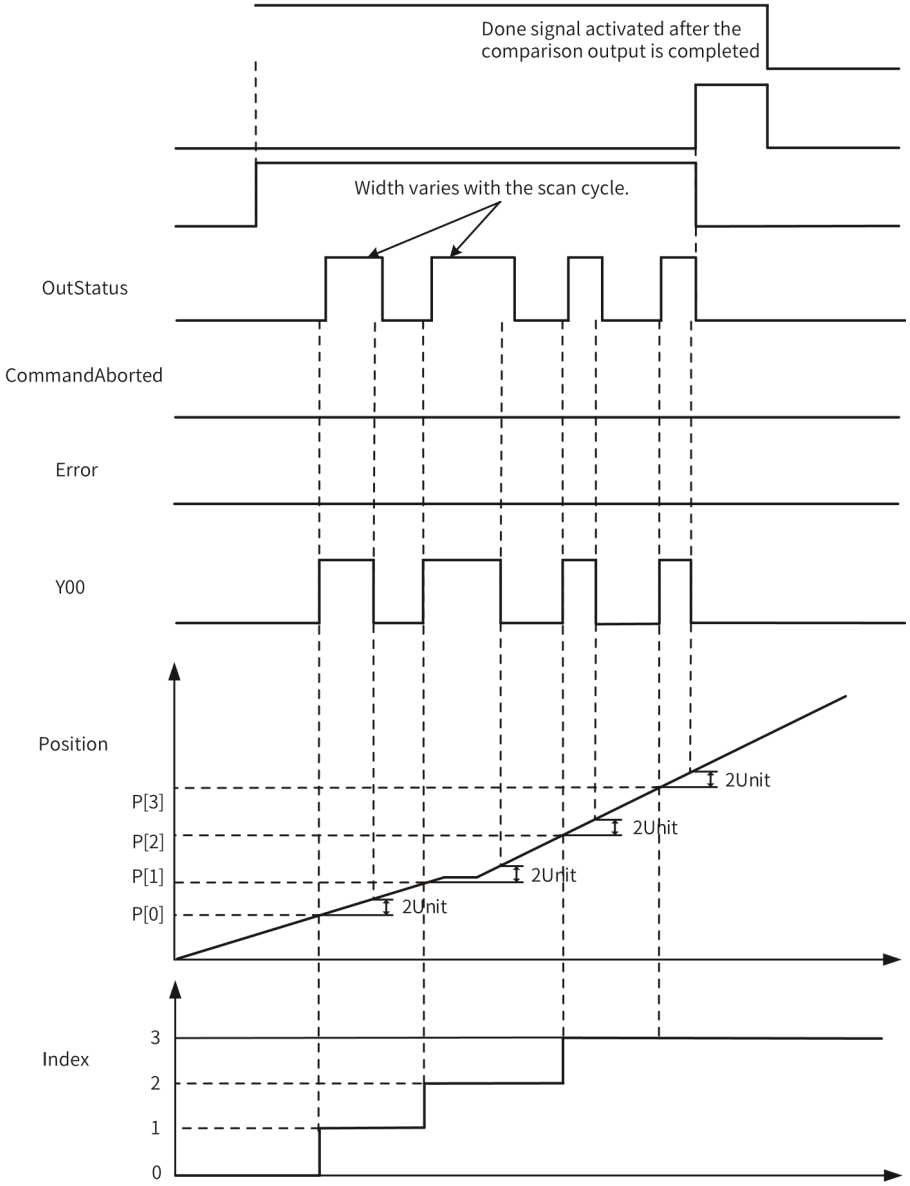
2. Before the comparison output is completed, if the Enable input is set to OFF, the subsequent comparison points are not compared any more.



3. Pulse mode

   When Mode is set to 2, the DO terminal outputs high level after the encoder axis reaches the comparison point. Parameter specifies the number of encoder pulses (unit: Unit) during which the output remains high level. Size specifies the number of comparison points and Array specifies the comparison point array. Among the output parameters, OutStatus indicates the output state of the comparison output terminal, and Index indicates the index of the next array coordinate point to be compared.

   Assume that the comparison point array is P[4], and the output remains high level at each comparison point for 2 Units. The instruction starts to run after the Enable input becomes active. The timing diagram is as follows.

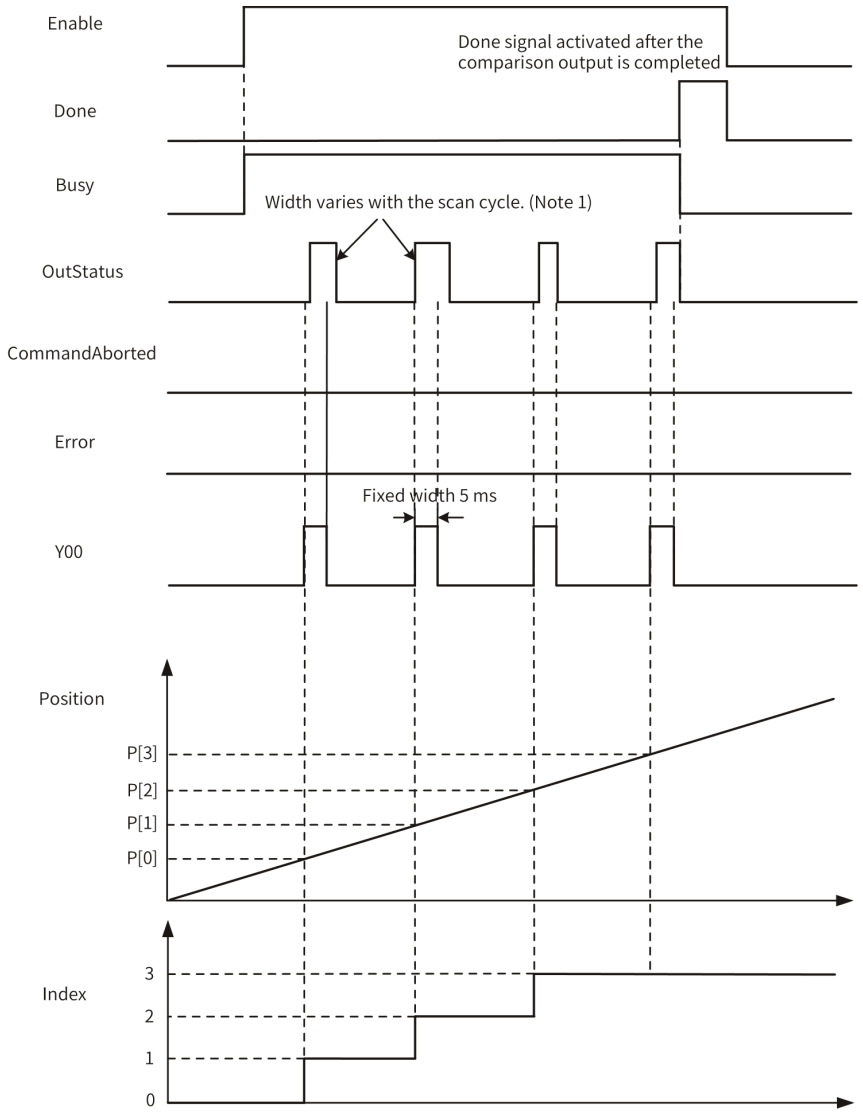


Before the comparison output is completed, if the Enable input is set to OFF, the subsequent comparison points are not compared any more, as in the case of the time mode.

4. Level mode

When Mode is set to 3, the level mode is used. Parameter specifies the initial level of the output terminal when the Enable signal of the instruction is active. If it is 0, the initial level of the Y output terminal is OFF; if it is not 0, the initial level of the Y output terminal is ON. Size specifies the number of comparison points and Array specifies the comparison point array. Among the output parameters, OutStatus indicates the output state of the comparison output terminal, and Index indicates the index of the next array coordinate point to be compared.

Assume that the comparison point array is P[4], and Parameter is set to a non-zero value. The instruction starts to run after the Enable input becomes active. The timing diagram is as follows.



As shown in the preceding figure, after the comparison output is completed, the state of the Y00 output terminal is still ON. If you want to set Y00 to OFF at this time, you need to call the ENC_ResetCompare instruction. For details, see the description of the ENC_ResetCompare instruction.

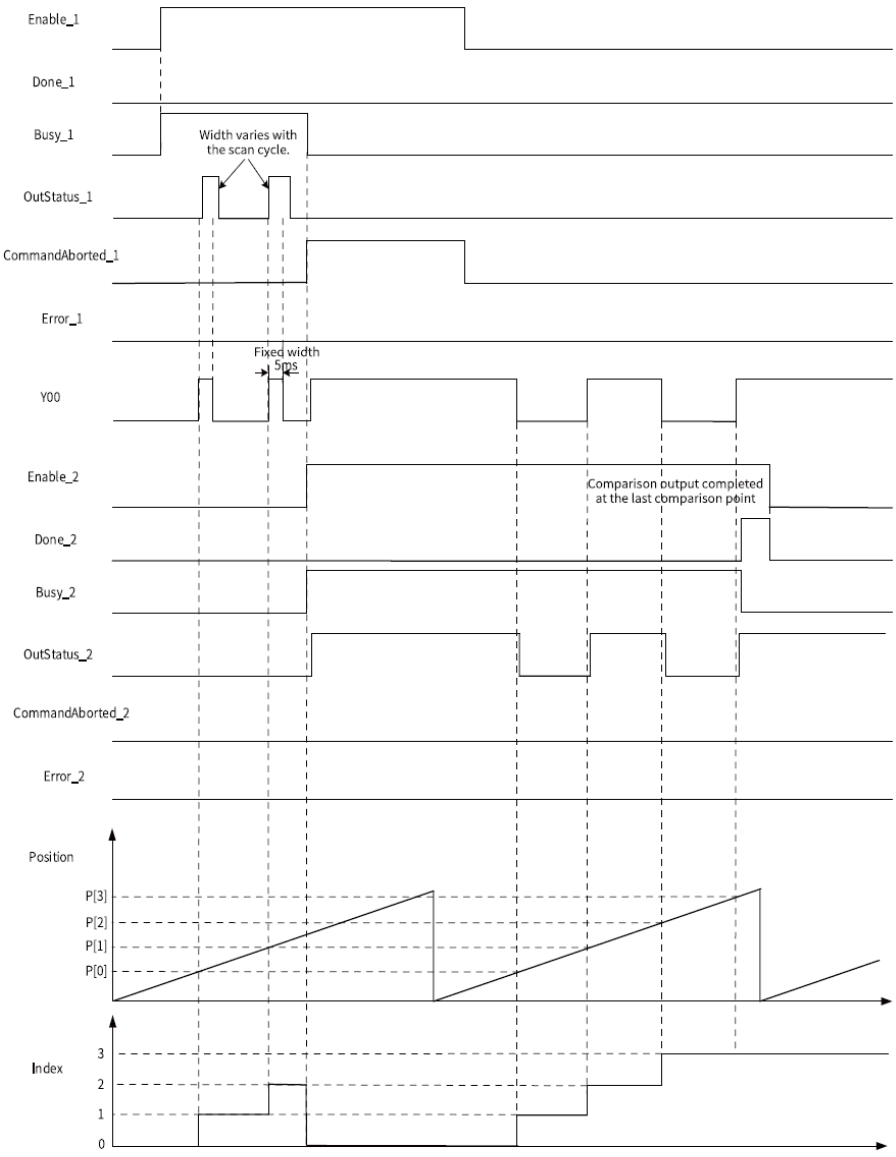
Before the comparison output is completed, if the Enable input is set to OFF, the comparison output stops and the output state of the Y00 terminal remains unchanged (for example, always ON). In this case, if you want to forcibly reset Y00 to OFF, you also need to call the ENC_ResetCompare instruction.

Multi-execution

If a new comparison output instruction is triggered during the comparison output process, the previous instruction being executed is aborted, its CommandAborted signal output becomes active, and the state of the Y output terminal is determined by the new instruction.

The following is an example.

In the first ENC_ArrayCompare instruction, Mode is set to 1 (time mode), the comparison point array is P[4], and the output time at each comparison point is 5 ms. The instruction starts to run after the Enable input becomes active. After the first instruction runs for a period of time, the second ENC_ArrayCompare instruction is triggered. In the second instruction, Mode is set to 3 (level mode), the comparison point array is P[4], and Parameter is set to a non-zero value. The timing diagram is as follows.



Function Description (Local Encoder Axis)

On the comparison output configuration interface of the local encoder axis, select **Compare Output Enable**, select an output terminal, and then select the pulse output unit (time or Unit).



Setting Comparison Points

Array in the instruction specifies the comparand array, and Size indicates the actual number of points to be compared. Ensure that the value of Size is less than or equal to the number of data entries specified by Array in the PLC program. If the value of Size is greater than the number of data entries specified by Array, the instruction will not report an error, but the array index out-of-bounds error and program execution error will occur inside the PLC.

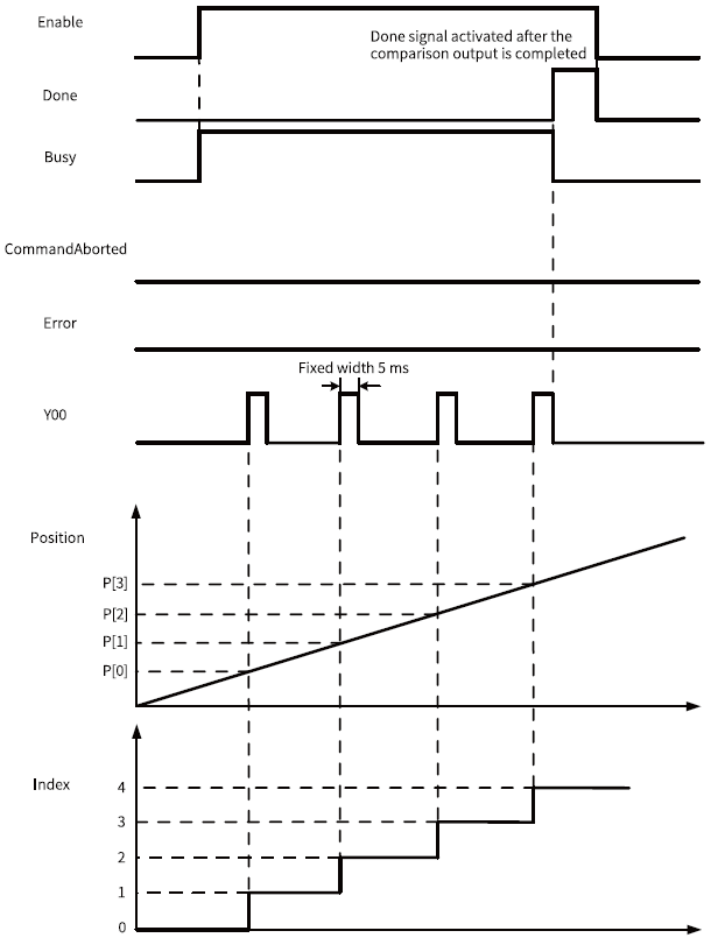
Setting the Comparison Output Mode

The basic principle of the comparison output is to set the DO terminal to high level after the encoder runs to the specified position. When the output is set to high level, the time duration or the number of consecutive pulses during which the output remains high level can be specified.

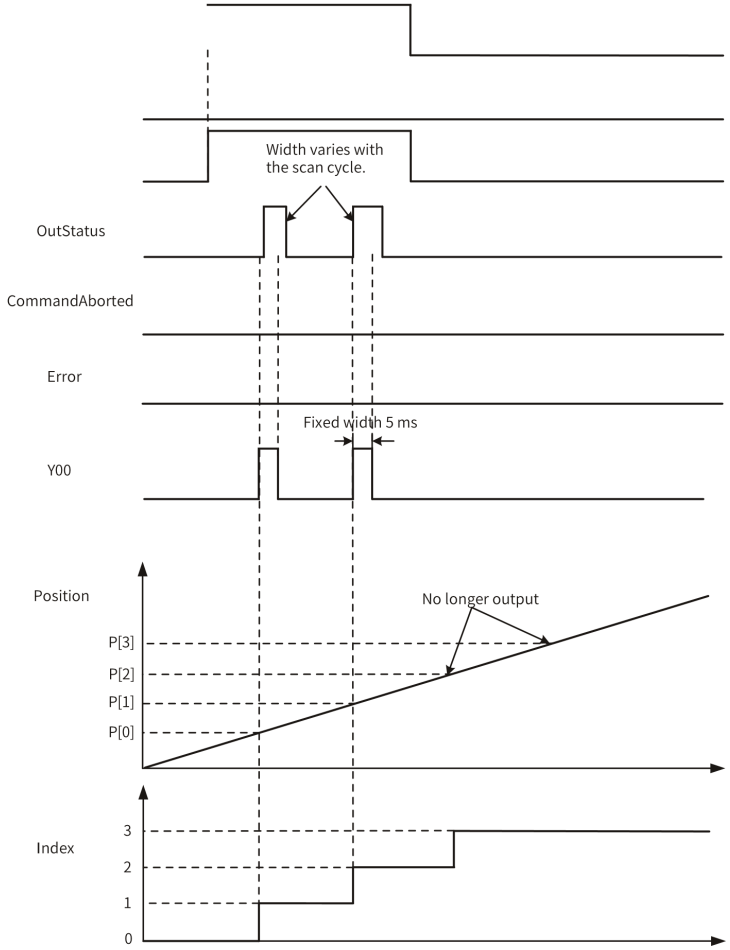
1. Time mode

   In time mode, the DO terminal outputs high level after the encoder axis reaches the comparison point. The time duration during which the output remains high level is configured on the background configuration interface. Size specifies the number of comparison points and Array specifies the comparison point array. Index indicates the next array coordinate point to be compared.

   Assume that the comparison point array is P[4], and the output time at each comparison point is 5 ms. The instruction starts to run after the Enable input becomes active. The timing diagram is as follows.



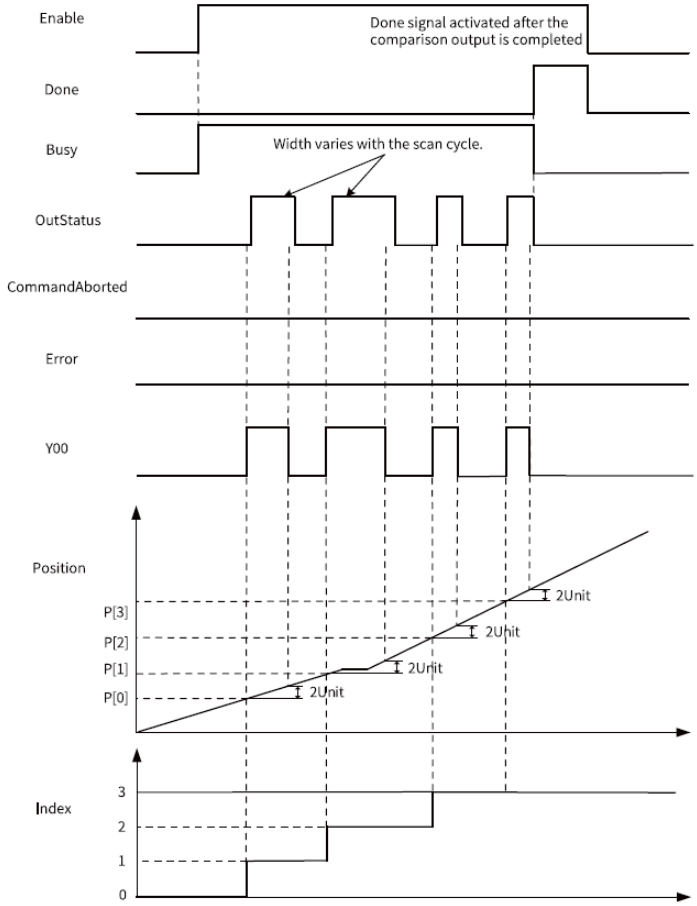
   Before the comparison output is completed, if the Enable input is set to OFF, the subsequent comparison points are not compared any more.



2. Pulse mode

When Unit is selected as the unit, the output remains high level for a specified number of encoder pulses. Size specifies the number of comparison points and Array specifies the comparison point array. Index indicates the next array coordinate point to be compared.

Assume that the comparison point array is P[4], and the output remains high level at each comparison point for 2 Units. The instruction starts to run after the Enable input becomes active. The timing diagram is as follows.



Before the comparison output is completed, if the Enable input is set to OFF, the subsequent comparison points are not compared any more, as in the case of the time mode.

Other Parameters

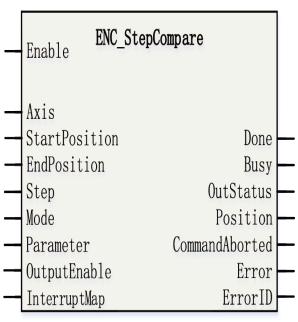
When OutputEnable is set to 1, the selected comparison output terminal generates comparison output signals. If OutputEnable is set to 0, no comparison output signals are generated.

InterruptMap is used to associate the comparison interrupt subprogram. When it is set to 0, no interrupt subprogram is associated. When it is set to 1 to 16, the specified interrupt subprogram is associated.

5.15.7 ENC_StepCompare

ENC_StepCompare -- Encoder one-dimensional step comparison

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_StepCompare	Encoder axis one-dimensional step comparison		ENC_StepCompare(Enable := ???, Axis := ???, StartPosition := ???, EndPosition := ???, Step := ???, Mode := ???, Parameter := , OutputEnable := , InterruptMap := ,

			Done => , Busy => , OutStatus => , Position => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	ENC_StepCompare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Encoder axis	No	-	-	_sENC_EXT_AXIS
S2	StartPosition	Comparison start position	No	-	-	REAL
S3	EndPosition	Comparison end position	No	-	-	REAL
S4	Step	Step	No	-	-	REAL
S5	Mode	Comparison mode 0: Reserved 1: Time control 2: Pulse control 3: Level control	Yes	-	0 to 3	INT
S6	Parameter	Control parameters Time control: output active duration, in μs Pulse control: output active pulse count, in Unit Level control: initial level; 0 indicates low level and non-zero indicates high level.	Yes	0	Positive number 0 Negative number	REAL
S7	OutputEnable	Hardware output enable (applicable to only the local encoder axis) 0: Disabled 1: Enabled	Yes	0	0 to 1	INT

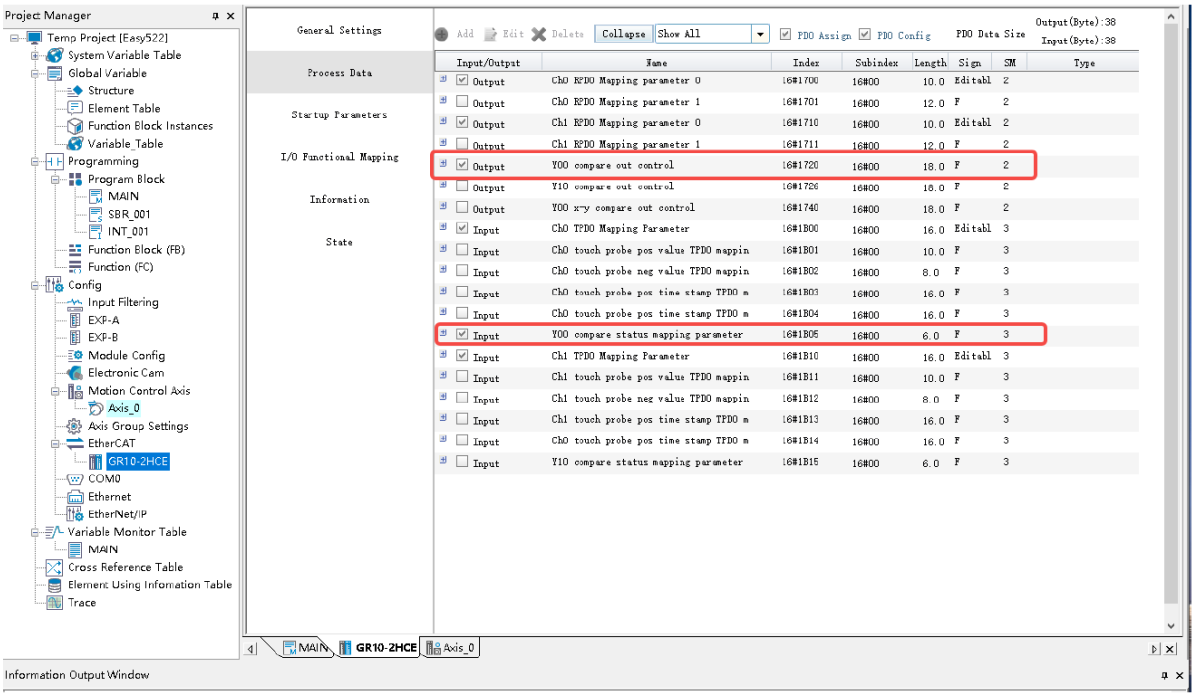
S8	Interrupt Map	Interrupt ID (applicable to only the local encoder axis) 0: No interrupt is generated. 1: Associate with comparison interrupt 1. … 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	OutStatus	Port output state (applicable to only the bus encoder axis)	Yes	OFF	ON OFF	BOOL
D4	Position	Next comparand	Yes	0		REAL
D5	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT

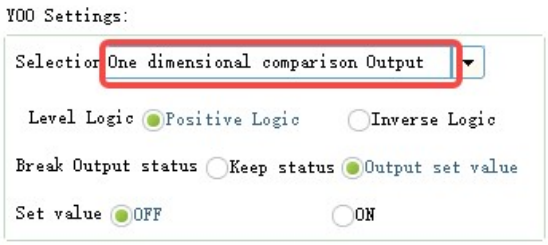
NOTE

*1: For details, see 5.15.22 Error Codes.

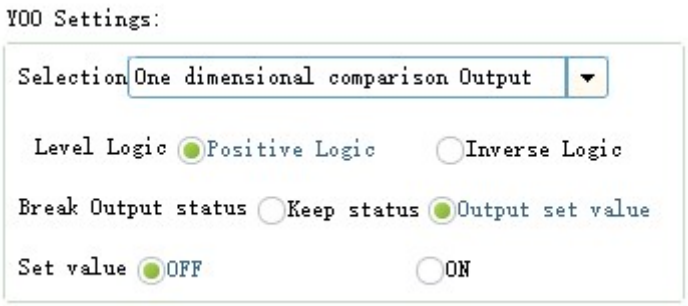
Function Description (Bus Encoder Axis)

When the bus encoder axis is associated with the first channel of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, Y00 of the GR10-2HCE module can be used for comparison output.





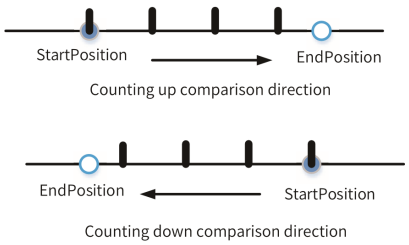
When the bus encoder axis is associated with the second channel of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, Y10 of the GR10-2HCE module can be used for comparison output.



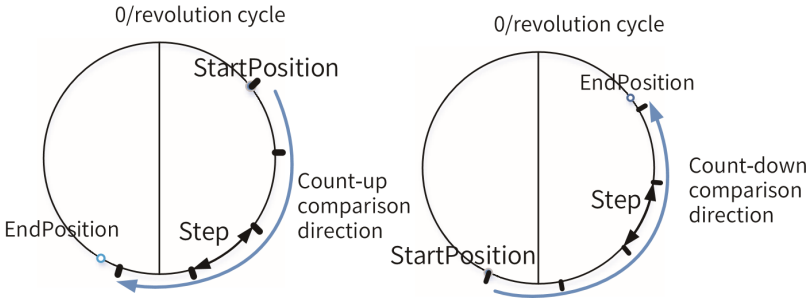
Setting Comparison Points

In this instruction, StartPosition specifies the start position of the comparison points, and EndPosition specifies the end position of the comparison points.

- In linear mode, when StartPosition is less than EndPosition, set Step to a positive value, which indicates the up-counting comparison mode. When StartPostion is greater than EndPosition, set Step to a negative value, which indicates the down-counting comparison mode.



- In ring mode, when StartPosition is less than EndPosition, set Step to a positive value, which indicates the up-counting comparison mode. When StartPostion is greater than EndPosition, set Step to a negative value, which indicates the down-counting comparison mode.



NOTE

*1: For details, see 5.15.22 Error Codes.

Selecting the Comparison Mode

The step comparison instruction supports the time control, pulse control, and level control modes, and the mode is specified by the Mode parameter of the instruction. For details about the three modes, see the description of the ENC_ArrayCompare instruction.

Multi-execution

If a new comparison output instruction is triggered during the comparison output process, the previous instruction being executed is aborted, its CommandAborted signal output becomes active, and the state of the Y output terminal is determined by the new instruction.

Function Description (Local Encoder Axis)

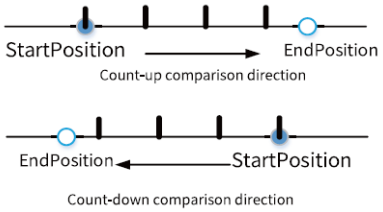
On the comparison output configuration interface of the local pulse axis, select **Compare Output Enable**, select an output terminal, and then select the pulse output unit (time or Unit).



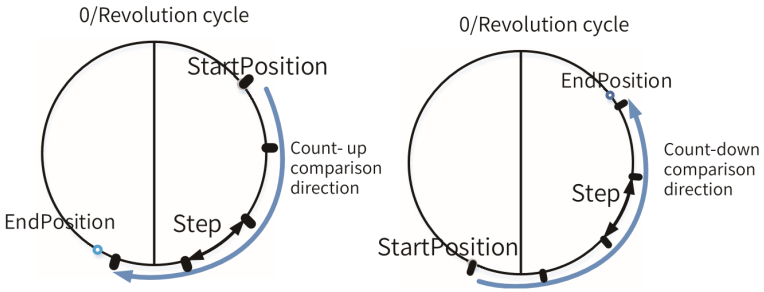
Setting Comparison Points

In this instruction, StartPosition specifies the start position of the comparison points, and EndPosition specifies the end position of the comparison points. Step must be set to a positive value.

In linear mode, when StartPosition is less than EndPosition, the up-counting comparison mode is used. When StartPostion is greater than EndPosition, the down-counting comparison mode is used.



In ring mode, when StartPosition is less than EndPosition, the up-counting comparison mode is used. When StartPostion is greater than EndPosition, the down-counting comparison mode is used.



NOTE

The first comparison point must be the point specified by StartPosition, but the last comparison point is not necessarily the point specified by EndPosition. For example, assume that the start point is 10, the end point is 25, and the step is 10. In this case, the comparison output signal is generated only at position 10 and position 20.

Selecting the Comparison Mode

The step comparison instruction supports the time control and pulse control modes, which is configured on the background configuration interface. For details about the two modes, see the description of the ENC_StepCompare instruction.

Other Parameters

When OutputEnable is set to 1, the selected comparison output terminal generates comparison output signals. If OutputEnable is set to 0, no comparison output signals are generated.

InterruptMap is used to associate the comparison interrupt subprogram. When it is set to 0, no interrupt subprogram is associated. When it is set to 1 to 16, the specified interrupt subprogram is associated.

Multi-execution

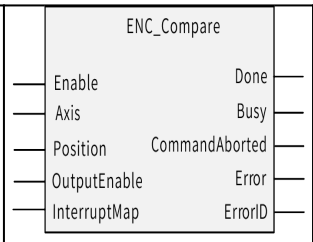
If a new comparison output instruction is triggered during the comparison output process, the previous instruction being executed is aborted, its CommandAborted signal output becomes active, and the state of the Y output terminal is determined by the new instruction.

5.15.8 ENC_Compare

ENC_Compare -- Single-point comparison output

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_Compare			-

	Single-point comparison output		

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	ENC_Compare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Encoder axis	No	-	-	_sENC_EXT_AXIS
S2	Position	Comparison position	No	-	-	REAL
S3	OutputEnable	Hardware output enable (applicable to only the local encoder axis) 0: Disabled 1: Enabled	Yes	0	0 to 1	INT16
S4	InterruptMap	Interrupt ID (applicable to only the local encoder axis) 0: No interrupt is generated. 1: Associate with comparison interrupt 1. … 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT16
D1	Done	Completion flag	Yes	FALSE	ON or OFF	BOOL
D2	Busy	Executing	Yes	FALSE	ON or OFF	BOOL
D3	Command Aborted	Abortion of execution	Yes	FALSE	ON or OFF	BOOL
D4	Error	Error	Yes	FALSE	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	-	INT

Function Description (Bus Encoder Axis)

The bus encoder axis does not support this instruction.

Function Description (Local Encoder Axis)

On the comparison output configuration interface of the local pulse axis, select Compare Output Enable, select an output terminal, and then select the pulse output unit (time or Unit).



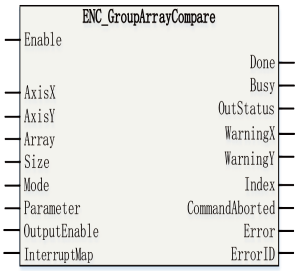
When OutputEnable is set to 1, the configured comparison output terminal generates comparison output signals. If OutputEnable is set to 0, no comparison output signals are generated.

InterruptMap is used to associate the comparison interrupt subprogram. When it is set to 0, no interrupt subprogram is associated. When it is set to 1 to 16, the specified interrupt subprogram is associated.

5.15.9 ENC_GroupArrayCompare

ENC_GroupArrayCompare -- Encoder two-dimensional array comparison (applicable to only the bus encoder axis)

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_GroupArrayCompare	Two-dimensional array comparison of the encoder axis group (applicable to only the bus encoder axis)		ENC_GroupArrayCompare(Enable := ???, AxisX := ???, AxisY := ???, Array := ???, Size := ???, Mode := ???, Parameter := , OutputEnable := , InterruptMap := , Done => , Busy => , OutStatus => , WarningX => , WarningY => , Index => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-

32-bit Instruction	ENC_GroupArrayCompare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisX	Encoder axis (x-axis)	No	-	-	_sENC_EXT_AXIS
S2	AxisY	Encoder axis (y-axis)	No			_sENC_EXT_AXIS
S3	Array	Comparand array	No	-	-	_sPoint2D[0..1000]
S4	Size	Number of comparands	No	-	1 to 1000	INT
S5	Mode	Comparison mode 0: Configuration mode (applicable to only the local encoder axis) 1: Time control 2: Reserved 3: Level control	No		0 to 3	INT
S6	Parameter	Control parameters Time control: output active duration, in μs Level control: initial level; 0 indicates low level and non-zero indicates high level.	Yes	0	Positive number 0 Negative number	REAL
S7	OutputEnable	Reserved	Yes	0	0 to 1	INT
S8	Interrupt Map	Reserved	Yes	0	0 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	OutStatus	Port output state	Yes	OFF	ON OFF	BOOL

D4	WarningX	Warning output of x-axis	Yes	OFF	ON OFF	BOOL
D5	WarningY	Warning output of y-axis	Yes	OFF	ON OFF	BOOL
D6	Index	Index of the next comparand	Yes	0	0 to 999	INT
D7	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D8	Error	Error	Yes	OFF	ON OFF	BOOL
D9	ErrorID	Error code	Yes	0	*1	INT

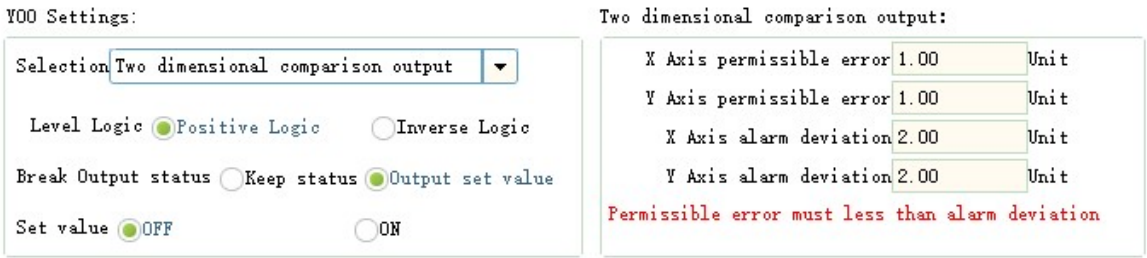
NOTE

*1: For details, see 5.15.22 Error Codes.

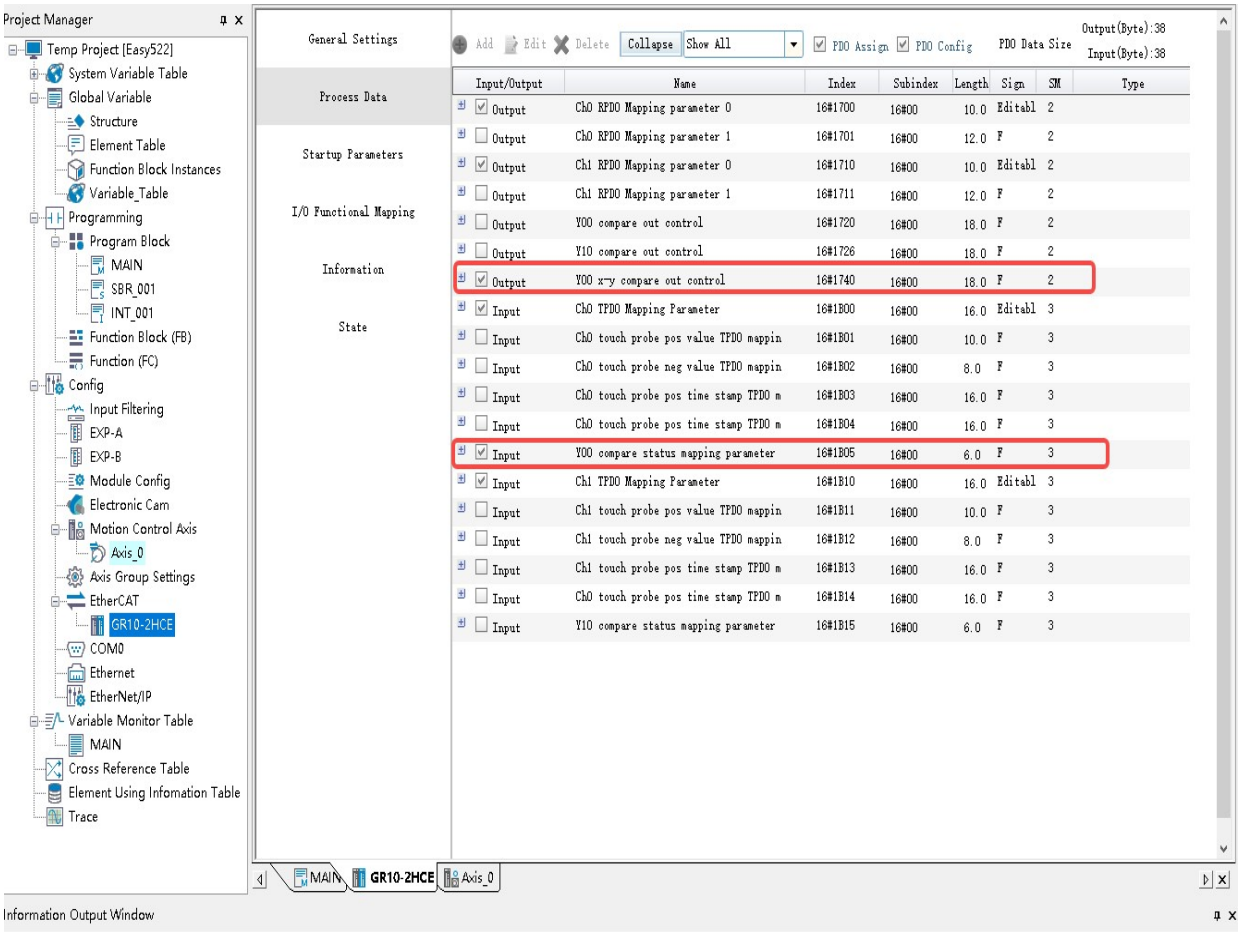
Function Description (Bus Encoder Axis)

The two channels of the GR10-2HCE module can work together to implement the two-dimensional comparison output function. The first channel is used as the x-axis input, and the second channel is used as the y-axis input. Therefore, when calling this instruction, ensure that the encoder axes specified by AxisX and AxisY are bound to the same GR10-2HCE module. Moreover, ensure that AxisX is bound to the first channel and AxisY is bound to the second channel.

You need to assign the two-dimensional comparison output function to the Y00 output terminal on the AxisX mode/parameter configuration interface, and set the allowable error and warning deviation for each axis of the two-dimensional comparison output.



The PDO options as shown in the following figure need be selected on the process data page of the GR10-2HCE module. In this case, Y00 of the GR10-2HCE module can be used as a two-dimensional comparison output terminal.



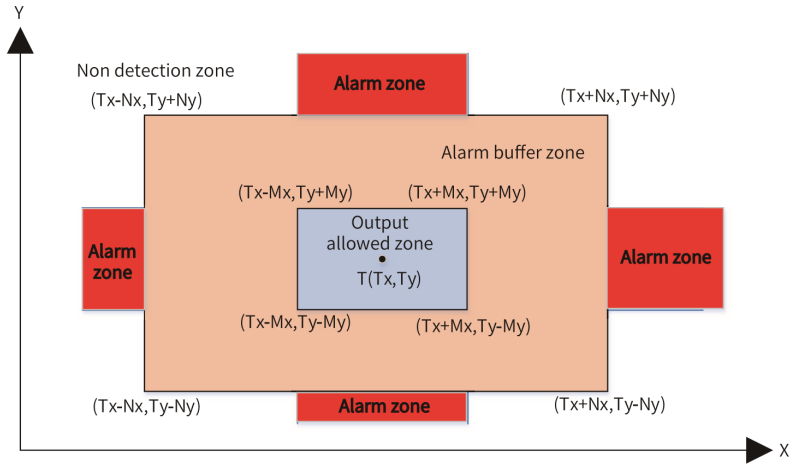
Two-dimensional Comparison Output Principles

Set T (Tx, Ty) as the target point on the plane, M (Mx, My) as the maximum allowable position error, and N (Nx, Ny) as the warning buffer detection range.

As shown in the following figure, when the system runs to the allowable output area, it is considered to have entered the target area. After the system enters the target area, the controller selects the output based on the motion profile of the encoder axis to achieve optimal control (Note 1).

When the x-axis (y-axis) has entered the allowable output area, but the y-axis (x-axis) has not entered the warning buffer area, as shown in the areas in red in the following figure, the WarningY (WarningX) output of the instruction becomes active and remains active until the y-axis (x-axis) enters the warning buffer area.

In the non-detection area and warning buffer area, since the comparison output conditions are not satisfied, there is no comparison output signal or warning signal.



> **NOTE**
>
> After the comparison point areas are reached, if the encoder input does not change any more, the comparison output is generated after 1 second.

Comparison Output Modes

The two-dimensional comparison output instruction supports the time control and level control modes. For details, see the description of the ENC_ArrayCompare instruction.

Multi-execution

If a new comparison output instruction is triggered during the comparison output process, the previous instruction being executed is aborted, its CommandAborted signal output becomes active, and the state of the Y output terminal is determined by the new instruction.

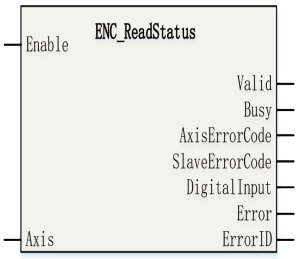
Function Description (Local Encoder Axis)

The local encoder axis does not support this instruction.

5.15.10 ENC_ReadStatus

ENC_ReadStatus -- Read encoder state

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_ReadStatus	Read encoder axis state (applicable to only the bus encoder axis)		ENC_ReadStatus(Enable := ???, Axis := ???, Valid => , Busy => , AxisErrorCode => , SlaveErrorCode => , DigitalInput => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	ENC_ReadStatus: Continuous execution
32-bit Instruction	-

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Bus encoder axis	No	-	-	_sENC_EXT_AXIS
D1	Valid	Active	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	AxisErrorCode	Axis error code	Yes	0	-	INT
D4	SlaveErrorCode	Drive error code	Yes	0	-	INT
D5	DigitalInput	DI terminal state Bit 0: CHn-X0 terminal state Bit 1: CHn-X1 terminal state Bit 2: CHn-X2 terminal state Bit 3: CHn-X3 terminal state Others: Reserved	Yes	0	-	INT
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see 5.15.22 Error Codes.

Function Description (Bus Encoder Axis)

The ENC_ReadStatus instruction reads the state of a bus encoder axis.

AxisErrorCode displays the error code of the bus encoder axis. For details, see the list of error codes.
SlaveErrorCode displays the error code of the GR10-2HCE module. For details, see the list of error codes in the *GR10-2HCE User Manual*.

DigitalInput displays the DI terminal state of the GR10-2HCE module. When the axis is bound to the CH0 channel, it displays the states of X00, X01, X02, and X03. When the axis is bound to the CH1 channel, it displays the states of X10, X11, X12, and X13.

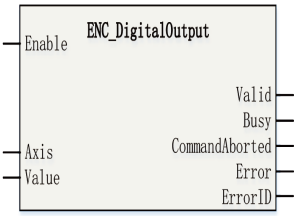
Function Description (Local Encoder Axis)

The local encoder axis does not support this instruction.

5.15.11 ENC_DigitalOutput

ENC_DigitalOutput -- Encoder DO control

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_DigitalOutput	Encoder DO control (applicable to only the bus encoder axis)		ENC_DigitalOutput(Enable := ???, Axis := ???, Value := , Valid => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	ENC_DigitalOutput: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Bus encoder axis	No	-	-	_sENC_EXT_AXIS
S2	Value	DO terminal set value Bit 0: CHn-Y0 output state Bit 1: CHn-Y1 output state Bit 2: CHn-Y2 output state Others: Reserved	Yes	0	-	INT

D1	Valid	Output active	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see 5.15.22 Error Codes.

Function Description (Bus Encoder Axis)

The ENC_DigitalOutput instruction sets the DO terminal state of the GR10-2HCE module.

When the bus encoder axis is bound to the CH0 channel of the GR10-2HCE module, the instruction controls the states of Y00, Y01, and Y02. When the axis is bound to the CH1 channel of the GR10-2HCE module, the instruction controls the states of Y10, Y11, and Y12.

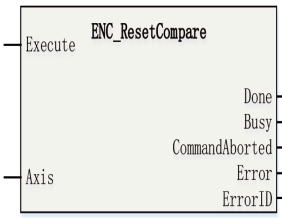
Function Description (Local Encoder Axis)

The local encoder axis does not support this instruction.

5.15.12 ENC_ResetCompare

ENC_ResetCompare -- Reset the encoder comparison output

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_ResetCompare	Reset the encoder axis comparison output (applicable to only the bus encoder axis)		ENC_ResetCompare(Execute := ???, Axis := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	ENC_ResetCompare: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Bus encoder axis	No	-	-	_sENC_EXT_AXIS
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

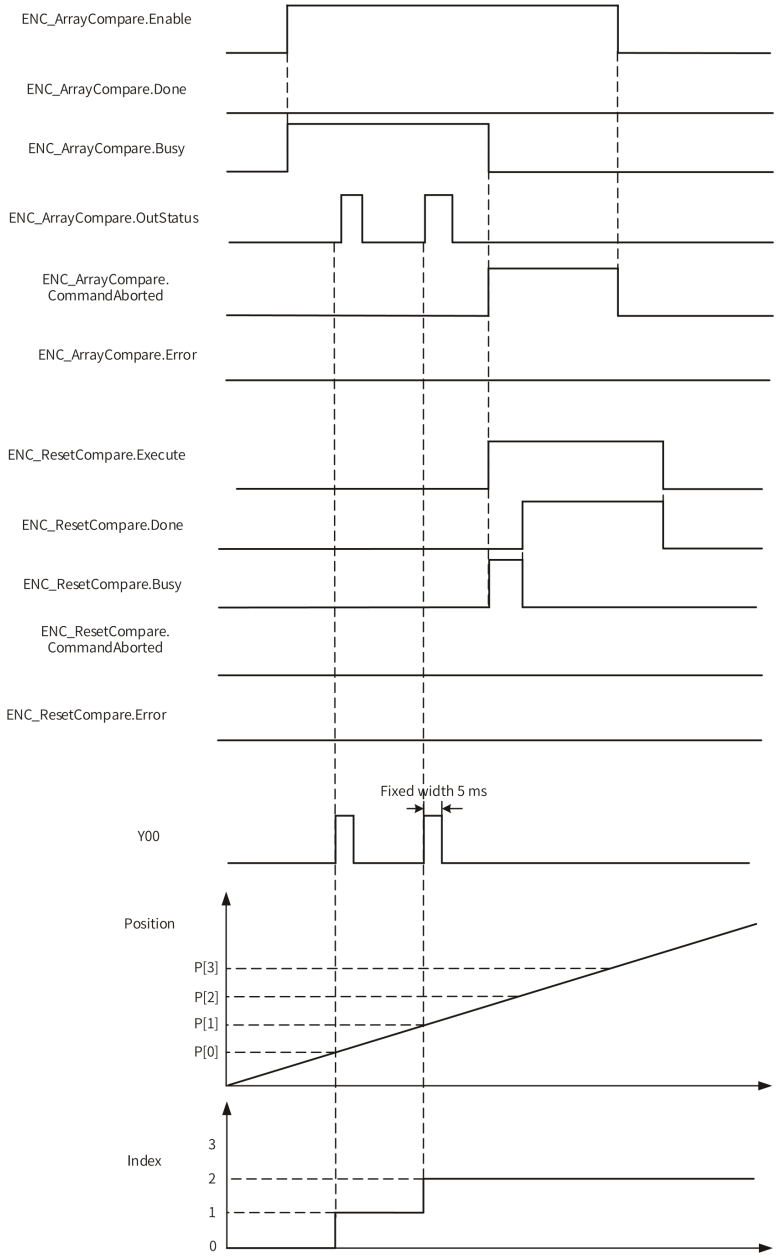
NOTE

*1: For details, see 5.15.22 Error Codes.

Function Description (Bus Encoder Axis)

The ENC_ResetCompare instruction aborts the execution of three comparison output instructions (ENC_StepCompare, ENC_GroupArrayCompare, and ENC_ArrayCompare) and sets the comparison output terminal to OFF.

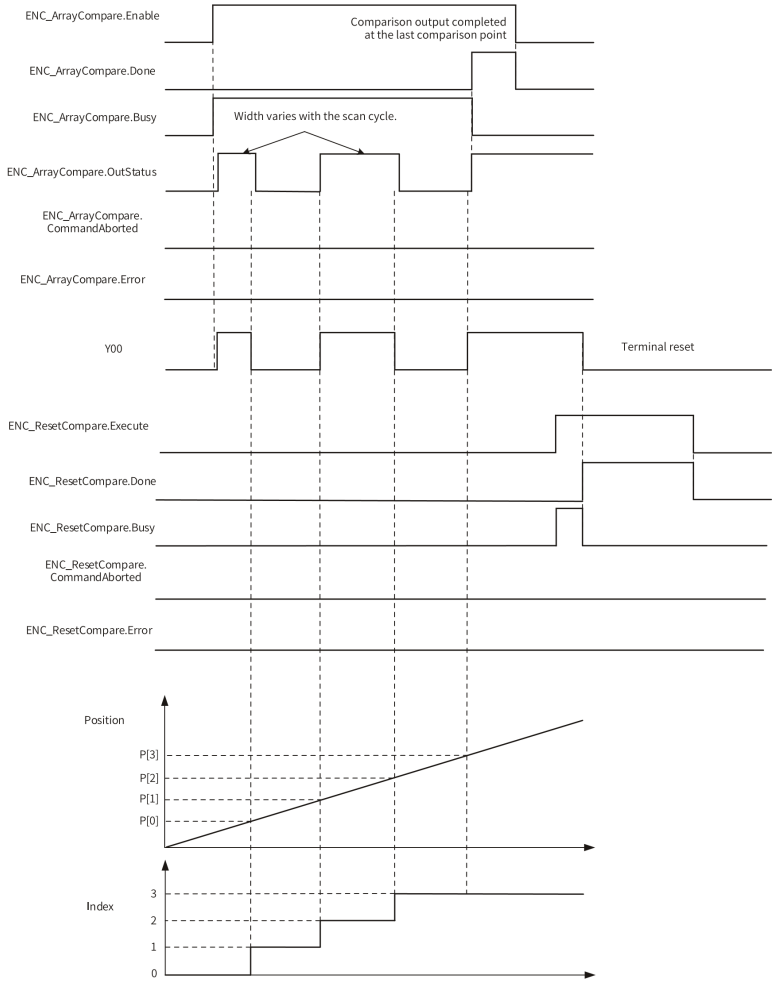
1. Abortion rules in time or pulse mode
   When the comparison output instruction is set to time or pulse mode, if this instruction is called when the Done signal of the comparison output instruction is OFF, the comparison output instruction is aborted, its CommandAborted signal output becomes active, and the comparison output terminal is forcibly set to OFF. After that, the Done signal output of ENC_ResetCompare instruction becomes active.



2. Abortion rules in level mode

   When the comparison output instruction is set to level mode, if this instruction is called when the Done signal of the comparison output instruction is OFF, the instruction processing is the same as that in time or pulse mode mentioned above.

   If the ENC_ResetCompare instruction is called when the Done signal of the comparison output instruction is ON, that is, when the comparison output is completed, the output signals (Done, Busy, and CommandAborted) remain unchanged, the controlled comparison output terminal is forcibly set to OFF, and the Done signal output of the instruction becomes ON.



Multi-execution

If a new comparison output reset instruction is triggered during the comparison reset process, the previous instruction being executed is aborted, and its CommandAborted signal output becomes active.

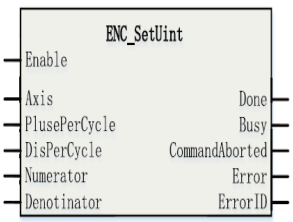
Function Description (Local Encoder Axis)

The local encoder axis does not support this instruction.

5.15.13 ENC_SetUnit

ENC_SetUnit -- Set gear ratio

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_SetUnit	Set gear ratio		ENC_SetUnit(Enable := ???, Axis := ???, PlusePerCycle := ???, DisPerCycle := ???, Numerator := ???, Denotinator := ???, Done => , Busy => ,

Instruction	Name	LD Expression	LiteST Expression
			CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	ENC_SetUnit: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name	No	-		_sENC_EXT_AXIS
S2	PlusePerCycle	Number of pulses per revolution of the encoder	No	-	Positive number	DINT
S3	DisPerCycle	Distance per revolution of the worktable	No	-	0.01 to 9999999	REAL
S4	Numerator	Gear ratio (numerator)	No	1	Positive number	DINT
S5	Denotinator	Gear ratio (denominator)	No	1	Positive number	DINT
D1	Done	Completion flag	Yes	FALSE	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	FALSE	ON or OFF	BOOL
D3	CommandAborted	Abortion	Yes	FALSE	ON or OFF	BOOL
D4	Error	Error flag	Yes	FALSE	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	-	INT

Function Description (Bus Encoder Axis)

The bus encoder axis does not support this instruction.

Function Description (Local Encoder Axis)

The ENC_SetUnit instruction reconfigures the gear ratio of the local encoder axis with the counting function disabled after the PLC is powered on, downloads the program, or executes a RUN or STOP operation.

The gear ratio of the local encoder axis is calculated as follows:

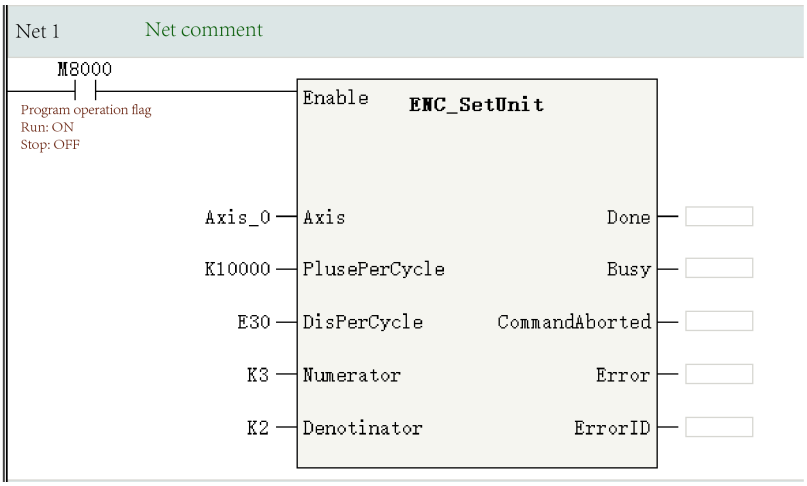
$$\text{Gear ratio} = \frac{PlusePreCycle * Numerator}{DisPerCycle * Denotinator}$$

Example

This example describes how the local encoder axis parameters are automatically modified after power-on of the PLC.

Parameter	Value
Number of pulses per revolution of the encoder	10000
Distance per revolution of the worktable	30
Gear ratio (numerator)	3
Gear ratio (denominator)	2

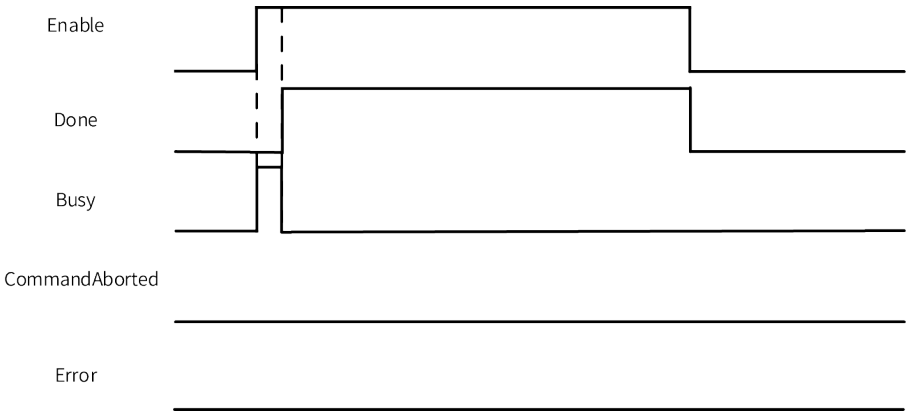
Add the following program in the PLC. Use M8000 to trigger the instruction.



NOTE

When the program is running, calling this instruction re-initializes the local encoder axis and causes an abrupt change to the feedback position of the local encoder axis.

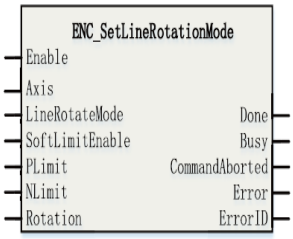
Timing Diagram



5.15.14 ENC_SetLineRotationMode

ENC_SetLineRotationMode -- Set axis operation mode

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_SetLineRotationMode	Set axis operation mode		ENC_SetLineRotationMode(Enable := ???, Axis := ???, LineRotateMode := ???, SoftLimitEnable := ???, Plimit := ???, Nlimit := ???, Rotation := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	ENC_SetLineRotationMode: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name	No	-		_sENC_EXT_AXIS
S2	LineRotateMode	Linear or rotation mode 0: Linear mode 1: Rotation mode	No	-	0 to 1	INT

S3	SoftLimitEnable	Limiting enable in linear mode ON: Enabled OFF: Disabled	No	-	Positive number	BOOL
S4	PLimit	Positive limit in linear mode	No	-		REAL
S5	NLimit	Negative limit in linear mode	No	-		REAL
S6	Rotation	Rotation cycle in rotation mode	No	-		REAL
D1	Done	Completion flag	Yes	FALSE	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	FALSE	ON or OFF	BOOL
D3	CommandAborted	Abortion	Yes	FALSE	ON or OFF	BOOL
D4	Error	Error flag	Yes	FALSE	ON or OFF	BOOL
D5	ErrorID	Error code	Yes	0	-	INT

Function Description (Bus Encoder Axis)

The bus encoder axis does not support this instruction.

Function Description (Local Encoder Axis)

The ENC_SetLineRotationMode instruction reconfigures the linear or rotation mode of the local encoder axis with the counting function disabled after the PLC is powered on, downloads the program, or executes a RUN or STOP operation.

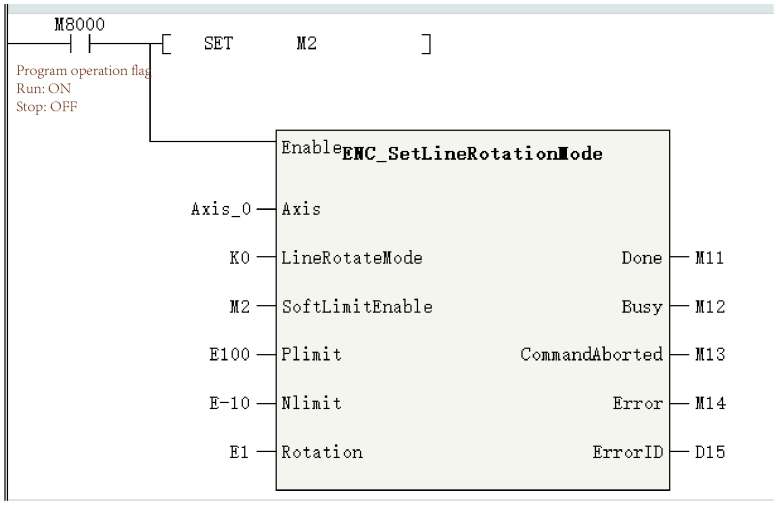
When LineRotateMode is set to 0, the local encoder axis is in linear mode.

In linear mode, when SoftLimitEnable is set to OFF, limiting is disabled. When SoftLimitEnable is set to ON, limiting is enabled. In this case, PLimit represents the positive limit, and NLimit represents the negative limit.

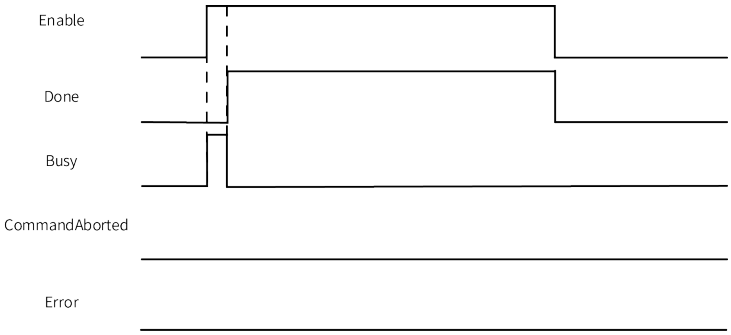
When LineRotateMode is set to 1, the local encoder axis is in rotation mode. In this case, Rotation represents the rotation cycle.

Example

In this example, the local encoder axis switches to linear mode automatically after power-on of the PLC. Limiting is enabled. The positive limit is 100, and the negative limit is -10. The program is as follows.



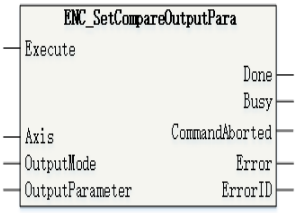
Timing Diagram



5.15.15 ENC_SetCompareOutputPara

ENC_SetOutputPara -- Set comparison output configuration parameters

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ENC_SetOutputPara	Set comparison output configuration parameters		-

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	ENC_SetCompareOutputPara: Set comparison output configuration parameters					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis		No	-	0 to 32,767	

		Axis name or axis ID				_sENC_EXT_AXIS
S2	OutputMode	Comparison mode • 0: Time unit • 1: Pulse unit	No	-	0 to 1	INT
S3	OutputParameter	Control parameters • Time control: output active duration, in ms • Pulse control: output active pulse count, in Unit	No	-	Positive number	REAL
D1	Done	Completion flag	Yes	FALSE	TRUE FALSE	BOOL
D2	Busy	Busy flag	Yes	FALSE	TRUE FALSE	BOOL
D3	CommandAborted	Abortion flag	Yes	FALSE	TRUE FALSE	BOOL
D4	Error	Error flag	Yes	FALSE	TRUE FALSE	BOOL
D5	ErrorID	Error code	Yes	0	-	INT

Function Description (Bus Encoder Axis)

The bus encoder axis does not support this instruction.

Function Description (Local Encoder Axis)

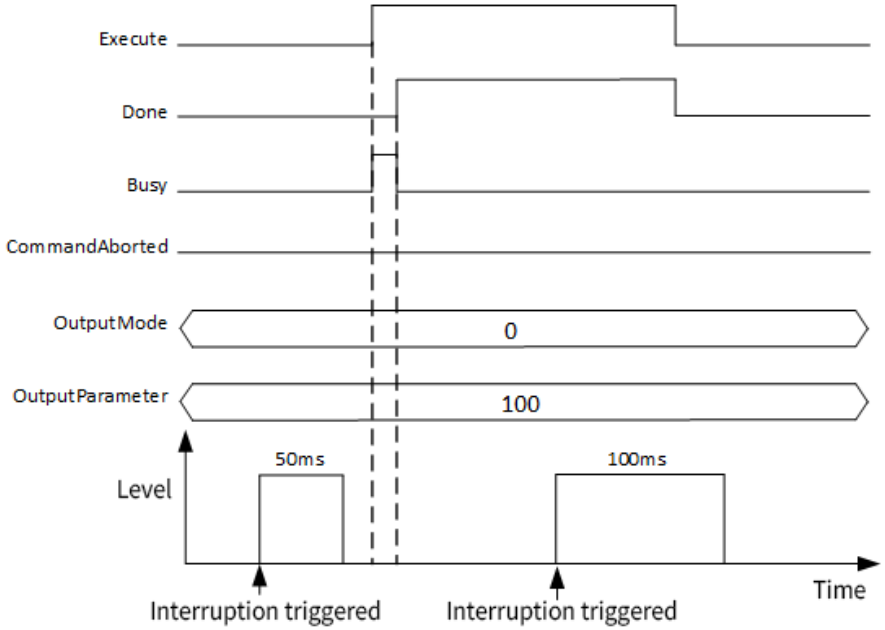
The ENC_SetCompareOutputPara instruction reconfigures the comparison output configuration parameters (comparison mode and pulse width) of the local encoder axis during project running.

If OutputMode is set to 0, the time unit in ms is used. The valid range of this parameter is 0.1 ms to 6553.5 ms. If OutputMode is set to 1, the pulse unit in Unit is used. Ensure that the value after conversion to the pulse unit is in the range of 1 to 65535. For example, the Unit is 1 mm, which is converted to 50 pulses. If the pulse width is 5 mm, the encoder outputs 250 pulses. If the parameter value is out of range, the pulse width must be corrected to the extreme value.

After the parameters are configured, to implement the position comparison output function of the local encoder axis, the Easy series PLC must work with the ENC_Compare, ENC_ArrayCompare, and ENC_StepCompare instructions, and the H5U series PLC must work with the HC_Compare, HC_ArrayCompare, and HC_StepCompare instructions.

> **CAUTION**
>
> After the instruction is triggered in pulse control mode, conversion to the output pulse width is performed based on the gear ratio of the current cycle.

Timing Diagram



Error Codes

Error Code	Description
120	OutputMode is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
121	OutputParameter is set incorrectly in the instruction. Ensure that the value of this parameter is a positive number and the value is greater than 0 after unit conversion.
122	Failed to set the pulse width for comparison output.

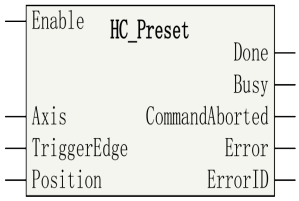
5.15.16 HC_Preset

Description

This instruction sets the counter value as the preset value according to the trigger signal.

HC_Preset: High-speed counter preset

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
HC_Preset	High-speed counter preset		HC_Preset(Enable := ???, Axis := ???, TriggerEdge := , Position := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	HC_Preset: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT
S2	TriggerEdge	Trigger edge 0: Triggered on the rising edge of the instruction 1: Triggered on the rising edge of the input terminal X 2: Triggered on the falling edge of the input terminal X 3: Triggered on the rising or falling edge of the input terminal X	Yes	0	0 to 3	INT
S3	Position	Preset position (unit: Unit)	No	-	-	REAL
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL

D3	Command Aborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Instruction Error Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	√	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	√[1]	-	√	-	√	-	-	-	-
D3	√[1]	-	√	-	√	-	-	-	-
D4	√[1]	-	√	-	√	-	-	-	-
D5	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Function

The HC_Preset instruction assigns the counter axis position according to the preset condition.

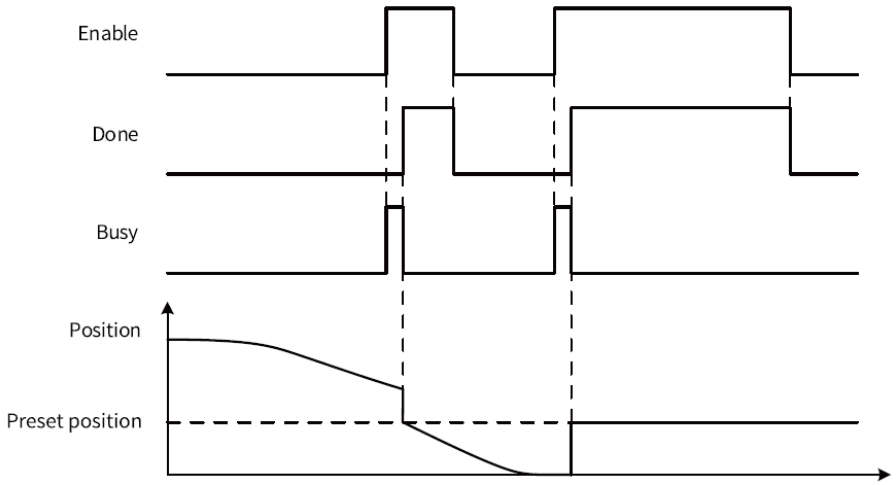
TriggerType specifies the preset condition, including the rising edge of the instruction or external terminal input.

Items	Settings	Definitions
TriggerType	0	Triggered on the rising edge of the instruction flow
	1	Triggered on the rising edge of the external input X
	2	Triggered on the falling edge of the external input X
	3	Triggered on the rising or falling edge of the external input X

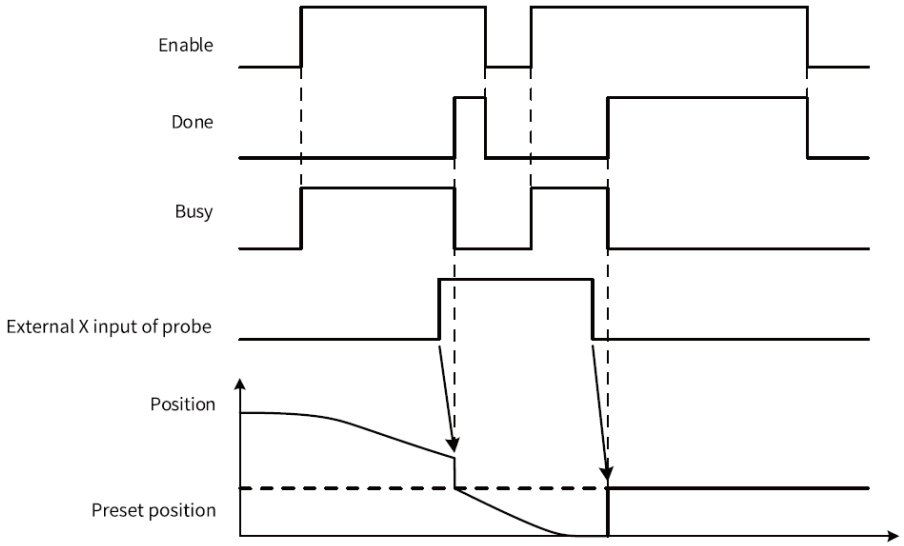
When the preset condition is set to the external input X, you need to select the preset function in counter parameter configuration and select the input terminal and trigger condition. The input terminal can be set to any one of X0 to X7, and the trigger condition can be the rising edge or falling edge.

Sequence Diagram

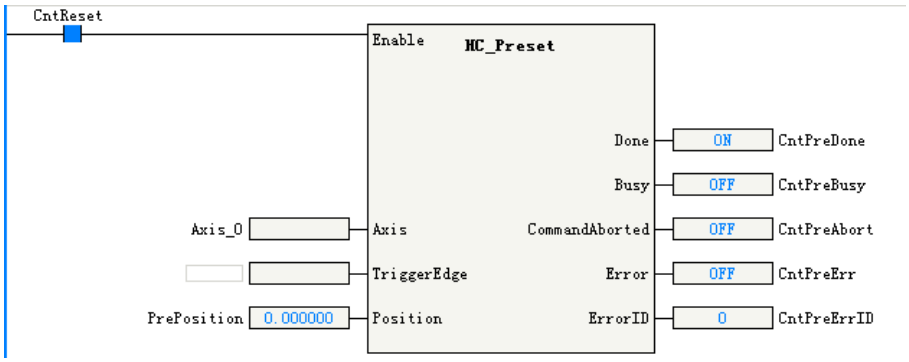
- The timing diagram of the instruction is as follows when TriggerEdge is set to the rising edge of the instruction (TriggerEdge = 0).



- The timing diagram of the instruction is as follows when TriggerEdge is set to the rising or falling edge of the external input X (TriggerEdge = 3).



Example



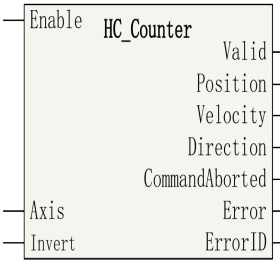
5.15.17 HC_Counter

Description

This instruction controls the high-speed counter to start or stop counting.

HC_Counter: High-speed counter enable

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
HC_Counter	High-speed counter enable		HC_Counter(Enable := ???, Axis := ???, Invert := , Valid => , Position => , Velocity => , Direction => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	HC_Counter: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT

S2	Invert	Counting inversion	Yes	0	0 to 1	INT
D1	Valid	Active state, which is ON when the counter enters the counting state	Yes	OFF	ON OFF	BOOL
D2	Position	Current position (unit: Unit)	Yes	0	-	REAL
D3	Velocity	Current velocity (unit: Unit/s)	Yes	0	-	REAL
D4	Direction	Counting direction	Yes	OFF	ON OFF	BOOL
D5	Command Aborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Instruction Error Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	-	-	-	√	√	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	-	√	-	√	-	-	-	-
D5	√[1]	-	√	-	√	-	-	-	-
D6	√[1]	-	√	-	√	-	-	-	-
D7	-	-	-	√	√	√	-	-	-

> **NOTE**
>
> [1] The X element is not supported.

Function

The HC_Counter instruction implements position counting and velocity measurement of the counter axis.

The counter axis position (unit: Unit) varies within a certain range based on the mode setting.

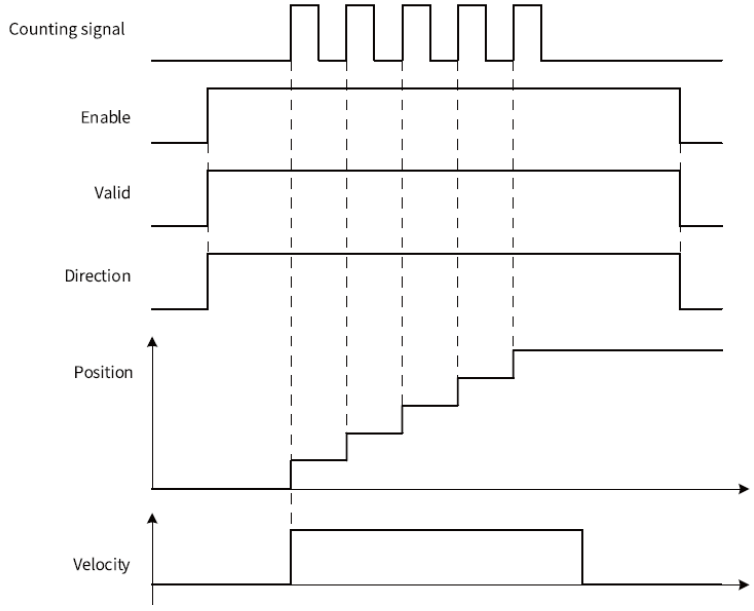
Invert (counting inversion)

Invert specifies the counting direction of the counter. The following table lists the counting directions of different counting modes. Modification on Invert takes effect only after this function block instruction is enabled again.

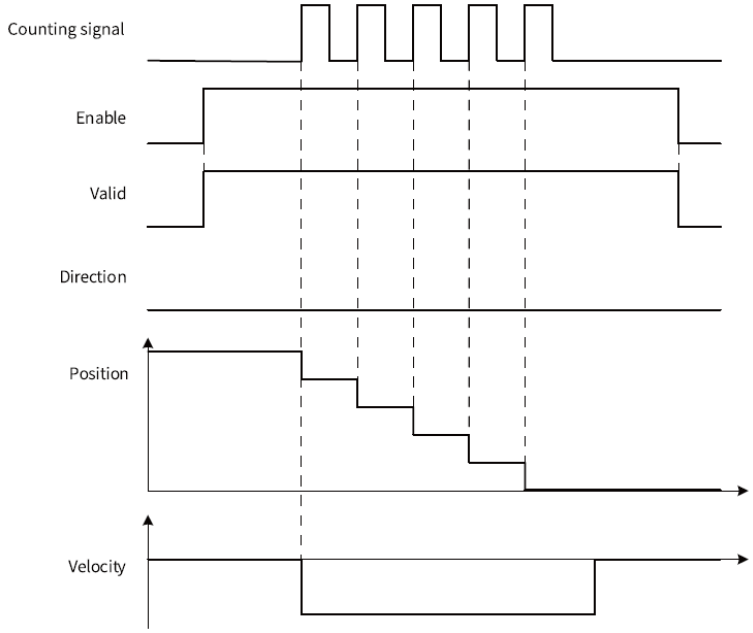
Invert	Phase A/B	Pulse+Direction	CW/CCW	Single-phase counter
0	Phase A leading phase B, counting up Phase B leading phase A, counting down	Direction signal = OFF, counting down Direction signal = ON, counting up	Phase A, counting up Phase B, counting down	Counting up
1	Phase A leading phase B, counting down Phase B leading phase A, counting up	Direction signal = OFF, counting up Direction signal = ON, counting down	Phase A, counting down Phase B, counting up	Counting down

Sequence Diagram

- In pulse+direction mode, if the direction signal is ON and Invert is set to 0, or the direction signal is OFF and Invert is set to 1, the counter counts up, as shown in the following figure.

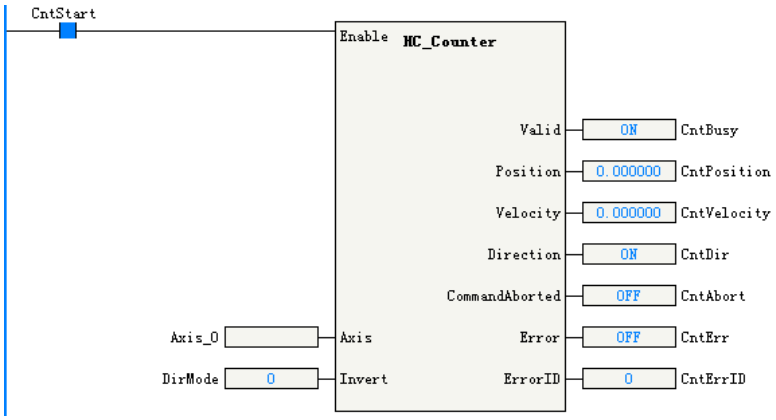


- In pulse+direction mode, if the direction signal is ON and Invert is set to 1, or the direction signal is OFF and Invert is set to 0, the counter counts down, as shown in the following figure.



Example

The counter axis velocity is the current real-time velocity (unit: Unit/s). The minimum velocity that can be measured by the counter axis is the velocity corresponding to one pulse of the counter within 1s. If one pulse of the counter corresponds to 0.1 Unit, the minimum velocity that can be measured is 0.1 Unit/s.



5.15.18 HC_TouchProbe

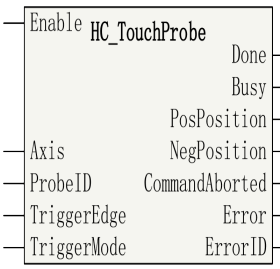
Introduction

The HC_TouchProbe instruction records the counter value based on the trigger signal.

HC_TouchProbe -- High-speed counter probe

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
HC_TouchProbe	High-speed counter probe		HC_TouchProbe(Enable := ???, Axis := ???, ProbeID := ???,

Instruction	Name	LD Expression	LiteST Expression
			TriggerEdge := , TriggerMode := , Done => , Busy => , PosPosition => , NegPosition => , CommandAborted => , Error => , ErrorID =>);

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	HC_TouchProbe: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name or axis ID, which specifies the local encoder axis to be operated	No	-	0 to 32,767	INT16
S2	ProbeID	Probe ID 0: Probe 1 1: Probe 2	No	0	0 to 1	INT16
S3	TriggerEdge	Trigger edge 1: Triggered on the rising edge of the external input X 2: Triggered on the falling edge of the external input X 3: Triggered on the rising and falling edges of the external input X	Yes	1	1 to 3	INT16
S4	Trigger Mode	Trigger mode 0: Single trigger 1: Continuous trigger	Yes	0	0 to 1	INT16
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3		Position latched on the rising edge, in Unit	Yes	0	-	

	PosPosition					REAL 32
D4	NegPosition	Position latched on the falling edge, in Unit	Yes	0	-	REAL 32
D5	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT16

NOTE

*1: For details, see 5.15.22 Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	√[1]	-	√	-	√	-	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	√[1]	-	√	-	√	-	-	-	-
D6	√[1]	-	√	-	√	-	-	-	-
D7	-	-	-	√	√	√	-	-	-

> **NOTE**
>
> [1]: The X element is not supported.

Function and Instruction Description

The HC_TouchProbe instruction can latch the counter axis position value when the external input trigger condition is met.

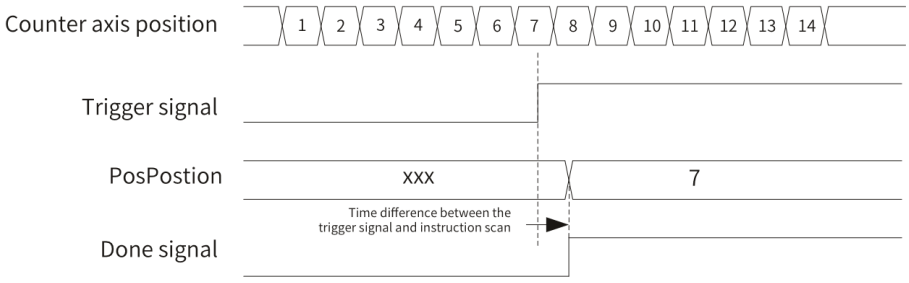
Each counter axis supports two probes. During use, you must enable the corresponding probe function, select the input terminal, and set the trigger condition on the counter parameter configuration interface. The input terminal can be set to any one of X1 to X7.

TriggerEdge specifies the probe trigger edge. The rising edge trigger position is latched in the output parameter PosPosition, and the falling edge trigger position is latched in the output parameter NegPosition.

Item	Value	Definition
TriggerEdge	1	Triggered on the rising edge of the external input X
	2	Triggered on the falling edge of the external input X
	3	Triggered on the rising and falling edges of the external input X

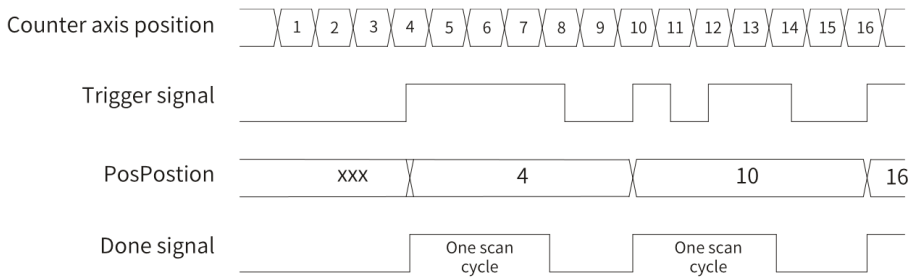
TriggerMode in the instruction can be set to the single trigger or continuous trigger mode.

- If the single trigger mode is used, when the function block instruction energy flow is active and the external input trigger condition is met, the counter axis position is latched once, and the Done signal is output. The counter axis position is latched in real time based on the trigger edge, which is not affected by program execution. During instruction execution, affected by the scan cycle, when the program scans and executes to the latched instruction, it updates the latched position to the output parameter of the instruction.



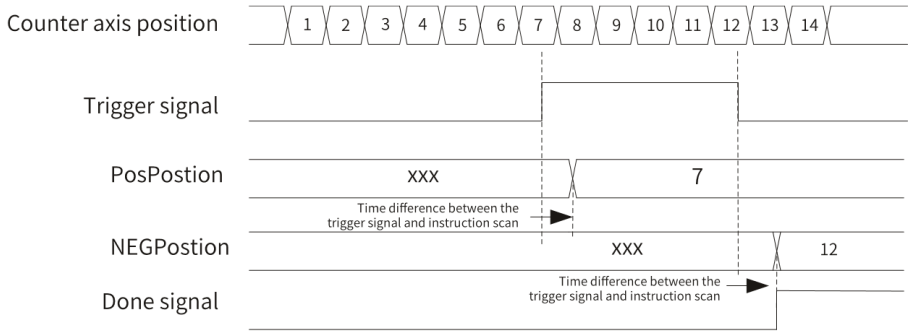
Single triggering on the rising edge

- If the continuous trigger mode is used, when the function block instruction energy flow is active and the external input trigger condition is met, the counter axis position is latched, and the Done signal that is active for one scan cycle is output. When the Done signal becomes OFF and the external input trigger condition is met, the counter axis position continues to be latched and the Done signal that is active for one scan cycle is output. During the scan cycle in which the Done signal is active, if the external input trigger condition is met, the counter axis position is not latched at this time.

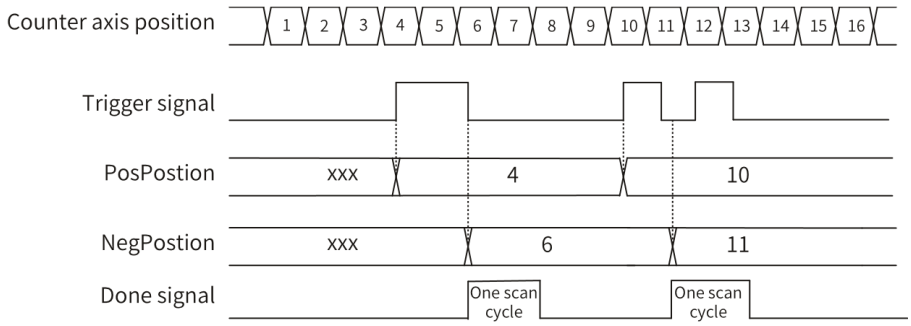


Continuous triggering on the rising edge

- When the dual-edge trigger mode is used, the Done signal is output after the instruction is triggered on both the rising and falling edges to complete the latch. In single trigger mode, the Done signal remains active until the instruction execution is completed. In continuous trigger mode, the Done signal is active for one scan cycle, during which the latch signal is not responded.



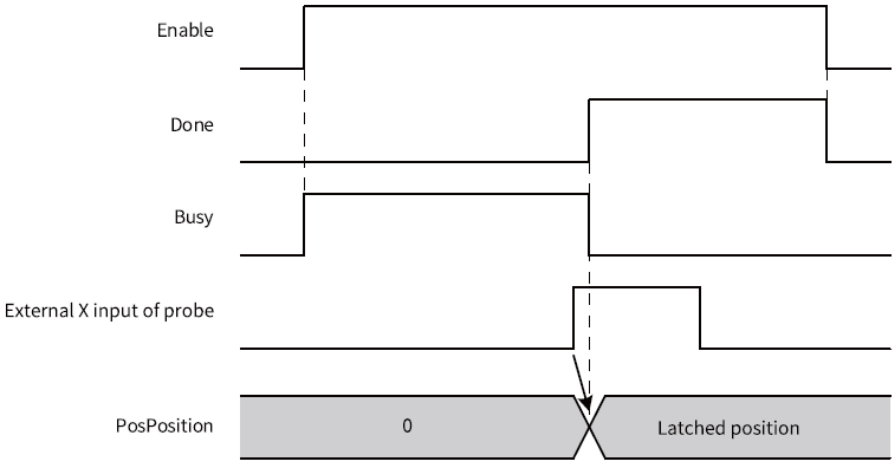
Single triggering on both the rising and falling edges



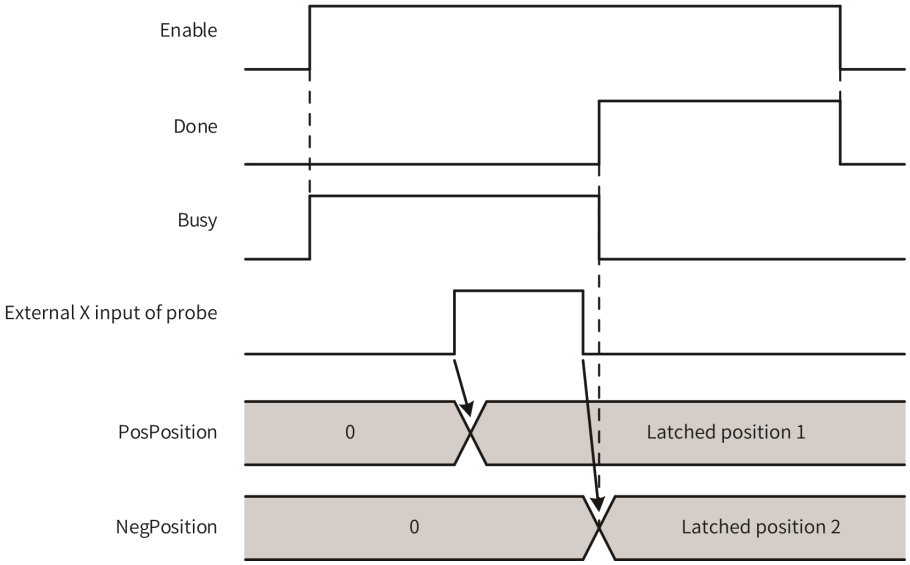
Continuous triggering on both the rising and falling edges

Timing Diagram

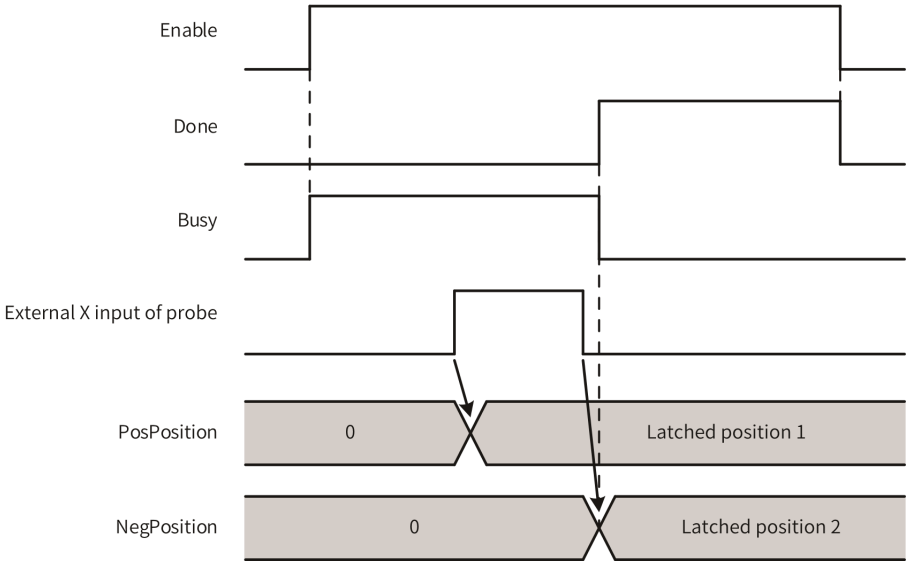
- TriggerEdge is set to the rising edge of the external input X (TriggerEdge = 1) and TriggerMode is set to the single trigger mode (TriggerMode = 0).



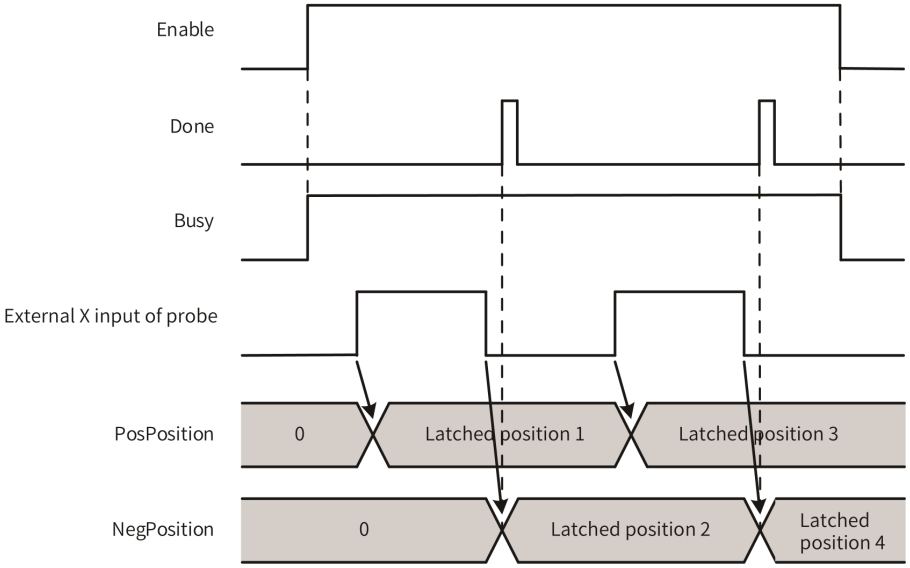
- TriggerEdge is set to the falling edge of the external input X (TriggerEdge = 2) and TriggerMode is set to the single trigger mode (TriggerMode = 0).



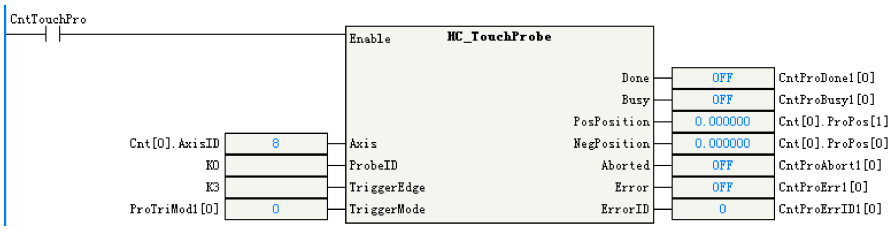
- TriggerEdge is set to the rising and falling edges of the external input X (TriggerEdge = 3) and TriggerMode is set to the single trigger mode (TriggerMode = 0).



- TriggerEdge is set to the rising and falling edges of the external input X (TriggerEdge = 3) and TriggerMode is set to the continuous trigger mode (TriggerMode = 1).



Instruction Example



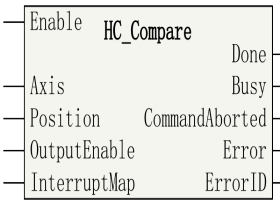
5.15.19 HC_Compare

Description

This instruction detects whether the counter count value reaches the specified value.

HC_Compare: High-speed counter comparison

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
HC_Compare	High-speed counter comparison		HC_Compare(Enable := ???, Axis := ???, Position := ???, OutputEnable := , InterruptMap := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	HC_Compare: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT
S2	Position	Comparison position (unit: Unit)	No	-	-	REAL
S3	OutputEnable	Hardware output enable 0: Disabled	Yes	0	0 to 1	INT

		1: Enabled				
S4	Interrupt Map	Interrupt generation and association when the comparand and count value match 0: No interrupt is generated. 1: Associate with comparison interrupt 1. ... 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Instruction Error Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	√	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	√[1]	-	√	-	√	-	-	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D3	√[1]	-	√	-	√	-	-	-	-
D4	√[1]	-	√	-	√	-	-	-	-
D5	-	-	-	√	√	√	-	-	-

NOTE

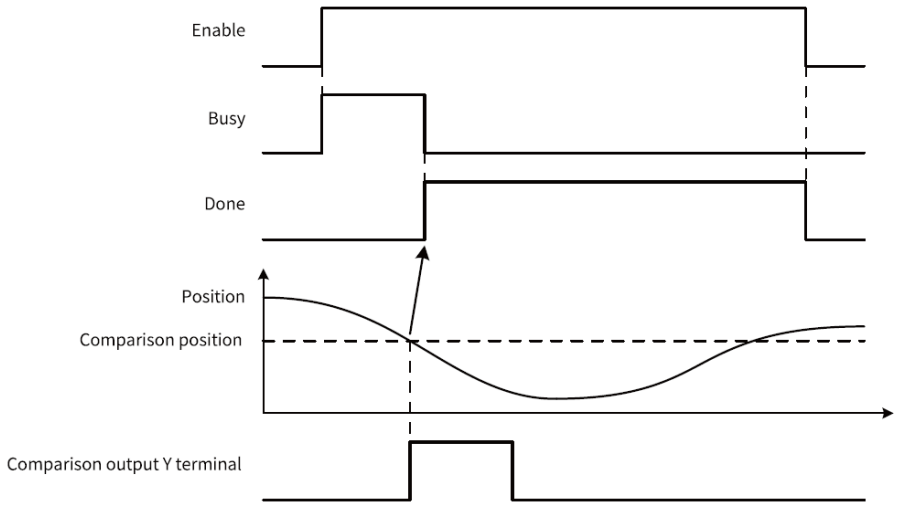
[1] The X element is not supported.

Function and Instruction Description

The HC_Compare instruction compares the counter axis position with a single position.

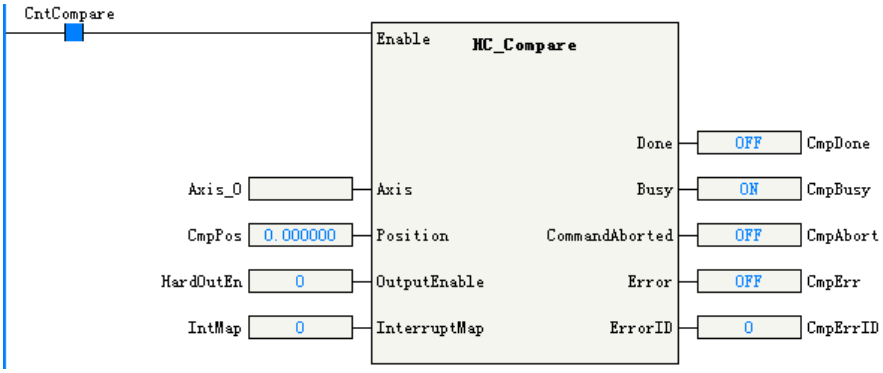
Sequence Diagram

The timing diagram of the instruction is as follows when the hardware output is enabled (OutputEnable = 1).



Example

The instruction compares the counter axis with a single position. When the instruction flow is active, the Done signal is output after the counter axis position reaches the comparison position.



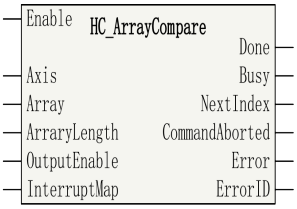
5.15.20 HC_ArrayCompare

Description

This instruction continuously detects whether the counter count value reaches the specified array sequence.

HC_ArrayCompare: High-speed counter array comparison

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
HC_ArrayCompare	High-speed counter array comparison		HC_ArrayCompare(Enable := ???, Axis := ???, Array := ???, ArrayLength := ???, OutputEnable := , InterruptMap := , Done => , Busy => , NextIndex => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	HC_ArrayCompare: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT

S2	Array	Comparison position array (unit: Unit)	No	-	-	REAL
S3	ArrayLength	Array length	No	-	0 to 100	INT
S4	Output Enable	Hardware output enable 0: Disabled 1: Enabled	Yes	0	0 to 1000	INT
S5	InterruptMap	Interrupt generation and association when the comparand and count value match 0: No interrupt is generated. 1: Associate with comparison interrupt 1. ... 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	NextIndex	Index of the next comparand	Yes	0	0 to 100	INT
D4	Aborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D5	Error	Error	Yes	OFF	ON OFF	BOOL
D6	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Instruction Error Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	√[1]	-	√	-	√	-	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	-	√	-	√	-	-	-	-
D5	√[1]	-	√	-	√	-	-	-	-
D6	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Function

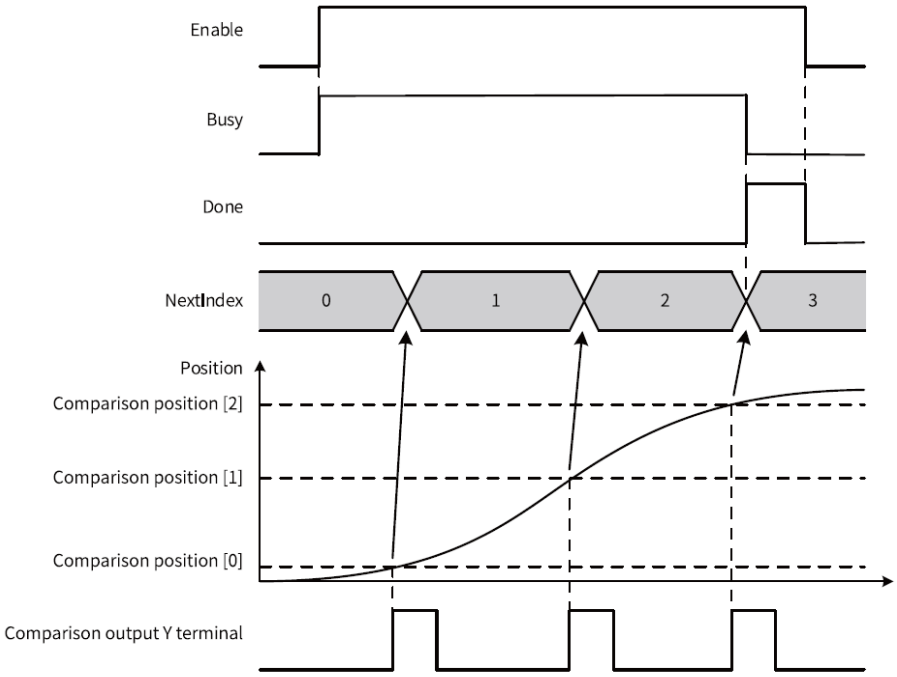
The HC_ArrayCompare instruction compares the counter axis position with multiple positions continuously.

When the instruction flow is active, the counter axis position is compared with the position value in array 0. If they are equal, the counter axis position is compared with the next position value in the array. ArrayLength in the instruction specifies the array length. After all the positions are compared, the Done signal is output.

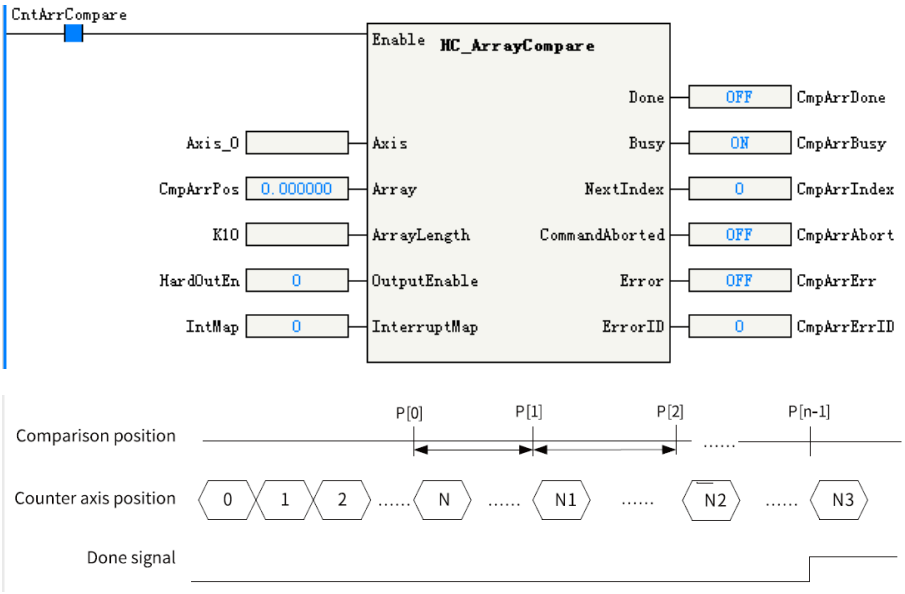
The output parameter NextIndex indicates the index of the next comparison point, that is, the number of completed comparison points.

Sequence Diagram

The timing diagram of the instruction is as follows when the hardware output is enabled (OutputEnable = 1) and three positions are compared (ArrayLength = 3).



Example



5.15.21 HC_StepCompare

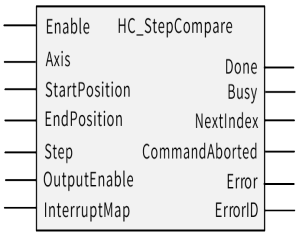
Description

This instruction continuously detects whether the counter count value reaches the continuous sequence with specified range and step.

HC_StepCompare: High-speed counter step comparison

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
HC_StepCompare	High-speed counter step comparison		HC_StepCompare(Enable := ???, Axis := ???,

Instruction	Name	LD Expression	LiteST Expression
			StartPosition := ???, EndPosition := ???, Step := ???, OutputEnable := , InterruptMap := , Done => , Busy => , NextIndex => , CommandAborted => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	HC_StepCompare: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT
S2	StartPosition	Start position (unit: Unit)	No	-	-	REAL
S3	EndPosition	End position (unit: Unit)	No	-	-	REAL
S4	Step	Step (unit: Unit)	No	-	Positive number	REAL
S5	OutputEnable	Hardware output enable 0: Disabled 1: Enabled	Yes	0	0 to 1	INT
S6	InterruptMap	Interrupt generation and association when the comparand and count value match 0: No interrupt is generated. 1: Associate with comparison interrupt 1. ... 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT

D1	Done	0: Not completed 1: Completed	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	FLASE	ON OFF	BOOL
D3	NextIndex	Index of the next comparand	Yes	0	0 to 100	INT
D4	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D5	Error	Error	Yes	OFF	ON OFF	BOOL
D6	ErrorID	Error code	Yes	0	*1	INT

NOTE

*1: For details, see Instruction Error Codes.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	√	-
S3	-	-	-	√	√	√	√	√	-
S4	-	-	-	√	√	√	√	√	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	√[1]	-	√	-	√	-	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	-	√	-	√	-	-	-	-
D5	√[1]	-	√	-	√	-	-	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D6	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Function

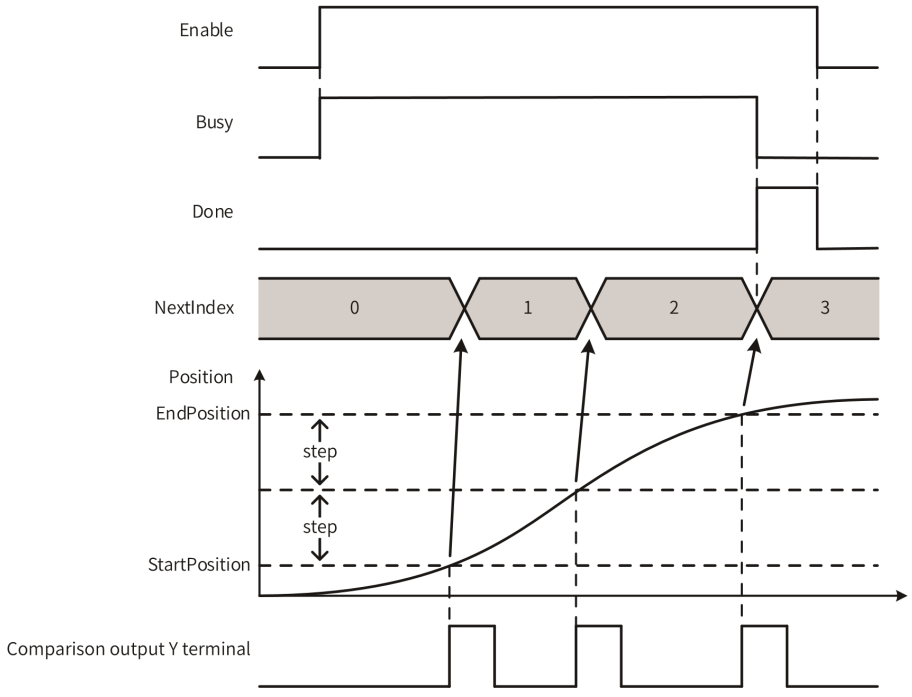
The HC_StepCompare instruction compares the counter axis position with consecutive positions with equal steps.

When the instruction flow is active, the counter axis position is compared with the position specified by StartPosition. When they are equal, the comparison position increases or decreases by a value specified by Step and then is compared with the counter axis position. After the last comparison position is compared, the Done signal is output.

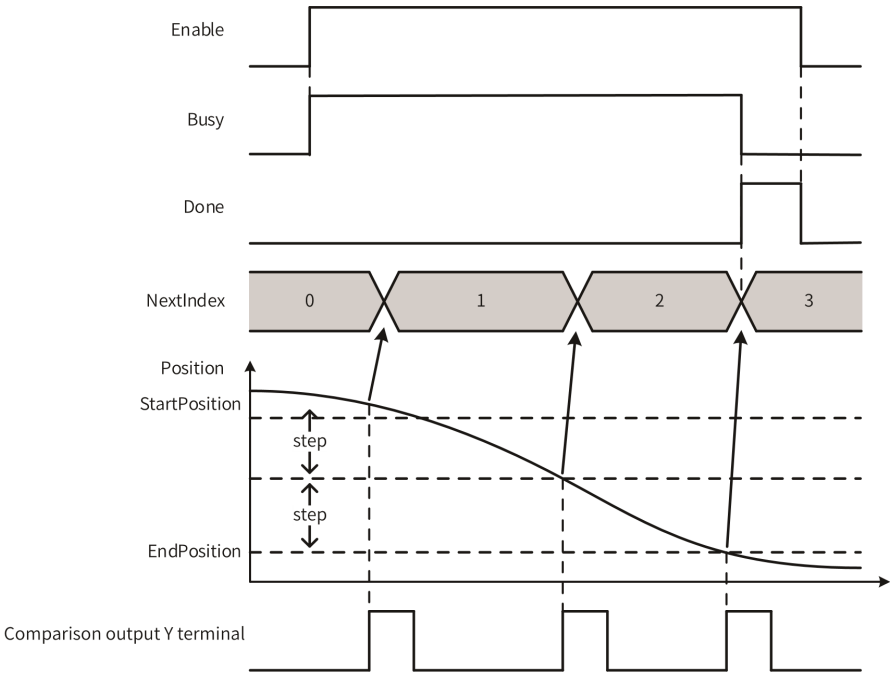
The output parameter NextIndex indicates the index of the next comparison point. The index starts from 0, that is, 0 indicates the first comparison point. Therefore, this index number is equal to the number of comparison points that have been compared.

Sequence Diagram

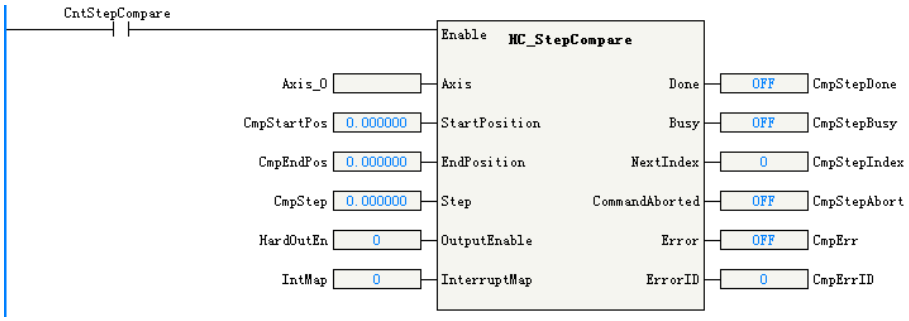
- The timing diagram of the instruction is as follows when the hardware output is enabled (OutputEnable = 1) and StartPosition is less than EndPosition.



- The timing diagram of the instruction is as follows when the hardware output is enabled (OutputEnable = 1) and StartPosition is greater than EndPosition.



Example



5.15.22 Error Codes

The following table lists the error codes of the high-speed counter function blocks.

Error code	Description
0	No error
100	The axis ID is invalid./It's not the local encoder axis.
101	The imaginary axis mode is not supported.
102	The ENC_SetUnit instruction cannot be enabled after encoder counting is enabled. Ensure that the instruction is enabled before the encoder is enabled, and that PlusePerCycle is set properly.
103	DisPerCycle is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
104	Numerator is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.

Error code	Description
105	Denotinator is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
106	Failed to set the gear ratio.
107	The ENC_SetLineRotationMode instruction is configured after encoder counting is enabled. Ensure that the instruction is enabled before the encoder is enabled, and that LineRotateMode is set properly.
108	PLimit is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
109	NLimit is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
110	PLimit and NLimit are set incorrectly in the instruction. Ensure that the parameter values are within the permissible range.
111	Rotation is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
113	The current parameters do not meet the conditions for enabling software limiting.
114	Failed to set the linear/rotary mode.
200	An invert parameter input error occurs.
201	A trigger mode parameter input error occurs.
202	A trigger edge parameter input error occurs. The trigger edge is invalid or the X input is not configured.
203	FirstPosition is set incorrectly in the instruction.
204	LastPosition is set incorrectly in the instruction.
205	Failed to execute the instruction because the parameters in the probe instruction are set incorrectly.
206	Failed to set the preset position due to a parameter exception.
300	A probe ID parameter input error occurs.
301	An output enable parameter input error occurs. The output enable signal is invalid or the Y output is not configured.
400	An interrupt mapping parameter input error occurs.
500	A preset position parameter input error occurs.

Error code	Description
501	A comparison position parameter input error occurs.
502	A start position parameter input error occurs.
503	An end position parameter input error occurs.
504	A step parameter input error occurs.
600	An array length parameter input error occurs or the number of positions for step comparison exceeds 100.
1000	Counting exceeded the lower limit.
1001	Counting exceeded the upper limit.

The following table lists the error codes of the bus encoder axis instructions.

Error code	Cause	Solution
9701	The encoder axis instruction failed to request the memory.	1. Check whether the PLC memory runs out. 2. Contact the manufacturer.
9702	1. The encoder axis type is incorrect. 2. The requested encoder axis does not exist. 3. The instruction is not supported in offline commissioning.	1. This instruction does not support the set axis type. Check whether the axis type setting is incorrect. 2. The instruction is not supported in offline commissioning.
9703	Failed to configure the axis.	Check whether the board software and the software tool match.
9704	Counter operation command is not configured in I/O mapping of the encoder axis.	Configure Counter operation command in I/O mapping of the encoder axis.
9705	Counter status is not configured in I/O mapping of the encoder axis.	Configure Counter status in I/O mapping of the encoder axis.
9706	Encoder present position is not configured in I/O mapping of the encoder axis.	Configure Encoder present position in I/O mapping of the encoder axis.
9707	Pulse rate is not configured in I/O mapping of the encoder axis.	Configure Pulse rate in I/O mapping of the encoder axis.
9708	The positive limit of the encoder axis is not greater than the negative limit.	Ensure that the positive limit of the encoder axis is greater than the negative limit.

Error code	Cause	Solution
9709	The positive limit of the encoder axis is greater than 2147483647 after being converted into the pulse unit.	Ensure that the positive limit of the encoder axis is less than or equal to 2147483647 after being converted into the pulse unit.
9710	The negative limit of the encoder axis is less than -2147483648 after being converted into the pulse unit.	Ensure that the negative limit of the encoder axis is greater than or equal to -2147483648 after being converted into the pulse unit.
9711	The revolution cycle of the encoder axis in ring mode is greater than 2147483647 after being converted into the pulse unit.	Ensure that the revolution cycle of the encoder axis in ring mode is less than or equal to 2147483647 after being converted into the pulse unit.
9712	The encoder axis is changed while the ENC_Counter instruction is still active.	Do not change the encoder axis while the ENC_Counter instruction is still active.
9713	GR10-2HCE module fault	Check the fault code object dictionary of the GR10-2HCE module and troubleshoot the fault according to the fault code.
9714	Failed to reset the encoder axis fault.	1. The current fault of the encoder axis does not support reset. 2. The encoder shaft enters the faulty state immediately after the fault is reset. Check the axis fault codes and slave fault codes to further determine the fault type.
9715	The ENC_Reset instruction is called when the encoder axis is not faulty.	Do not call the ENC_Reset instruction when the encoder axis is not faulty.
9716	The value of TriggerMode of the ENC_Preset instruction is out of range.	Ensure that the parameter value is within the allowable range.
9717	The value of Position of the ENC_Preset instruction is greater than 9999999.	Ensure that the value of Position of the ENC_Preset instruction is less than or equal to 9999999.
9718	Physical output command is not configured in I/O mapping of the encoder axis.	Configure Physical output command in I/O mapping of the encoder axis.
9719	The preset position or comparison output position of the encoder axis instruction is greater than the positive limit.	Ensure that the preset position or comparison output position of the encoder axis instruction is less than or equal to the positive limit.
9720	The preset position or comparison output position of the encoder axis instruction is less than the negative limit.	Ensure that the preset position or comparison output position of the encoder axis instruction is greater than or equal to the negative limit.
9721	The preset position or comparison output position of the encoder axis instruction is	Ensure that the preset position or comparison output position of the encoder axis instruction is

Error code	Cause	Solution
	greater than 2147483647 or less than –2147483648 after being converted into the pulse unit.	between -2147483648 and +2147483647 after being converted into the pulse unit.
9722	The preset position or comparison output position of the encoder axis instruction is greater than or equal to the revolution cycle in ring mode.	Ensure that the preset position or comparison output position of the encoder axis instruction is less than the revolution cycle in ring mode.
9723	The value of ProbeID of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9724	The value of TriggerEdge of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9725	The value of TerminalSource of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9726	The value of TriggerMode of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9727	The probe status word is not associated in I/O mapping of the encoder axis.	Ensure that the probe status word is associated in I/O mapping of the encoder axis.
9728	The probe feedback position is not associated in I/O mapping of the encoder axis.	Ensure that the probe feedback position is associated in I/O mapping of the encoder axis.
9729	The control word is not associated in I/O mapping of the encoder axis.	Ensure that the control word is associated in I/O mapping of the encoder axis.
9730	The probe window function of the encoder axis is enabled, but the start position of the window is not less than the end position.	Ensure that the start position of the probe window is less than the end position.
9731	The Xn0 terminal is not assigned with the probe function.	Assign the probe function to the Xn0 terminal.
9732	The Xn1 terminal is not assigned with the probe function.	Assign the probe function to the Xn1 terminal.
9733	The instruction is not supported by the local encoder axis.	Note that this instruction applies only to the bus encoder axis.
9734	The instruction is not supported by the bus encoder axis.	Note that this instruction applies only to the local encoder axis.
9742	Compare mode is not configured in I/O mapping of the encoder axis.	Configure Compare mode in I/O mapping of the encoder axis.

Error code	Cause	Solution
9743	Compare pulse/time is not configured in I/O mapping of the encoder axis.	Configure Compare pulse/time in I/O mapping of the encoder axis.
9744	Compare size/step is not configured in I/O mapping of the encoder axis.	Configure Compare size/step in I/O mapping of the encoder axis.
9745	Compare point value 1 is not configured in I/O mapping of the encoder axis.	Configure Compare point value 1 in I/O mapping of the encoder axis.
9746	Compare point value 2 is not configured in I/O mapping of the encoder axis.	Configure Compare point value 2 in I/O mapping of the encoder axis.
9747	Physical output status is not configured in I/O mapping of the encoder axis.	Configure Physical output status in I/O mapping of the encoder axis.
9748	Compare error code is not configured in I/O mapping of the encoder axis.	Configure Compare error code in I/O mapping of the encoder axis.
9749	Current compare number/position is not configured in I/O mapping of the encoder axis.	Configure Current compare number/position in I/O mapping of the encoder axis.
9750	Failed to obtain the start address of the array of the single-axis array comparison output instruction.	1. Check whether the PLC memory is sufficient. 2. Check whether the software tool and board software match. 3. Check whether the array of the instruction is out of bounds.
9751	Failed to obtain the start address of the axis group of the axis group array comparison output instruction.	1. Check whether the PLC memory is sufficient. 2. Check whether the software tool and board software match. 3. Check whether the array of the instruction is out of bounds.
9752	The bus encoder axis is not associated with any slave.	Associate the bus encoder axis with a slave.
9753	The x-axis and y-axis of the axis group array comparison instruction are not associated with the same slave.	Associate the x-axis and y-axis of the axis group comparison output instruction with the same slave.
9754	The x-axis of the axis group array comparison instruction is not associated with the first channel of the slave.	Associate the x-axis of the axis group comparison output instruction with the first channel of the slave.
9755		

Error code	Cause	Solution
	The y-axis of the axis group array comparison instruction is not associated with the second channel of the slave.	Associate the y-axis of the axis group comparison output instruction with the second channel of the slave.
9756	The Yn0 terminal is not assigned with the one-dimensional comparison output function.	Assign the one-dimensional comparison output function to the Yn0 output terminal corresponding to the channel.
9757	The absolute value of the start value of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9758	The absolute value of the end value of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9759	The absolute value of the step of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9760	The absolute value of Parameter of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9761	The value of Mode of the encoder axis comparison output instruction is out of range.	Ensure that the parameter value is within the allowable range.
9762	The time for time control of the encoder axis comparison output is out of range.	Ensure that the parameter value is within the allowable range.
9763	The step of the encoder axis step comparison output instruction is 0.	Set the step of the step comparison output instruction to a value other than 0.
9764	The start position of the step comparison output instruction of the encoder axis is equal to the end position.	Ensure that the start position of the step comparison output instruction is not equal to the end position.
9765	The start position of the step comparison output instruction of the encoder axis is greater than the end position, but the step is positive.	Set the step to a negative value.
9766	The start position of the step comparison output instruction of the encoder axis is less than the end position, but the step is negative.	Set the step to a positive value.

Error code	Cause	Solution
9767	The value of Size of the encoder axis array comparison output instruction is out of range.	Ensure that the parameter value is within the allowable range.
9768	The absolute value of the target position of the encoder axis array comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9769	The axis is performing one-dimensional comparison output and must not be aborted by a two-dimensional comparison output instruction.	Wait for the one-dimensional comparison output to complete or stop the one-dimensional comparison output before executing the two-dimensional comparison output instruction.
9770	The EtherCAT slave is disconnected during operation.	Check whether the EtherCAT slave is disconnected during operation.
9771	The bus encoder axis is in offline commissioning mode.	The bus encoder axis does not support the offline commissioning mode.
9772	The DI terminal is not assigned with the preset position function.	Assign the preset position function to the DI terminal before calling the preset position instruction.
9773	The value of Parameter in the comparison instruction is out of range when the pulse output mode is selected.	Do not set Parameter to 0 or a negative value when the pulse output mode is selected in the comparison instruction.
9774	The 2HCE module fails when the comparison output instruction is called.	1. Ensure that the input parameters are within the allowable range. 2. Check whether I/O mapping of the encoder axis is manually modified and whether it meets the I/O mapping configuration requirements of the comparison output instruction.
9775	The set position in ring mode is less than 0.	Set the position in ring mode to a value greater than or equal to 0.
9776	The Yn0 terminal is not assigned with the two-dimensional comparison output function.	Assign the two-dimensional comparison output function to the Yn0 output terminal corresponding to the channel.
9777	The axis is performing two-dimensional comparison output and cannot be aborted by a one-dimensional comparison output instruction.	Wait for the two-dimensional comparison output to complete or stop the two-dimensional comparison output before calling the one-dimensional comparison output instruction.

5.16 Timer Instructions

5.16.1 Parameters

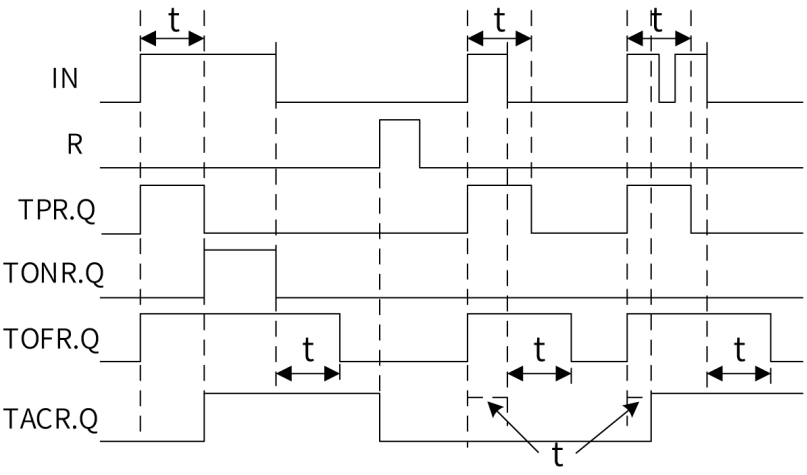
The PLC supports four types of timers: pulse timer (TPR), on-delay timer (TONR), off-delay timer (TOFR), and accumulating timer.

The time base of the timers is 1ms, and the timer count value and state are updated when the timer instruction is executed. The program supports a maximum of 4096 timer instructions. The instruction parameters of these four types of timers are the same, which are listed as follows:

Parameters are shown in the following table.

Name	Definitions	Data type	Description
IN	Instruction execution input	/	Start input
PT	Input variable	DINT	Delay
R	Input variable	BOOL	Reset input
Q	Output variable	BOOL	Timer output
ET	Output variable	DINT	Current timing time

Timer timing



5.16.2 List of Instructions

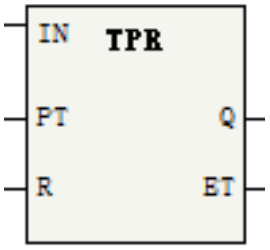
The following table lists the timer instructions.

Category	Instruction	Function
Timer instruction	TPR	Pulse timer
	TONR	Connection delay timer
	TOFR	Off delay timer

Category	Instruction	Function
	TACR	Accumulation timer

5.16.3 TPR

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TPR	Pulse timer		TPR(IN := ???, PT := ???, R := , Q => , ET =>);

The following table describes the instruction format.

16-bit Instruction	-			
32-bit Instruction	TPR: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	PT	Preset timing duration, in ms	-	DINT
S2	R	Reset*1	-	BOOL
D1	Q	Output result*1	-	BOOL
D2	ET	Elapsed time*1, in ms	-	DINT

NOTE

[1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

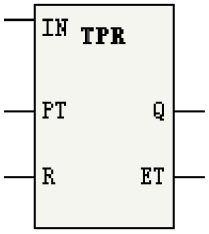
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	√	√	-	-	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

NOTE

[1]: The X element is not supported.

Function and Parameter Description



When the IN input flow of the timer instruction changes from OFF to ON, the timer starts timing and the output Q turns ON. At this time, no matter how the IN input flow changes, Q remains ON for the time period specified by PT. When the timing duration reaches the time period specified by PT, Q changes to OFF.

During timing of the timer, ET outputs the current timing duration. After the timing duration reaches the value specified by PT, if the IN input flow is ON, the ET value is retained. If the IN input flow is OFF, the ET value becomes 0.

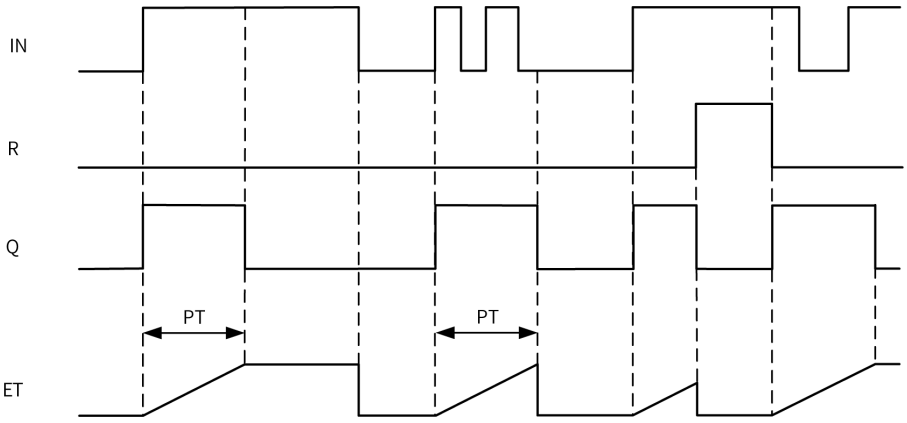
During timing, if the reset input R changes from OFF to ON, the timing duration of the TPR timer is reset to 0, and the output Q turns OFF. After the reset input R turns OFF, if the IN input flow is active, the timer resumes timing.

Parameter description

PT ranges from 0 ms to 2,147,483,647 ms (about 24 days). If the value of PT is less than or equal to 0, it is considered 0.

Timing Diagram

The following figure shows the timing diagram of the parameters IN, R, Q, and ET.

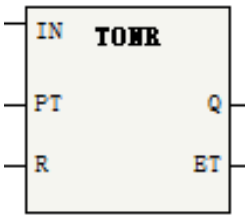


NOTE

The output parameter Q is updated in the PLC main task. Affected by the PLC scan cycle, the instruction may not immediately output the result when the time specified by PT elapses. The output may be delayed to certain extent, with a maximum delay of one PLC scan cycle.

5.16.4 TONR

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TONR	Connection delay timer		TONR(IN := ???, PT := ???, R := , Q => , ET =>);

The following table describes the instruction format.

16-bit Instruction	-			
32-bit Instruction	TONR: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	PT	Preset timing duration, in ms	-	DINT
S2	R	Reset*1	-	BOOL
D1	Q	Output result*1	-	BOOL
D2	ET	Elapsed time*1, in ms	-	DINT

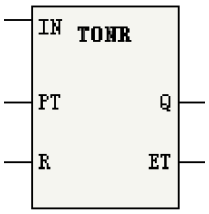
> **NOTE**
>
> [1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	√	√	-	-	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

> **NOTE**
>
> [1]: The X element is not supported.

Function and Instruction Description



When the IN input flow of the timer instruction changes from OFF to ON, the timer starts timing and the output Q turns OFF. During the period when the IN input flow remains ON, the running time of the timer is the time specified by PT. After the timing duration reaches the time period specified by PT, Q turns ON. During the timing process or after timing is completed, when the IN input flow changes to OFF, timing ends and Q turns OFF. During the period when the IN input flow remains OFF, Q remains OFF.

When the IN input flow is ON, ET outputs the current timing duration during timing of the timer, and the ET value is retained after the timing duration reaches the value specified by PT. When the IN input flow is OFF, the ET value becomes 0.

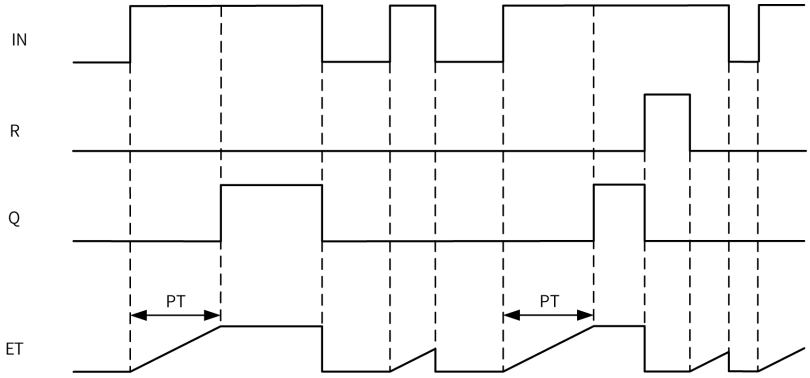
During timing, if the reset input R changes from OFF to ON, the timing duration of the TONR timer is reset to 0, and the output Q turns OFF. After the reset input R turns OFF, to resume timer timing, you need to set the IN input flow to ON again.

Parameter description

PT ranges from 0 ms to 2,147,483,647 ms (about 24 days). If the value of PT is less than or equal to 0, it is considered 0.

Timing Diagram

The following figure shows the timing diagram of the parameters IN, R, Q, and ET.

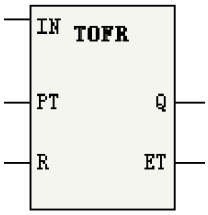


NOTE

The output parameter Q is updated in the PLC main task. Affected by the PLC scan cycle, the instruction may not immediately output the result when the time specified by PT elapses. The output may be delayed to certain extent, with a maximum delay of one PLC scan cycle.

5.16.5 TOFR

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TOFR	Off delay timer		TOFR(IN := ???, PT := ???, R := , Q => , ET =>);

The following table describes the instruction format.

16-bit Instruction	-			
32-bit Instruction	TOFR: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	PT	Preset timing duration, in ms	-	DINT
S2	R	Reset*1	-	BOOL
D1	Q	Output result*1	-	BOOL
D2	ET	Elapsed time*1, in ms	-	DINT

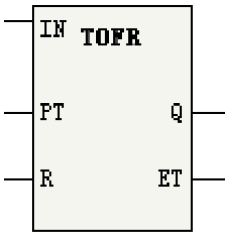
> **NOTE**
>
> [1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	√	√	-	-	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

> **NOTE**
>
> [1]: The X element is not supported.

Function and Instruction Description



When the IN input of the timer instruction changes from OFF to ON, the timer starts timing and the output Q turns ON. When the IN input flow changes from ON to OFF, during the period when the IN input flow remains OFF, the running time of the timer is the time specified by PT. After the timing duration reaches the time period specified by PT, Q turns OFF. During the period when the IN input flow remains OFF, Q remains OFF.

When the IN input flow is ON, the ET output becomes 0. When the IN input changes from ON to OFF, ET outputs the current timing duration during timing of the timer, and the ET value is retained after the timing duration reaches the value specified by PT.

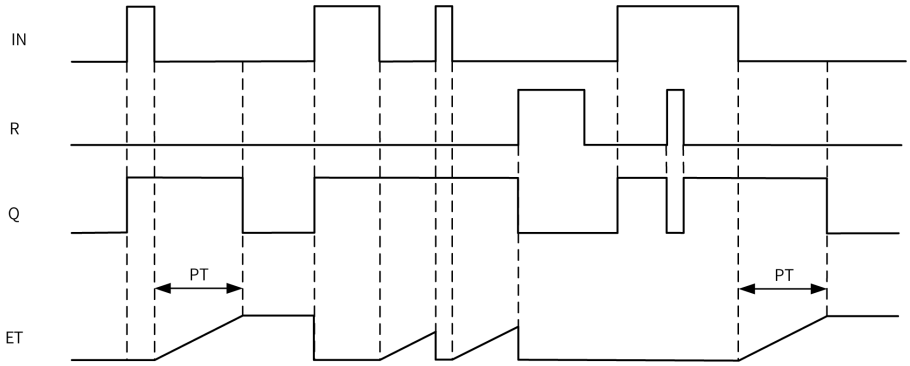
When the IN input flow is ON, if the reset input R changes from OFF to ON, the output Q turns OFF. If R resumes OFF, the output Q resumes ON. When the IN input flow changes from ON to OFF, if the reset input R changes from OFF to ON during the timing process or after timing is completed, the output Q turns OFF, and ET is reset to 0. After the reset input R turns OFF, to resume timing, you need to set the IN input flow to OFF again.

Parameter description

PT ranges from 0 ms to 2,147,483,647 ms (about 24 days). If the value of PT is less than or equal to 0, it is considered 0.

Timing Diagram

The following figure shows the timing diagram of the parameters IN, R, Q, and ET.

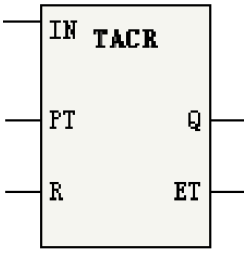


NOTE

The output parameter Q is updated in the PLC main task. Affected by the PLC scan cycle, the instruction may not immediately output the result when the time specified by PT elapses. The output may be delayed to certain extent, with a maximum delay of one PLC scan cycle.

5.16.6 TACR

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TACR	Accumulation timer		TACR(IN := ???, PT := ???, R := , Q => , ET =>);

The following table describes the instruction format.

16-bit Instruction	-			
32-bit Instruction	TACR: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	PT	Preset timing duration, in ms	-	DINT
S2	R	Reset*1	-	BOOL
D1	Q	Output result*1	-	BOOL

D2	ET	Elapsed time*1, in ms	-	DINT

NOTE

[1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

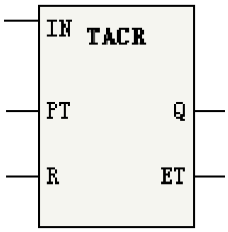
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	√	√	-	-	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

NOTE

[1]: The X element is not supported.

Function and Instruction Description



When the IN input flow of the timer instruction is ON, if the timer value has not reached the time period specified by PT, the timer continues to count, and the output Q is OFF. When the timing duration reaches the time period specified by PT, Q turns ON. During the timing process, if IN changes from ON to OFF, the timing duration is retained. When IN turns ON again, the timer starts counting from the current retained value. After the time specified by PT is reached, Q becomes ON.

When the IN input flow is ON, ET outputs the current timing value. After the timing duration reaches the time period specified by PT, the ET value is retained. When the IN input flow turns OFF, ET remains unchanged.

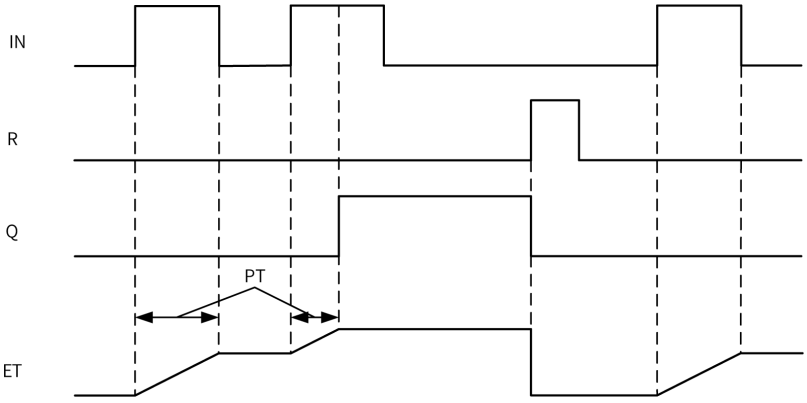
During the timing process or after timing is completed, if the reset input R changes from OFF to ON, the output Q turns OFF, and ET is reset to 0. After the reset input R turns OFF, to resume timing, you need to set the IN input flow to OFF again.

Parameter description

PT ranges from 0 ms to 2,147,483,647 ms (about 24 days). If the value of PT is less than or equal to 0, it is considered 0.

Timing Diagram

The following figure shows the timing diagram of the parameters IN, R, Q, and ET.



NOTE

The output parameter Q is updated in the PLC main task. Affected by the PLC scan cycle, the instruction may not immediately output the result when the time specified by PT elapses. The output may be delayed to certain extent, with a maximum delay of one PLC scan cycle.

5.17 Pointer Instructions

5.17.1 List of Instructions

The following table lists the pointer instructions.

Category	Instruction	Function
Pointer instruction	PTGET	Pointer variable assignment
	PTINC	Pointer variable address incremented by 1
	PTDEC	Pointer variable address decremented by 1
	PTADD	Pointer variable address addition
	PTSUB	Pointer variable address subtraction
	PTSET	Pointer variable assignment
	PTMOV	Pointer variable mutual assignment
	PTLD>	Pointer variable contact comparison greater than
	PTLD>=	Pointer variable contact comparison greater than or equal to

Category	Instruction	Function
	PTLD<=	Pointer variable contact comparison less than or equal to
	PTLD=	Pointer variable contact comparison equal to
	PTLD<>	Pointer variable contact comparison not equal to
	PTAND>	Pointer variable AND contact comparison greater than
	PTAND>=	Pointer variable AND contact comparison greater than or equal to
	PTAND<	Pointer variable AND contact comparison less than
	PTAND<=	Pointer variable AND contact comparison less than or equal to
	PTAND=	Pointer variable AND contact comparison equal to
	PTAND<>	Pointer variable AND contact comparison not equal to
	PTOR>	Pointer variable OR contact comparison greater than
	PTOR>=	Pointer variable OR contact comparison greater than or equal to
	PTOR<	Pointer variable OR contact comparison less than
	PTOR<=	Pointer variable OR contact comparison less than or equal to
	PTOR=	Pointer variable OR contact comparison equal to
	PTOR<>	Pointer variable OR contact comparison not equal to

5.17.2 PTGET

PTGET -- Pointer variable assignment

16-bit Instruction	PTGET (bit): Continuous execution/PTGETP: Pulse execution			
32-bit Instruction	-			
16-bit Instruction	PTGET (word): Continuous execution/PTGETP: Pulse execution			
32-bit Instruction	PTGET (dword): Continuous execution/PTGETP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Pointer variable	Start address of the target	-	POINTER
S2	Target variable	Start address of the target pointed to by the pointer variable	-	BOOL, INT, DINT, and REAL

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√[1]	-	-	-
S2	√	√	√	√	√	-	-	-	-

NOTE

[1]: Only the pointer variable POINTER is supported.

Function and Instruction Description

- The PTGET instruction obtains the address of a bit, word, or dword element or variable.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

Point the pointer variable PT5 to the D10 element.



5.17.3 PTINC

PTINC -- Pointer variable address incremented by 1

16-bit Instruction	-			
32-bit Instruction	PTINC: Continuous execution/PTINCP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Pointer variable	Pointer variable	-	POINTER

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	-	-	√[1]	-	-	-

NOTE

[1]: Only the pointer variable POINTER is supported.

Function and Instruction Description

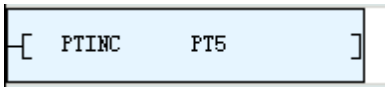
- The PTINC instruction points to the next element of the same type as the current variable based on the unit of the variable pointed to by the pointer variable POINTER. For example, the pointer points to the next bit element if the current variable is a bit element. The pointer points to the next word variable if the current variable is a word variable. The pointer points to the next dword variable if the current variable is a dword variable.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

1. Use the PT5 instruction to point to the D10 element.



2. PT5 points to the next element, that is, D11.



5.17.4 PTDEC

PTDEC -- Pointer variable address decremented by 1

16-bit Instruction	-			
32-bit Instruction	PTDEC: Continuous execution/PTDECP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Pointer variable	Pointer variable	-	POINTER

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	-	-	√[1]	-	-	-

NOTE

[1]: Only the pointer variable POINTER is supported.

Function and Instruction Description

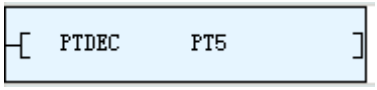
- The PTDEC instruction points to the previous element of the same type as the current variable based on the unit of the variable pointed to by the pointer variable POINTER. For example, the pointer points to the previous bit element if the current variable is a bit element. The pointer points to the previous word variable if the current variable is a word variable. The pointer points to the previous dword variable if the current variable is a dword variable.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

1. Use the PT5 instruction to point to the D10 element.



2. PT5 points to the previous element, that is, D9.



5.17.5 PTADD

PTADD -- Pointer variable address addition

16-bit Instruction	PTADD: Continuous execution/PTADDP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Source pointer	Source pointer	-	POINTER
S2	Offset address	Offset address	0 to 32,767	INT
D	Target pointer	Target pointer	-	POINTER

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√[1]	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S2	-	-	-	√	√	-	√	-	-
D	-	-	-	-	-	√[1]	-	-	-

NOTE

[1]: Only the pointer variable POINTER is supported.

Function and Instruction Description

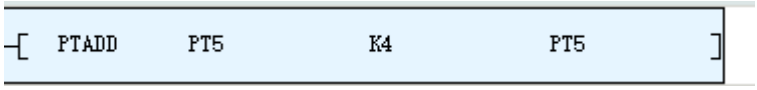
- The PTADD instruction points to the next *n*th element of the current variable based on the unit of the variable pointed to by the pointer variable POINTER. For example, the pointer points to the next *n*th bit element if the current variable is a bit element. The pointer points to the next *n*th word variable if the current variable is a word variable. The pointer points to the next *n*th dword variable if the current variable is a dword variable.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

1. Use the PT5 instruction to point to the D10 element.



2. When the pointer PT5 points to the D10 element, after this instruction is executed, PT5 points to the 4th element next to D10, that is, D14.



3. When the pointer PT5 points to the D10 element, after this instruction is executed, PT6 points to the 5th element next to the position pointed to by PT5, that is, D15, but PT5 still points to D10.



5.17.6 PTSUB

PTSUB -- Pointer variable address subtraction

16-bit Instruction	PTSUB: Continuous execution/PTSUBP: Pulse execution
32-bit Instruction	-

Operand	Name	Description	Range	Data Type
S1	Source pointer	Source pointer	-	POINTER
S2	Offset address	Offset address	0 to 32,767	INT
D	Target pointer	Target pointer	-	POINTER

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√[1]	-	-	-
S2	-	-	-	√	√	-	√	-	-
D	-	-	-	-	-	√[1]	-	-	-

NOTE

[1]: Only the pointer variable POINTER is supported.

Function and Instruction Description

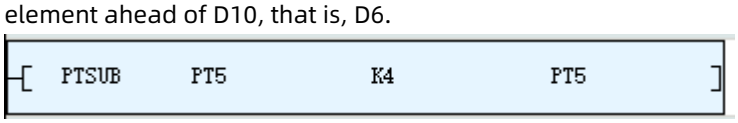
- The PTSUB instruction points to the *n*th element before the current variable based on the unit of the variable pointed to by the pointer variable POINTER. For example, the pointer points to the *n*th bit element ahead if the current variable is a bit element. The pointer points to the *n*th word variable ahead if the current variable is a word variable. The pointer points to the *n*th dword variable ahead if the current variable is a dword variable.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

1. Use the PT5 instruction to point to the D10 element.



2. When the pointer PT5 points to the D10 element, after this instruction is executed, PT5 points to the 4th element ahead of D10, that is, D6.



3. When the pointer PT5 points to the D10 element, after this instruction is executed, PT6 points to the 5th element ahead of the position pointed to by PT5, that is, D5, but PT5 still points to D10.



5.17.7 PTSET

The PTSET instruction assigns a value to a pointer variable and points to the target variable with specified variable length.

PTSET -- Assign value to pointer variable

16-bit Instruction	-			
32-bit Instruction	PTSET: Continuous execution/PTSETP: Pulse execution, 13 steps			
Operand	Name	Description	Range	Data Type
S1	Pointer element	-	-	POINTER
S2	Target variable	Start address of the target pointed to by the pointer variable	-	BOOL, INT, DINT, and REAL
S3	Variable length	-	-	DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√[1]	-	-	-
S2	√	√	√	√	√	-	-	-	-
S3	-	-	-	√	√	-	√	-	-

NOTE

[1]: Only the pointer variable POINTER is supported.

Function and Instruction Description

CAUTION

The PTSET instruction is a higher-order application and must be used with caution.

This instruction defines pointer variables of no specified type in the unit of bit. It can point to various basic types, arrays, and structures.

This instruction can be used as transit for forced type conversion.

Instruction Example

Pointers are used to assign the structure array. The pointer ptx points to the first element of the first structure of the structure array to assign values to the first element of each structure. The length of the variable is the length of each structure, that is, 80 bits.

str_arr	Stru[3]	...	Not retained			nBitLen:240
str_arr[0]	Stru	...				
i16d	INT	0				
bd	BOOL	OFF				
f32_d	REAL	0.000000				
b2d	BOOL	OFF				
str_arr[1]	Stru	...				
i16d	INT	0				
bd	BOOL	OFF				
f32_d	REAL	0.000000				
b2d	BOOL	OFF				
str_arr[2]	Stru	...				
i16d	INT	0				
bd	BOOL	OFF				
f32_d	REAL	0.000000				
b2d	BOOL	OFF				
ptx	POINTER	NULL	Not retained			nBitLen:32
axis	INT	0	Not retained			nBitLen:16

```
M0 ─┤↑├─[ PTSET   ptx      str_arr    K80  ]
                [ MOV     K1       axis       ]
                [ RST     M0       ]
M1 ─┤↑├─[ MOV     axis     *ptx       ]
                [ PTINC   ptx      ]
                [ INC     axis     ]
                [ PT>     ptx      str_arr[2].i16d ]─[ SET   M0   ]
```

After execution, the first element of each structure becomes 1, 2, and 3 respectively.

The following is an example of forced type conversion.

```
M0 ─┤↑├─[ PTSET   pty      i16_arr6   K16  ]
                [ DMOV    *pty     i32_arr1[0] ]
```

The execution result is equivalent to combining two i16_arr6 into a 32-bit number:

- The 16 bits of i16_arr6[0] is converted into the low-order 16 bits of i32arr1[0].
- The 16 bits of i16_arr6[1] is converted into the high-order 16 bits of i32arr1[0].

5.17.8 PTMOV

PTMOV -- Pointer variable mutual assignment

16-bit Instruction	-			
32-bit Instruction	PTMOV: Continuous execution/PTMOVP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source pointer	Source pointer	-	POINTER
D	Target pointer	Target pointer	-	POINTER

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	√[1]	-	-	-
D	-	-	-	-	-	√[1]	-	-	-

NOTE

[1]: Only the pointer variable POINTER is supported.

Function and Instruction Description

- The PTMOV instruction backs up the address of the pointer variable. It makes two pointer variables point to the same address.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

1. Use the PT5 instruction to point to the D10 element.



2. The pointer PT6 points to the position pointed to by PT5, that is, D10.



5.17.9 PT#

PT# -- Pointer variable contact comparison

Pointer variable contact comparison instructions include the PTLD, PTAND, and PTOR instructions, and # indicates >, >=, <, <=, =, or <>.

16-bit Instruction	-
32-bit Instruction	PTLD>: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTLD>=: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTLD<: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTLD<=: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTLD=: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTLD<>: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTAND>: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTAND>=: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTAND<: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTAND<=: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTAND=: Continuous execution

16-bit Instruction	-
32-bit Instruction	PTAND<>: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTOR>: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTOR>=: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTOR<: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTOR<=: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTOR=: Continuous execution
16-bit Instruction	-
32-bit Instruction	PTOR<>: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Current pointer	Current pointer	-	POINTER
S2	Comparison object	Comparison object	-	BOOL, INT, DINT, and REAL

NOTE

For the PTLD*, PTAND*, and PTOR* instructions, the input is PT*, and the corresponding instructions are automatically generated at the background.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√[1]	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S2	√	√	√	√	√	-	-	-	-

NOTE

[1]: Only the pointer variable POINTER is supported.

Function and Instruction Description

- The PT# instruction compares the address of the element pointed to by the pointer with the address of the comparison object.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.

Instruction Example

1. Use the PT5 instruction to point to the D10 element.



2. PT> PT5 D5: The output flow is ON. PT> PT5 D20: The output flow is OFF.

5.18 Communication Protocol Instructions

5.18.1 List of Instructions

The following table lists the communication protocol instructions.

Category	Communication Protocol	Instruction	Function	LD
Communication protocol instructions	Serial port free protocol	SerialSR	Serial port free protocol transmission and reception	LD
		SerialSend	Serial port free protocol transmission	LD and LiteST
		SerialRcv	Serial port free protocol reception	LD and LiteST

Category	Communication Protocol	Instruction	Function	LD
	Modbus protocol	MB_Master	Transmission and reception through the serial Modbus protocol	LD and LiteST
		MB_Client	Transmission and reception through the Modbus TCP protocol	LD and LiteST
	NTP	SysHC_NTPClient	NTP time synchronization	LD
	TCP/IP free protocol	TCP_Listen	TCP listening	LD
		TCP_Accept	Accept TCP connection request	LD
		TCP_Connect	Initiate TCP connection request	LD
		TCP_Close	Close TCP connection	LD
		TCP_Send	Send TCP data	LD
		TCP_Receive	Receive TCP data	LD
	UDP/IP free protocol	UDP_Bind	Bind UDP socket	LD
		UDP_Send	Send UDP data	LD
		UDP_Receive	Receive UDP data	LD
		UDP_JoinGroup	Add UDP port to multicast group	LD
	SFTP client	SFTP_GetFile	Obtain a file	LD and LiteST
		SFTP_PutFile	Upload a file	LD and LiteST
		SFTP_GetFileList	Obtain a file list	LD and LiteST
		SFTP_CreateDir	Create a directory	LD and LiteST
		SFTP_Delete	Delete a file or directory	LD and LiteST

Category	Communication Protocol	Instruction	Function	LD
		SFTP_Move	Move a file or directory	LD and LiteST
		SFTP_Rename File	Rename a file or directory	LD and LiteST
	EtherCAT protocol	ETC_ReadParameter_CoE	Read SDO parameters of EtherCAT slave	LD and LiteST
		ETC_WriteParameter_CoE	Write SDO parameters of EtherCAT slave	LD and LiteST
		ETC_RestartMaster	Restart EtherCAT master	LD and LiteST
	Ethernet/IP protocol	EIP_Generic_Service	Call Generic service	LD
		EIP_Get_Attributes_All	Call Get_Attributes_All service	LD
		EIP_Get_Attribute_Single	Call Get_Attribute_Single service	LD
		EIP_Set_Attributes_All	Call Set_Attributes_All service	LD
		EIP_Set_Attribute_Single	Call Set_Attribute_Single service	LD
		EIP_Apply_Attributes	Call Apply_Attributes service	LD
		EIP_NOP	Call NOP service	LD
		EIP_Reset	Call Reset service	LD
		EIP_Start	Call Start service	LD
		EIP_Stop	Call Stop service	LD
		EIP_OpenWithDataSize	Create EIP connection	LD

Category	Communication Protocol	Instruction	Function	LD
		EIP_Close	Close EIP connection	LD
		EIP_Read	Read CIP class 3	LD
		EIP_Write	Write CIP class 3	LD
	OPC UA protocol	OPCUA_Server_Shutdown	Shut down OPC UA server	LD and LiteST
	CAN free protocol	CANBus_DevRxMsg	Receive CAN free protocol data	LD and LiteST
		CANBus_DevTxMsg	Send CAN free protocol data	LD and LiteST

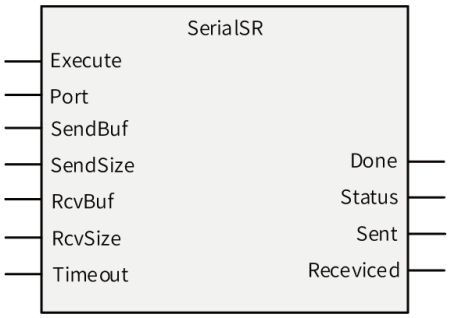
5.18.2 SerialSR

Introduction

SerialSR -- Serial port free protocol transmission and reception and free protocol cancellation

This instruction is used to implement free protocol communication through the serial port.

Graphic Block



The following table describes the instruction format.

16-bit Instruction	SerialSR: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Port	Port number	-	INT
S2	SendBuf	TX buffer	-	BYTE[] and INT[]

S3	SendSize	Number of bytes to transmit	0 to 256	INT
S4	RcvBuf	RX buffer	-	BYTE[] and INT[]
S5	RcvSize	Number of bytes to receive	0 to 256	INT
S6	Timeout	Receiving timeout, in ms[1]	-	INT
D1	Done	Completion flag[1]	-	BOOL
D2	Status	Instruction operation state[1]	-	INT
D3	Sent	Size of transmitted data[1]	-	INT
D4	Received	Size of received data[1]	-	INT

NOTE

[1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	-	-	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	√	√	-	-
D1	√[2]	√	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D3	-	-	-	√	√	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

NOTE

[2]: The X element is not supported.

Function and Instruction Description

- Function description
  The SerialSR instruction implements data transmission and reception through free protocols. After the instruction is triggered, data of the specified length is sent through the specified port. After the transmission is completed, data of the specified length is received. The corresponding output is updated during the transmission and reception process.

  When the SerialSR instruction is in the receive waiting state, you can set the system variable _SerialSR.abort to a non-zero value to terminate the current receiving process. The parameter modification takes effect immediately.

  You can use the system variables _SerialSR.startchar_en and _SerialSR.startchar[4] to set the receive start character function of the free protocol. The parameter modification takes effect next time data receiving is triggered. When _SerialSR.startchar_en is set to a value that falls within the range of 1 to 4 and equals to or smaller than the number of bytes to receive, the receive start character function is enabled. The start character length is specified by _SerialSR.startchar_en, and the start character content is specified by _SerialSR.startchar[4].

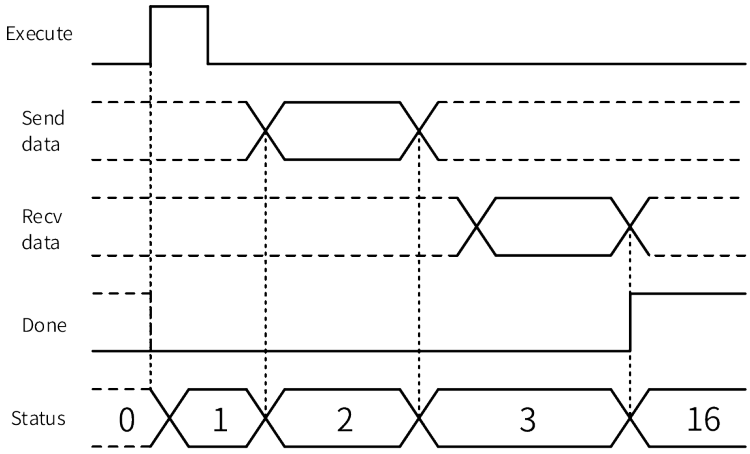
  You can use the system variables _SerialSR.endchar_en and _SerialSR.endchar[4] to set the receive end character function of the free protocol. The parameter modification takes effect next time data receiving is triggered. When _SerialSR.endchar_en is set to a value that falls within the range of 1 to 4, the receive end character function is enabled. The end character length is specified by _SerialSR.endchar_en, and the end character content is specified by _SerialSR.endchar[4].

  You can use the system variables _SerialSR.Bytetimeout_en and _SerialSR.Bytetimeout to set the receive byte timeout function of the free protocol. The parameter modification takes effect next time data receiving is triggered. When _SerialSR.Bytetimeout_en is set to ON, the receive byte timeout function is enabled. The byte timer is specified by _SerialSR.Bytetimeout, and the minimum value is 1, in ms*1. After the byte timeout function is enabled and takes effect, the timer starts when the start byte or frame start character is received. If the idle time between received bytes is greater than the set time, the current receiving process is terminated and the Done signal is set.

  Each serial port corresponds to a separate system variable. _SerialSR corresponds to COM0, and _SerialSR1 to _SerialSR15 correspond to COM1 to COM15 respectively.

- Parameter description
  - S1: Port number (The port number corresponds to the serial port. You need to set it to the serial port number selected for communication.)
  - S2: TX buffer. It is recommended that you set the size of this buffer to a value greater than 128 words or 256 bytes.
  - S3: Number of bytes to transmit. The data length ranges from 1 byte to 256 bytes. When this parameter is set to 0, no data is sent.
  - S4: RX buffer. It is recommended that you set the size of this buffer to a value greater than 128 words or 256 bytes.
  - S5: Number of bytes to receive. The data length ranges from 1 byte to 256 bytes. When this parameter is set to 0, no data is received.
  - S6: Timeout period. If the specified timeout period does not fall between 20 and 30000, in ms[1], it is automatically adjusted to the allowable range. (If the value is less than 20 ms, it is automatically changed to 20 ms. If the value is greater than 30000 ms, it is automatically changed to 30000 ms.) If this parameter is set to -1, the receive status remains and never times out.
  - D2: Operation state. 0: Empty; 1: Triggering; 2: Transmitting; 3: Receiving; 16: Completed; 32: Transmission error; 48: Reception error; 64: Other error.

Timing Diagram



NOTE

- The instruction is triggered and executed on the rising edge.
- The D2 parameter displays the operating state machine of the serial port, including values of normal and abnormal states. The error codes of the instruction are not displayed in D2, but in the error table as errors of standard instructions.
- The timeout period refers to the total timeout duration for both transmission and reception.
- This instruction is executed only when the port is available, and only one instruction can be executed at the same time on the same port. This instruction is not executed if a port conflict or protocol setting error occurs. For details about the relevant error codes, see standard instruction errors.

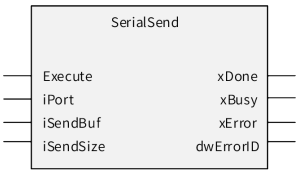
5.18.3 SerialSend

Introduction

SerialSend -- Serial port free protocol transmission

The SerialSend instruction implements free protocol communication through the serial port.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
SerialSend	Serial port free protocol transmission		SerialSend(Execute := ???, iPort := ???, iSendBuf := ???, iSendSize := ???, xDone => , xBusy => , xError => , dwErrorID =>);

The following table describes the instruction format.

Operand	Name	Description	Range	Data Type
16-bit Instruction	SerialSend: Triggered execution			
32-bit Instruction	---			
S1	iPort	Port number	0 to 15	INT
S2	iSendBuf	TX buffer	-	BYTE[] and INT[]
S3	iSendSize	Number of bytes to transmit	0 to 256	INT
D1	xDone	Completion flag[1]	ON or OFF	BOOL
D2	xBusy	Executing[1]	ON or OFF	BOOL
D3	xError	Error flag[1]	ON or OFF	BOOL
D4	dwErrorID	Error code[1]	For details, see 5.18.5 Error Codes of Serial Port Free Protocol Communication Instructions	INT

NOTE

[1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[2]	√	√	-	-	√	-	-	-
D2	√[2]	√	√	-	-	√	-	-	-
D3	√[2]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

NOTE

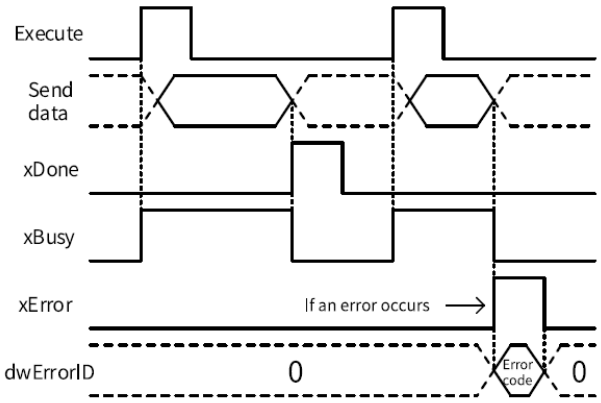
[2]: The X element is not supported.

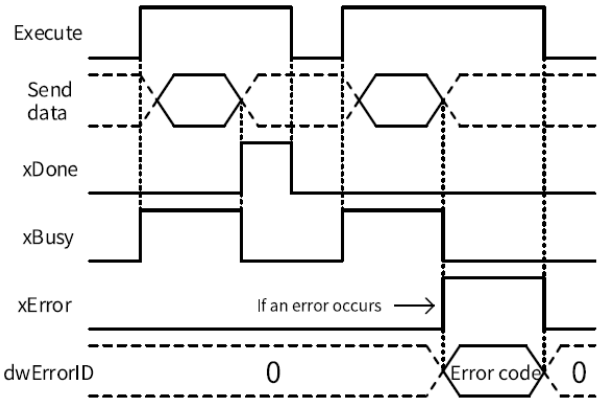
Function and Instruction Description

- Function description
  The SerialSend instruction implements data transmission of free protocols. After the instruction is triggered, data of the specified length is sent through the specified port. The corresponding output is updated during the transmission process.
- Parameter description
  S1: Port number (The port number corresponds to the serial port. You need to set it to the serial port number selected for communication.)

  S2: TX buffer. It is recommended that you set the size of this buffer to a value greater than 128 words or 256 bytes.

Timing Diagram





NOTE

- The time taken to send the data is limited by the serial port configuration and system scheduling. If the Done signal is reset, the instruction execution is completed.
- The instruction is triggered and executed on the rising edge.
- This instruction is executed only when the port is available, and only one instruction can be executed at the same time on the same port. This instruction is not executed if a port conflict or protocol setting error occurs. For details about the relevant error codes, see standard instruction errors.
- When the serial port RS232 or RS422 works in full duplex mode, one SerialSend instruction and one SerialRcv instruction can be called at the same time.

5.18.4 SerialRcv

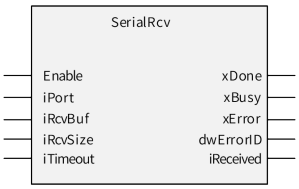
Introduction

SerialRcv -- Serial port free protocol reception and free protocol cancellation

The SerialRcv instruction is used to implement free protocol communication through the serial port.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
SerialRcv	Serial port free protocol reception		SerialRcv(Enable := ???, iPort := ???, iRcvBuf := ???, iRcvSize := ???,

Instruction	Name	LD Expression	LiteST Expression
			iTimeout := , xDone => , xBusy => , xError => , dwErrorID => , iReceived =>);

The following table describes the instruction format.

Operand	Name	Description	Range	Data Type
16-bit Instruction	SerialRcv: Continuous execution			
32-bit Instruction	---			
S1	iPort	Port number	0 to 15	INT
S2	iRcvBuf	RX buffer	-	BYTE[] and INT[]
S3	iRcvSize	Maximum number of bytes to receive	1 to 256	INT
S4	iTimeout	Receiving timeout[1]	-	INT
D1	xDone	Completion flag[1]	ON or OFF	BOOL
D2	xBusy	Executing[1]	ON or OFF	BOOL
D3	xError	Error flag[1]	ON or OFF	BOOL
D4	dwErrorID	Error code[1]	For details, see Error Codes of Serial Port Free Protocol Communication Instructions.	INT
D5	iReceived	Size of received data[1]	0 to 256	INT

NOTE

[1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
D1	√[2]	√	√	-	-	√	-	-	-
D2	√[2]	√	√	-	-	√	-	-	-
D3	√[2]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	√	-	-
D5	-	-	-	√	√	√	√	-	-

NOTE

[2]: The X element is not supported.

Function and Instruction Description

- Function Description
  The SerialRcv instruction implements data reception through free protocols. After the instruction is triggered, data of the specified length is received through the specified port. The corresponding output is updated during the reception process.

  When the SerialRcv instruction is in the receive waiting state, you can set the system variable _SerialSR.abort to a non-zero value to terminate the current reception process. One cycle after the xDone signal is set, the next reception process starts. The parameter modification takes effect immediately.

  You can use the system variables _SerialSR.startchar_en and _SerialSR.startchar[4] to set the receive start character function of the free protocol. The parameter modification takes effect next time data receiving is triggered. When _SerialSR.startchar_en is set to a value that falls within the range of 1 to 4 and equals to or smaller than the number of bytes to receive, the receive start character function is enabled. The start character length is specified by _SerialSR.startchar_en, and the start character content is specified by _SerialSR.startchar[4].

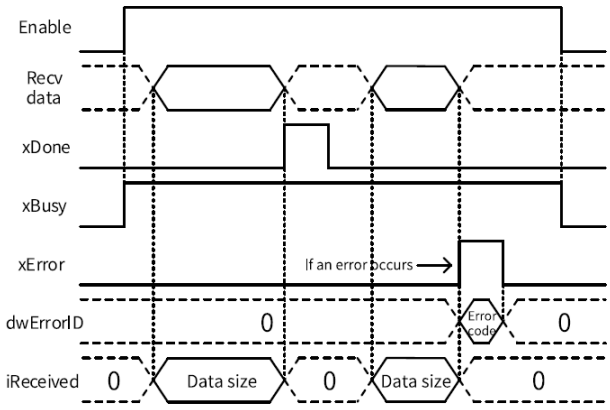
  You can use the system variables _SerialSR.endchar_en and _SerialSR.endchar[4] to set the receive end character function of the free protocol. The parameter modification takes effect next time data receiving is triggered. When _SerialSR.endchar_en is set to a value that falls within the range of 1 to 4, the receive end character function is enabled. The end character length is specified by _SerialSR.endchar_en, and the end character content is specified by _SerialSR.endchar[4].

You can use the system variables _SerialSR.Bytetimeout_en and _SerialSR.Bytetimeout to set the receive byte timeout function of the free protocol. The parameter modification takes effect next time data receiving is triggered. When _SerialSR.Bytetimeout_en is set to ON, the receive byte timeout function is enabled. The byte timer is specified by _SerialSR.Bytetimeout, and the minimum value is 1, in ms*1. After the byte timeout function is enabled and takes effect, the timer starts when the start byte or frame start character is received. If the idle time between received bytes is greater than the set time, the current receiving process is terminated. One cycle after the xDone signal is set, the next receiving process starts.

Each serial port corresponds to a separate system variable. _SerialSR corresponds to COM0, and _SerialSR1 to _SerialSR15 correspond to COM1 to COM15 respectively.

- Parameter description
  - S1: Port number (The port number corresponds to the serial port. You need to set it to the serial port number selected for communication.)
  - S2: RX buffer. It is recommended that you set the size of this buffer to a value greater than 128 words or 256 bytes.
  - S4: Timeout period. If the specified timeout period does not fall between 20 and 30000, in ms[1], it is automatically adjusted to the allowable range. If this parameter is set to -1, the receive status remains and never times out.
  - D1: Completion flag. When data of a specified length, a specified end character, or byte timeout signal is received, the current reception ends, and the completion flag is set to one scan cycle.

Timing Diagram



NOTE

- The instruction is executed when the enable signal is at a high level.
- The reception timeout period refers to the total period during which no frame is received.
- This instruction is executed only when the port is available, and only one instruction can be executed at the same time on the same port. This instruction is not executed if a port conflict or protocol setting error occurs. For details about the relevant error codes, see standard instruction errors.
- When the serial port RS232 or RS422 works in full duplex mode, one SerialSend instruction and one SerialRcv instruction can be called at the same time.

5.18.5 Error Codes of Serial Port Free Protocol Communication Instructions

The following table lists the error codes of serial port free protocol communication instructions.

Error code	Description
5601	The port number is out of range.
5602	Protocol error
5603	Port conflict
5604	The sent data length is out of range or smaller than 0.
5605	TX buffer error
5606	The received data length is out of range, or is equal to or smaller than 0.
5607	RX buffer error
5620	Port number changed
5621	Reception timeout

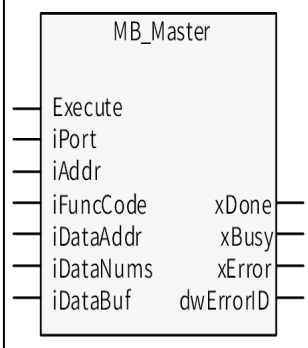
5.18.6 MB_Master

Introduction

MB_Master -- Transmission and reception through the serial Modbus protocol

The MB_Master instruction implements Modbus communication through the serial port.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MB_Master	Transmission and reception through the serial Modbus protocol		MB_Master(Execute := ?? ?, iPort := ???, iAddr := ???, iFuncCode := ???, diDataAddr := ???, iDataNums := ???, iDataBuf := ???, xDone => , xBusy => , xError => , dwErrorID =>);

16-bit Instruction	-
32-bit Instruction	MB_Master: Triggered execution

Operand	Name	Description	Range	Data Type
S1	iPort	Port number	0 to 15	INT
S2	iAddr	Slave station ID	0 to 255	INT
S3	iFuncCode	Function code	1 to 6, 15, or 16	INT
S4	diDataAddr	Address of the slave station to be accessed	0 to 65,535	DINT
S5	iDataNums	Bits or words to be accessed	-	INT
S6	iDataBuf	TX or RX buffer	-	BYTE[] and INT[]
D1	xDone	Completion flag[1]	ON or OFF	BOOL
D2	xBusy	Executing[1]	ON or OFF	BOOL
D3	xError	Error flag[1]	ON or OFF	BOOL
D4	dwErrorID	Error code[1]	For details, see 5.18.8 Error Codes of Modbus Communication Instructions.	INT

NOTE

[1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	√	-	-	-
D1	√[2]	√	√	-	-	√	-	-	-
D2	√[2]	√	√	-	-	√	-	-	-
D3	√[2]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

NOTE

[2]: The X element is not supported.

Function and Instruction Description

- Function description
  The MB_Master instruction implements Modbus communication through the serial port. After the instruction is triggered, the Modbus instruction is sent through the specified port. After the instruction is sent, the master station waits for the response instruction from the slave station.

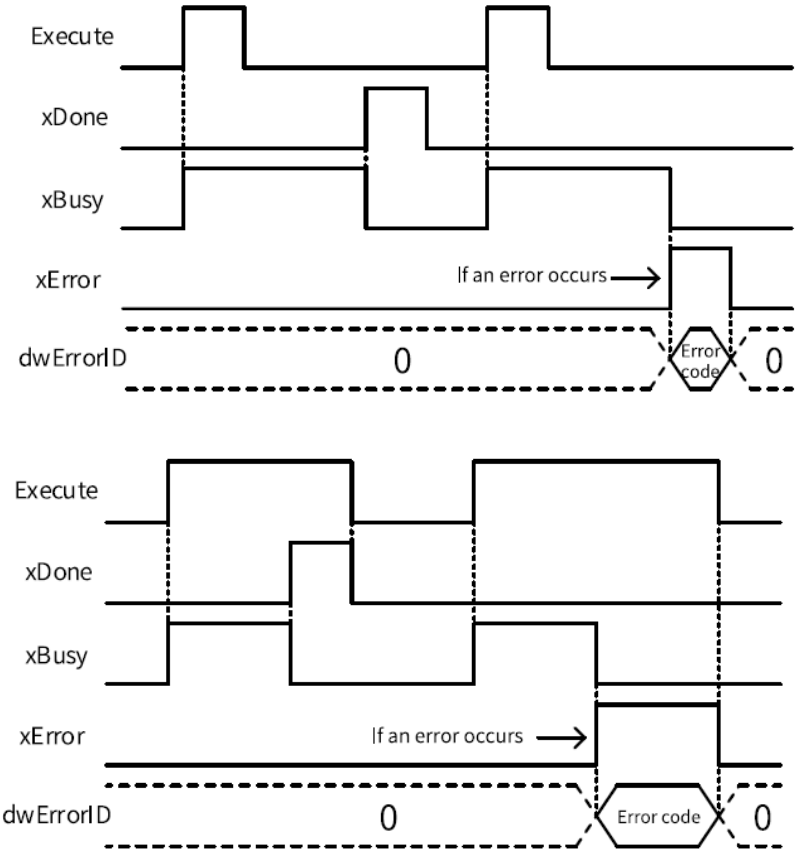
  Before using this instruction, you must set the communication protocol of the corresponding serial port to Modbus-RTU or Modbus-ASC master station.

  The system variable _MbMstEx.RetryTimes specifies the number of retransmissions. The value range is 0 to 15. The default value is 1. If the set value is out of this valid range, it will automatically be adjusted to the allowed range.

  Each serial port corresponds to a separate system variable. _MbMstEx corresponds to COM0, and _MbMstEx1 to _MbMstEx15 correspond to COM1 to COM15 respectively.

  Before using the MB_Master instruction, you must set the communication protocol of the corresponding COM port to Modbus-RTU or Modbus-ASC master station. The MB_Master instruction and Modbus serial port configuration table can be used simultaneously, and they share the timeout configuration (default: 500 ms).
- Parameter description
  - S1: Port number (The port number corresponds to the serial port. You need to set it to the serial port number selected for communication.)
  - S6: Buffer. You must set the buffer size to a value greater than 125 words or 250 bytes.

Timing Diagram



NOTE

- The instruction is triggered and executed on the rising edge.
- The MB_Master instruction and the Modbus configuration table for the same serial port share the same timeout configuration. The value range is 50 ms to 10000 ms, and the default value is 500 ms.
- This instruction is executed only when the port is available, and only one instruction can be executed at the same time on the same port. This instruction is not executed if a port conflict or protocol setting error occurs. For details about the relevant error codes, see standard instruction errors.

5.18.7 MB_Client

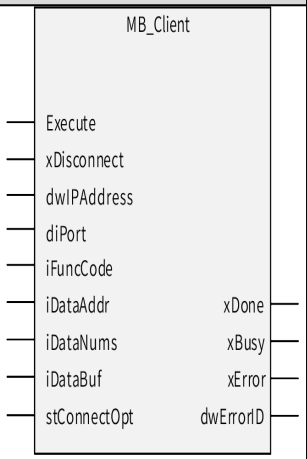
Description

MB_Client: Transmission and reception of the Modbus TCP protocol

This instruction is used to implement Modbus TCP Communication.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
MB_Client	Transmission and reception of the Modbus TCP protocol		MB_Client(Execute := ???, xDisconnect := , dwIPAddr := ???, diPort := ???,

Instruction	Name	LD Expression	LiteST Expression
			iFuncCode := ???, diDataAddr := ???, iDataNums := ???, iDataBuf := ???, stConnectOpt := ???, xDone => , xBusy => , xError => , dwErrorID =>);

16-bit instruction	-
32-bit instruction	MB_Client: Triggered execution

Operands	Name	Description	Value range	Data type
S1	xDisconnect	Disconnect[1]	ON/OFF	BOOL
S2	dwIPAddress	IP address of the server	-	IP
S3	diPort	Port number	1 to 65535	DINT
S4	iFuncCode	Parameter	1 to 6, 15, or 16	INT
S5	diDataAddr	Address of data to be accessed	0 to 65535	DINT
S6	iDataNums	Bits or words to be accessed	-	INT
S7	iDataBuf	TX or RX buffer	-	BYTE[] /INT[]
S8	stConnectOpt	Connection parameters and attributes	-	INT[]
D1	xDone	Completion flag[1]	ON/OFF	BOOL
D2	xBusy	Executing[1]	ON/OFF	BOOL
D3	xError	Error flag[1]	ON/OFF	BOOL
D4	dwErrorID	Error code[1]	For details, see 5.18.8 Error Codes of Modbus Communication Instructions.	INT

> **NOTE**
>
> [1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	√	√	√	-	-	√	-	-	-
S2	-	-	-	-	-	√	-	-	√
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	√	√	-	-
S7	-	-	-	√	√	√	-	-	-
S8	-	-	-	√	√	√	-	-	-
D1	√[2]	√	√	-	-	√	-	-	-
D2	√[2]	√	√	-	-	√	-	-	-
D3	√[2]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

> **NOTE**
>
> [2] The X element is not supported.

Function and Instruction Description

- Function description
  This instruction is used to implement client (master station) communication over Modbus TCP. After the instruction is triggered, an attempt is made to connect to the slave station based on the IP address and port number specified in the instruction. After the connection is set up, the Modbus instruction is sent. After the sending is completed, the master station waits for the response instruction from the slave station.

The MB_Client instruction supports up to 31 connections. When the number of connections exceeds 31, the instruction will report an error and the original connections need to be released.

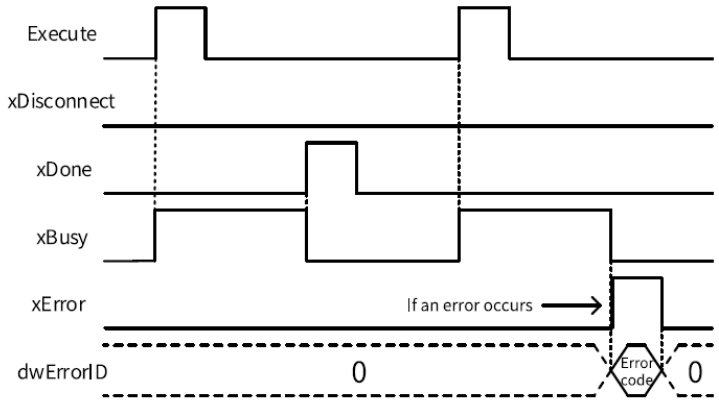
The MB_Client instruction and Modbus TCP configuration table can be used simultaneously without affecting each other.

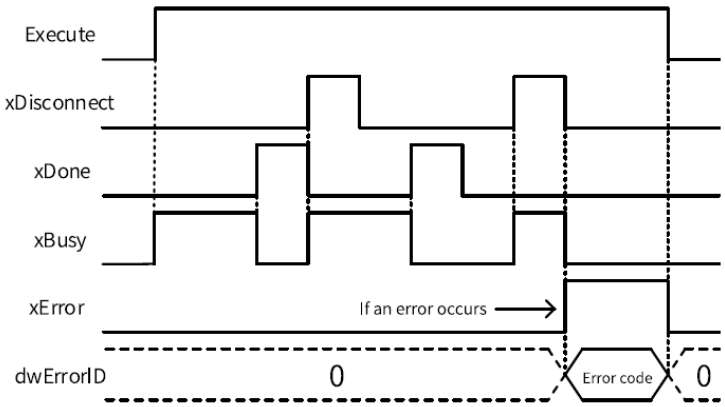
The MB_Client instruction execution cycle is affected by the program scan cycle. The repeat frame interval may deviate by two to three scan cycles.

- Parameter description
  - S1: Release the connection (When this parameter is set and the flow is enabled, the current connection is released.)
  - S7: Buffer. You must set the buffer size to a value greater than 125 word elements or 250 bytes.
  - S8: Connection parameters and attributes, such as the unit identifier, number of retransmissions, receive timeout. For details, see the following table.

Address	Data type	Value range	Default	Unit	Description
stConnectOpt	INT	0 to 1	-	-	Configuration enabling. 0: Use the default configuration, 1: Use the parameter configuration.
stConnectOpt+1	INT	0 to 255	255	-	Unit identifier
stConnectOpt+2	INT	0 to 15	1	Time	Repeated
stConnectOpt+3	INT	100 to 10000	500	ms	Reception timeout time

Sequence Diagram





NOTE

- The instruction is triggered and executed on the rising edge.
- The MB_Client instruction has a default receive timeout of 500 ms, with one repeat attempt.
- This instruction is only executed when the connection is available, and only one MB_Client instruction can be executed for a connection at any given time. This instruction is not executed if a connection conflict occurs. For details about the relevant error codes, see standard instruction errors.

5.18.8 Error Codes of Modbus Communication Instructions

The following table lists the error codes of Modbus communication instructions.

Error code	Description
6000	Network connection failed
6001	Parameter not supported
6002	Register/coil address out of range
6003	Improper data range
6004	Slave device fault
6128	Response station number and requested station number mismatch
6129	Response parameter and requested parameter mismatch
6130	Mismatched register/coil address in the response and request
6131	Mismatched response and request data
6240	Invalid mapping address in the configuration
6255	Slave station response timeout
6261	The port number is out of range.
6262	Protocol error

Error code	Description
6263	Port conflict
6264	Incorrect slave station node
6280	Slave station disabled
6290	Network connection pool already full
6291	Network connection already occupied
6292	Incorrect response ID
6293	Incorrect received data length
6294	Non existing network connection

5.18.9 SysHC_NTPClient

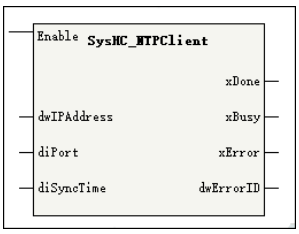
Introduction

SysHC_NTPClient – NTP time synchronization

The SysHC_NTPClient instruction is used to implement the NTP time synchronization function.

Version

- Firmware version: V6.5.0.0 and later
- Software version: AutoShop V4.11.0.0 and later

Instruction	Name	LD Expression	LiteST Expression
SysHC_NTPClient	NTP time synchronization		-

The following table describes the instruction format.

16-bit Instruction	---			
32-bit Instruction	SysHC_NTPClient: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	dwIPAddress	IP address of the server	-	DINT, IP, and POINTER
S2	diPort	Port number	1 to 65535	DINT and POINTER
S3	diSyncTime		10 to 86400	DINT and POINTER

		PLC synchronization time		
D1	xDone	Completion flag*1	ON or OFF	BOOL and POINTER
D2	xBusy	Executing*1	ON or OFF	BOOL and POINTER
D3	xError	Error flag*1	ON or OFF	BOOL and POINTER
D4	dwErrorID	Error code*1*2	-	INT and POINTER

NOTE

- *1: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.
- *2: For details, see Error Codes of Socket Communication Instructions.

The following table lists soft elements in the instruction.

Operand	Bit				Word			Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	Bits of Custom Word Variable	D, R, W	Pt	Custom Word Variable	K, H	E	
S1	_	-	-	-	√	√	-	-	-	√
S2	-	-	-	-	√	√	√	√	-	-
S3	-	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	√	-	√	-	-	-	-
D2	√[1]	√	√	√	-	√	-	-	-	-
D3	√[1]	√	√	√	-	√	-	-	-	-
D4	-	-	-	-	√	√	√	-	-	-

NOTE

[1]: The X element is not supported.

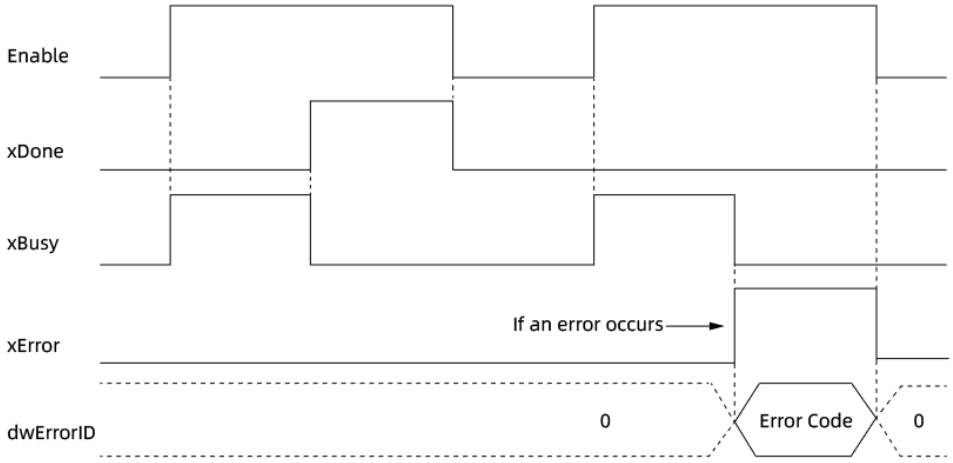
Function and Instruction Description

- Function description
  - The SysHC_NTPClient instruction is used to implement the NTP time synchronization function of the PLC. After the instruction is enabled, the PLC is connected to the specified NTP server and performs time synchronization at the specified cycle.

- The NTP synchronization accuracy of the system time is 100 ms. If the difference between the system time and the NTP server time is 100 ms, the PLC synchronizes the NTP server time as the system time and records a log.
- The NTP synchronization accuracy of the RTC time is 1s. If the difference between the system time and the NTP server time is 1s, the PLC synchronizes the NTP server time as the RTC time and records a log.

- Parameter description
  S3 specifies the synchronization time, in seconds. After this instruction is enabled, the PLC performs NTP time synchronization based on the S3 value. If the value is less than the lower limit or greater than the upper limit, the lower limit and upper limit are used automatically.

Timing Diagram



5.18.10 TCP_Listen

Description

TCP_Listen: TCP listening

The server must wait for the client's connection request. When working as a server, the local machine uses the TCP_Listen instruction to listen to connection requests from clients.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TCP_Listen	Serial port free protocol reception	TCP_Listen Execute Socket Port Active Busy Error ErrorID	-

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	TCP_Listen: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Socket	Socket*1	No	-	-	_sSocket

S2	Port	Local port to listen on*2	No	-	1 to 65535	DINT
D1	Active	Active state	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorID	Error code	Yes	0	*3	INT

NOTE

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: System internal ports (23, 12939, and 12940) and the Modbus-TCP server port (502) cannot be used.
- *3: For details, see 5.18.25 Error Codes of Socket Communication Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

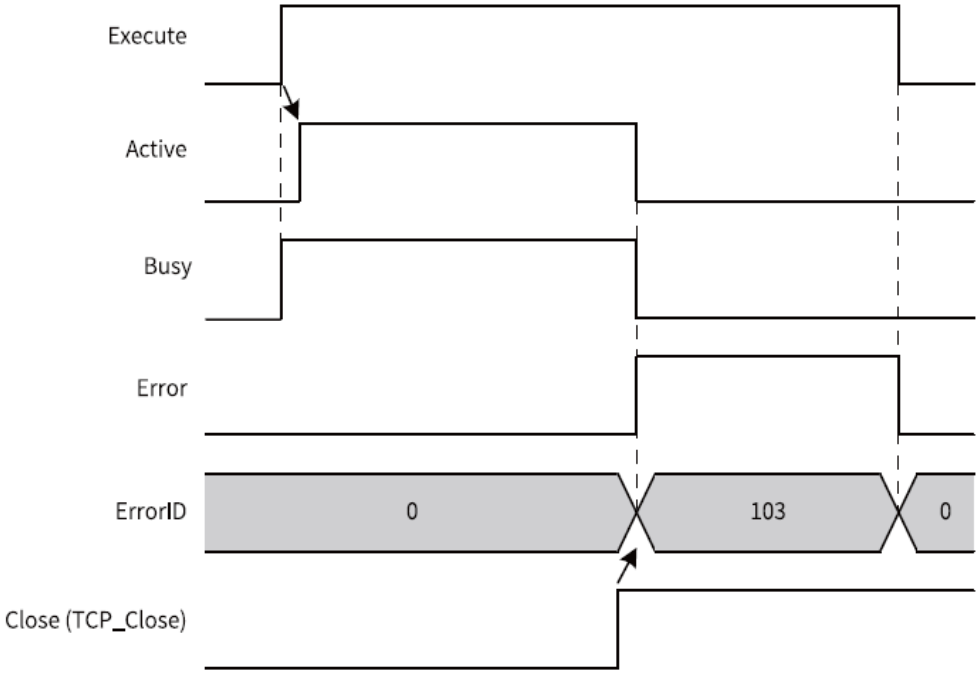
NOTE

[1] The X element is not supported.

Function

The server must wait for the client's connection request. The TCP_Listen instruction is used to listen on the specified local port to wait for client requests. Upon receiving a connection request from the client, the server needs to use the TCP_Accept instruction to establish communication with the client.

Sequence Diagram



5.18.11 TCP_Accept

Description

TCP_Accept: Connection request accept

Upon receiving a connection request from the client, a server in listening state will put the client in the waiting queue. When working as a server, the local machine uses the TCP_Accept instruction to receive connection request from clients.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TCP_Accept	Connection request accepted	TCP_Accept Excute ListeningSocket Connected Busy ConnectedSocket Error ErrorID	-

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	TCP_Accept: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	ListeningSocket	Listening socket*1	No	-	-	_sSocket
D1	Connected	Connection state	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	ConnectedSocket	Connection socket*1	No	-	-	_sSocket
D4	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Error code	Yes	0	*2	INT

NOTE

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see 5.18.25 Error Codes of Socket Communication Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	-	-	-	-
D4	√[1]	-	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

NOTE

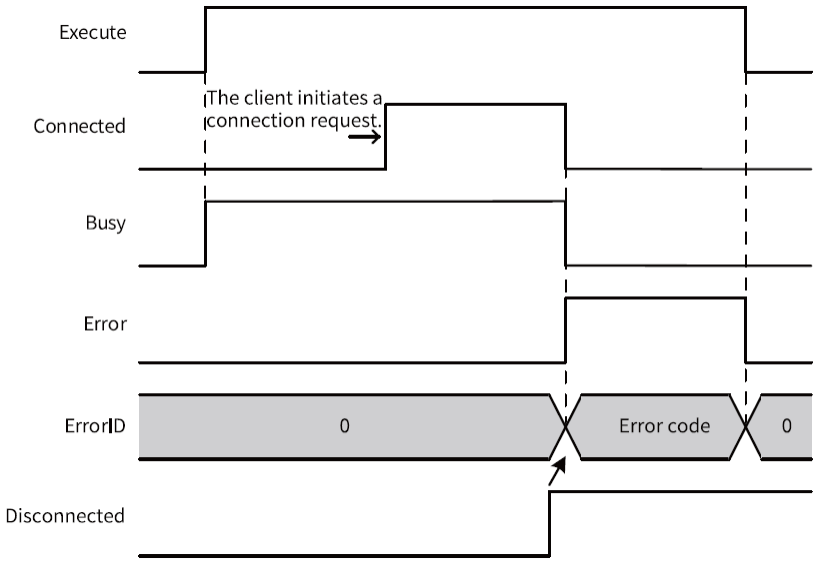
[1] The X element is not supported.

Function

Upon receiving a connection request from the client, a server in listening state needs to use the TCP_Accept instruction to establish communication with the client. After communication is successfully established, the server can transmit or receive data by using TCP_Send or TCP_Receive.

The server can establish communication with multiple clients through the same local port by executing multiple TCP_Accept instructions.

Sequence Diagram



5.18.12 TCP_Connect

Description

TCP_Connect: TCP connection request initiation

To communication with the server, a client needs to initiate a connection request to the server. When working as a client, the local machine uses the TCP_Connect instruction to initiate a connection request.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TCP_Connect	TCP connection request initiation	TCP_Connect Execute Connected Socket Busy IPAddress Error Port ErrorID	-

The instruction formats are as follows.

	-

16-bit instruction	
32-bit instruction	TCP_Connect: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	IPAddress	IP address	No	-	-	DINT, IP, POINTER
S3	Port	Port	No	-	1 to 65535	DINT
D1	Connected	Connection state	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorID	Error code	Yes	0	For details, See 5.18.25 Error Codes of Socket Communication Instructions.	INT

NOTE

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

NOTE

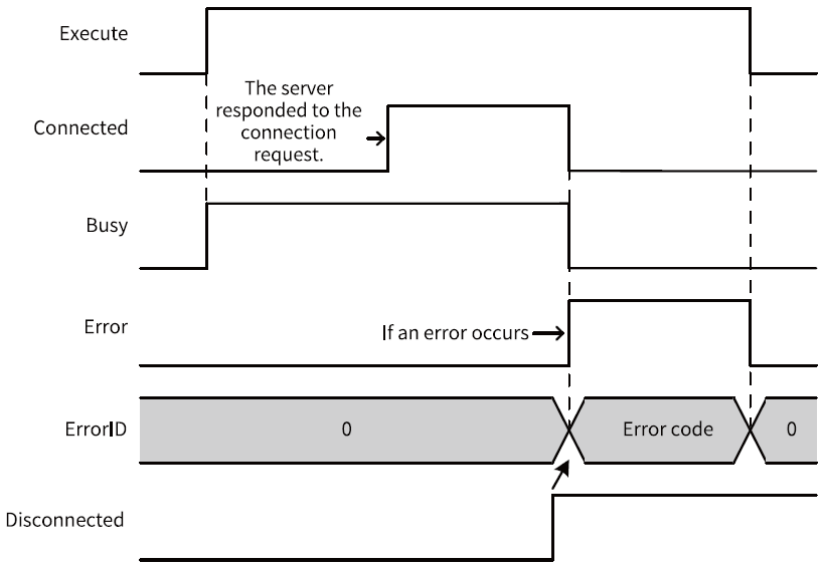
[1] The X element is not supported.

Function and Instruction Description

When the local machine works as a client and needs to communicate with the server, it executes the TCP_Connect instruction to connect to the specified port of the server. After the server accepts the connection request, the client can transmit or receive data by using TCP_Send or TCP_Receive.

After sending a connection request to the server by using the TCP_Connect instruction, the client waits at most 127 seconds. If the server does not respond, the connection fails.

Sequence Diagram



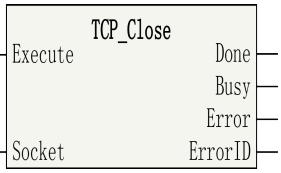
5.18.13 TCP_Close

Introduction

TCP_Close -- Close TCP connection

The TCP_Close instruction closes the connection or listening after communication is completed.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TCP_Close	Close TCP connection		-

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	TCP_Close: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Socket	Socket*1	No	-	-	_sSocket
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON or OFF	BOOL
D3	Error	Function block error flag	Yes	OFF	ON or OFF	BOOL
D4	ErrorID	Error code	Yes	0	For details, see 5.18.25 Error Codes of Socket Communication Instructions.	INT

NOTE

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, variables of the _sSocket type are not supported, and an INT[20] array can be used instead.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	--

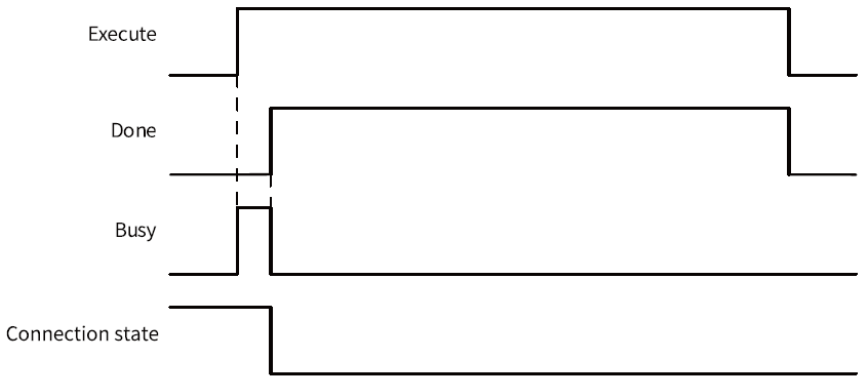
NOTE

[1]: The X element is not supported.

Function and Instruction Description

The TCP_Close instruction closes the connection, stops listening, or terminates the connecting socket after communication is completed.

Timing Diagram



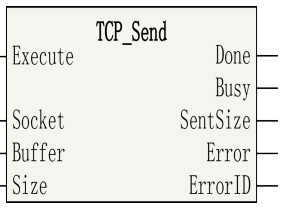
5.18.14 TCP_Send

Description

TCP_Send: TCP data transmission

After the connection between the server and the client is successfully established, data can be transmitted to the remote host by using the TCP_Send instruction.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TCP_Send	TCP data transmission		-

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	TCP_Send: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	Buffer	Data buffer	No	-	-	BYTE[]/INT[]
S3	Size	Data length, in byte	Yes	0	0 to 32767	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	SentSize	Size of sent data	Yes	0	0 to 32767	INT
D4	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Error code	Yes	0	For details, See 5.18.25 Error Codes of Socket Communication Instructions.	INT

NOTE

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

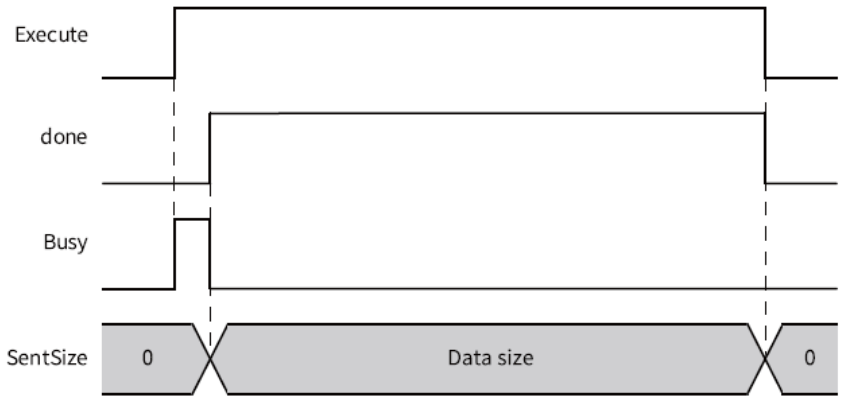
Function and Instruction Description

When the data size parameter (size) is 0, data is transmitted as character strings. That is, the data between the first byte and the terminator (excluding the terminator) in the data buffer (Buffer) is sent. The ASCII code of the terminator is 0.

After the connection between the server and the client is successfully established, the local machine uses the TCP_Send instruction to send the data in the buffer with the specified length to the remote host.

The data size (size) must be less than or equal to the actual size of the data buffer (Buffer); otherwise, there is a risk of out-of-bounds data access.

Sequence Diagram



> **NOTE**
>
> If this instruction is triggered again when the data frame of the instruction is still being sent, an error is reported due to abnormal timing. To avoid this problem, it is recommended that the next transmission or instruction be triggered upon the state change of the Done or Error signal.

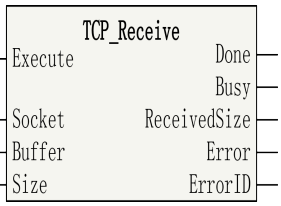
5.18.15 TCP_Receive

Description

TCP_Receive: TCP data reception

After the connection between the server and the client is successfully established, message data transmitted by the remote host can be obtained from the specified socket by using the TCP_Receive instruction.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TCP_Receive	TCP data reception		-

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	TCP_Receive: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	Buffer	Data buffer	No	-	-	BYTE[]/INT[]
S3	Size	Data length, in byte	No	-	1 to 32767	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	ReceivedSize	Size of received data	Yes	0	0 to 32767	INT

D4	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Error code	Yes	0	*2	INT

NOTE

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see 5.18.25 Error Codes of Socket Communication Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

NOTE

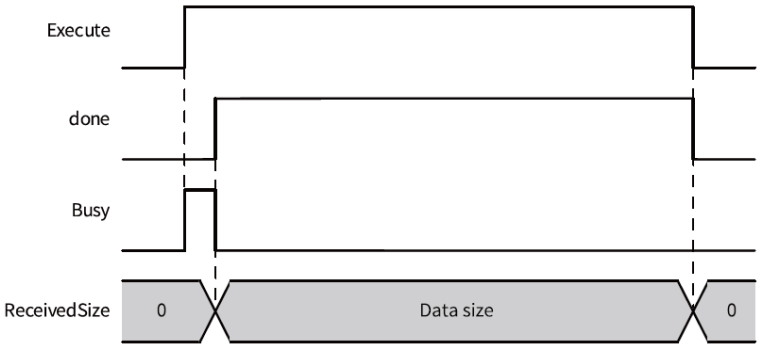
[1] The X element is not supported.

Function and Instruction Description

The data size (size) must be less than or equal to the actual size of the data buffer (Buffer); otherwise, there is a risk of out-of-bounds data access.

After the connection between the server and the client is successfully established, message data transmitted by the remote host to the local machine will be stored in the socket buffer area. The TCP_Receive instruction is used to obtain the received message data from the specified socket buffer area.

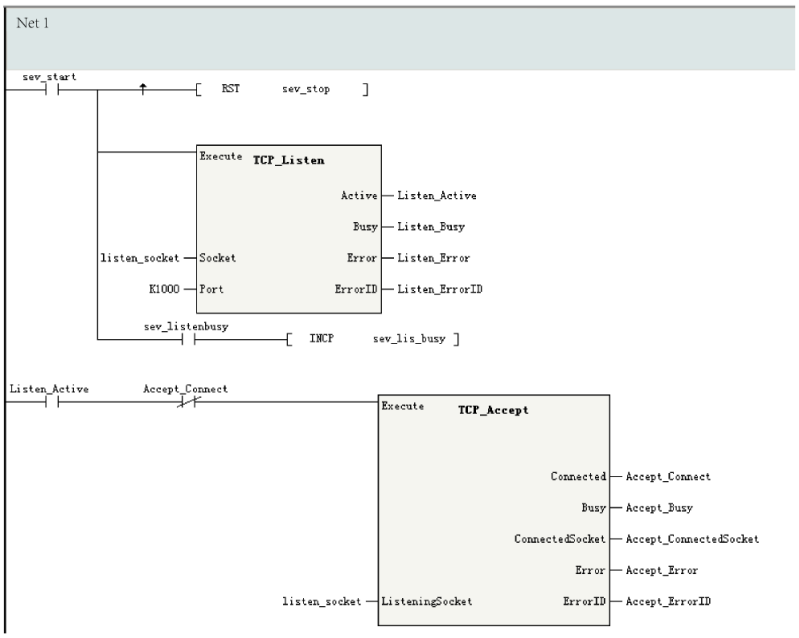
Sequence Diagram

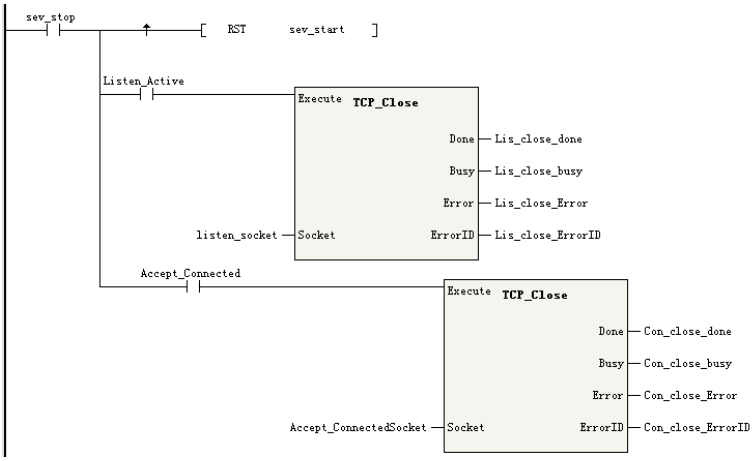


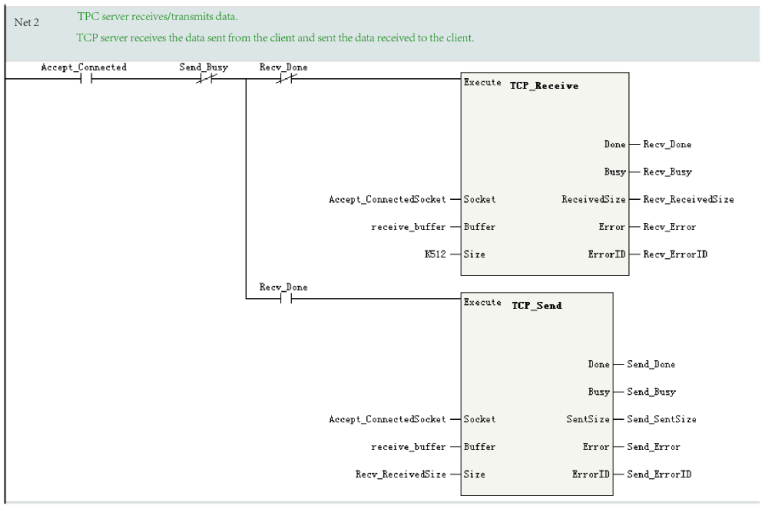
5.18.16 TCP Server Communication Instance

Working as a TCP server, the H5U listens on TCP port 1000 by using the TCP_Listen instruction and sends the data received from the client back to the client after accepting the connection request from the client.

The instance is programmed as follows:



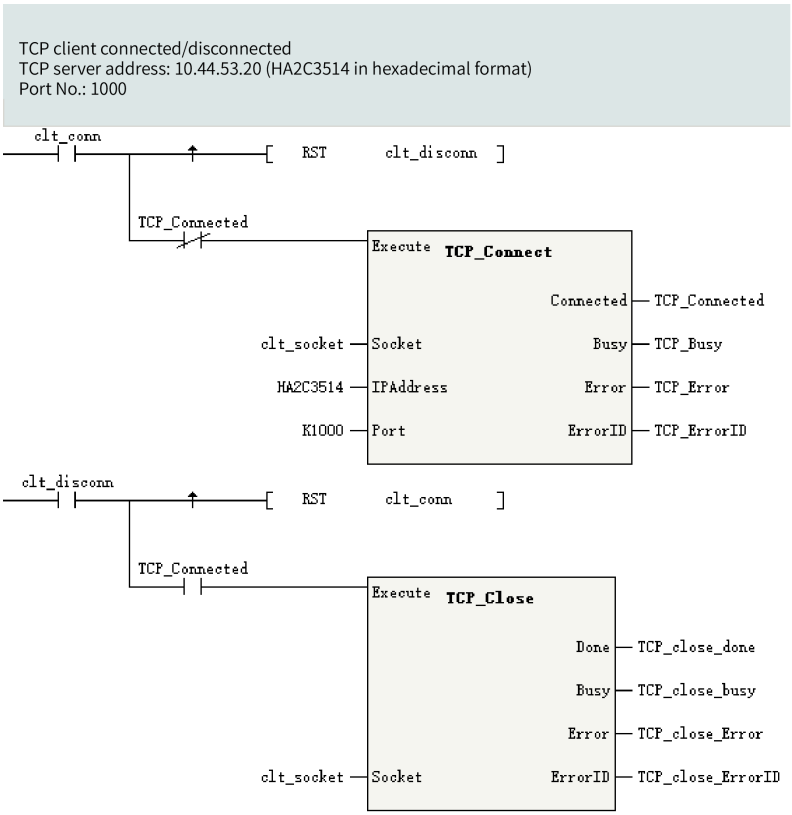


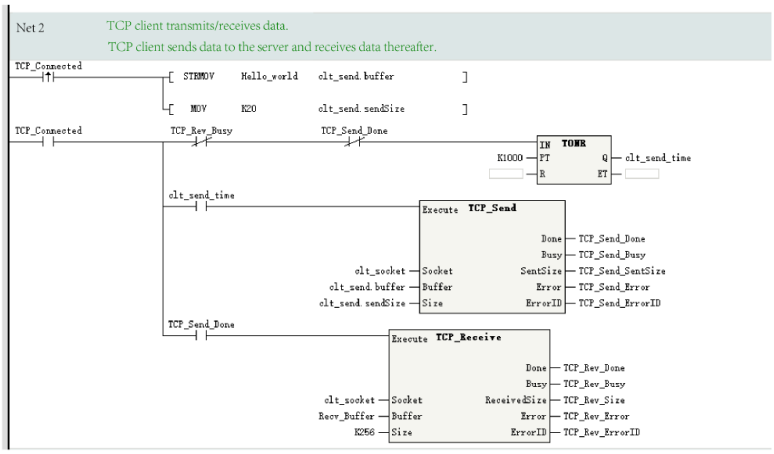


5.18.17 TCP Client Communication Instance

Working as a client, the H5U sends a connection request to the server at 10.44.53.20:1000. After the connection is established successfully, it sends “Hello!” to the server periodically (at an interval of 1 second) and stops sending "Hello!" after receiving any data sent by the server.

The instance is programmed as follows:





5.18.18 Connection-Oriented TCP Socket Communication

The Socket is a two-way communication interface. Hosts in the network transmits data through the interface provided by the socket.

The Ethernet socket interface is provided. By using sockets, users can easily implement communication between different devices with the TCP/IP network. The following table defines the socket structure type (_sSOCKET).

Member	Type	Read/Write	Function
ID	DINT	Read-only	ID
Type	INT	Read-only	Socket type 1: TCP 2: UDP
LocalPort	DINT	Read-only	Local port
RemoteIP	DINT	Read-only	Remote IP address
RemotePort	DINT	Read-only	Remote port
Active	BOOL	Read-only	Active state
Connected	BOOL	Read-only	Connection state
Listening	BOOL	Read-only	Listening state
Reserved0	BOOL[13]	Read-only	Reserved
Connections	INT	Read-only	Number of current connections
Reserved1	INT	Read-only	Reserved
Descriptor	DINT	Read-only	Reserved
ListeningSocket	DINT	Read-only	Reserved

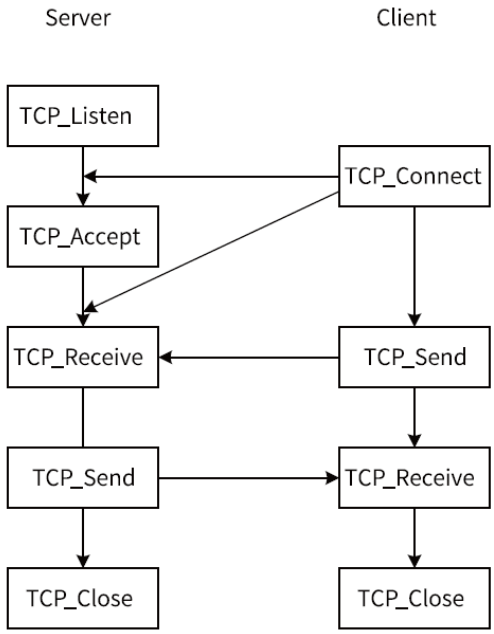
Member	Type	Read/Write	Function
Reserved2	DINT[2]	Read-only	Reserved

At present, AutoShop does not support system type custom variables, and structures are not supported as instruction input parameters. Therefore, the _sSOCKET variable is temporarily replaced by an INT[20] array.

The Transmission Control Protocol (TCP) is a connection-oriented, reliable, byte stream-based transport layer communication protocol.

An internetwork is very different from a single network because different parts of an internetwork can have vastly different topologies, bandwidths, latency, packet sizes, and other parameters. TCP is designed to dynamically adapt to these characteristics of the Internet and to demonstrate robustness in the face of various failures.

The process of a connection-oriented socket TCP communication interface is shown in the following figure.

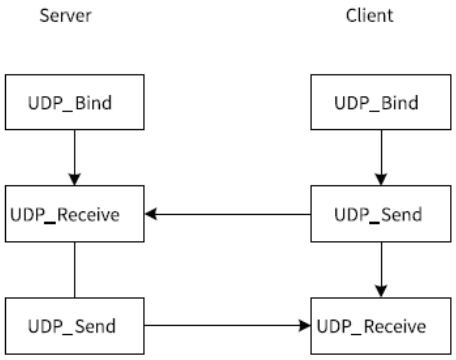


5.18.19 Connectionless UDP Socket Communication

User Datagram Protocol (UDP) is a connectionless transport layer protocol, which is mainly used in transmissions that do not require sequential arrival of packets. The check and sorting of packet transmission sequence is completed by the application layer. UDP provides simple and unreliable transaction oriented information transmission services.

UDP packets lack the QoS, sequence assurance, and flow control fields, resulting in poor reliability. However, because UDP protocol has fewer control options, it has low latency and high data transmission efficiency during data transmission. Therefore, UDP is suitable for applications with low reliability requirements or applications that can ensure reliability.

The following figure shows the UDP communication socket process.



5.18.20 UDP_Bind

Description

UDP_Bind: UDP socket binding

Before data transmission or reception through UDP, you need to bind the socket to a local port. The UDP_Bind instruction is used to bind the socket to the specified UDP port.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
UDP_Bind	UDP socket binding	UDP_Bind Enable — Active Busy Socket — Error Port — ErrorID	-

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	UDP_BIND: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	Port	Port*2	Yes	0	0 to 65535	DINT
D1	Active	Active state	Yes	FALSE	ON/OFF	BOOL
D2	Busy	Executing	Yes	FALSE	ON/OFF	BOOL
D3	Error	Function block error flag	Yes	FALSE	ON/OFF	BOOL

D4	ErrorID	Error code*3	Yes	0	-	INT

NOTE

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: System internal ports (12939 and 12940) cannot be used.
- *3: For details, see 5.18.25 Error Codes of Socket Communication Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

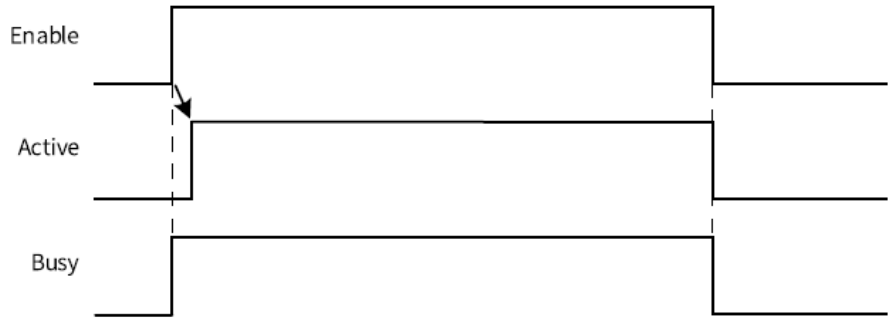
NOTE

[1] The X element is not supported.

Function and Instruction Description

Before data transmission or reception through UDP, you need to bind the socket to a local port. The UDP_Bind instruction is used to bind the socket to the specified UDP port. When the specified port ID is 0 or empty, UDP_Bind will automatically assign a random port ID, which is then indicated by LocalPort of the socket.

Sequence Diagram



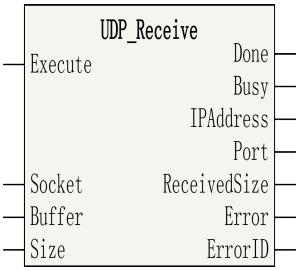
5.18.21 UDP_Receive

Description

UDP_Receive: UDP data reception

The UDP_Receive instruction is used to receive data sent by the remote host.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
UDP_Receive	UDP data reception		-

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	UDP_Receive: Continuous execution

Operands	Name	Description	是否为空	Default	Value range	Data type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	Buffer	Data buffer	No	-	-	BYTE[]/INT[]
S3	Size	Data size	No	-	1 to 32767	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	IPAddress	IP address	Yes	0	-	DINT

D4	Port	Port	Yes	0	1 to 65535	DINT
D5	ReceivedSize	Size of received data	Yes	0	0 to 32767	INT
D6	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D7	ErrorID	Error code*2	Yes	0	-	INT

NOTE

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see 5.18.25 Error Codes of Socket Communication Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-
D6	√[1]	√	√	-	-	√	-	-	-
D7	-	-	-	√	√	√	-	-	-

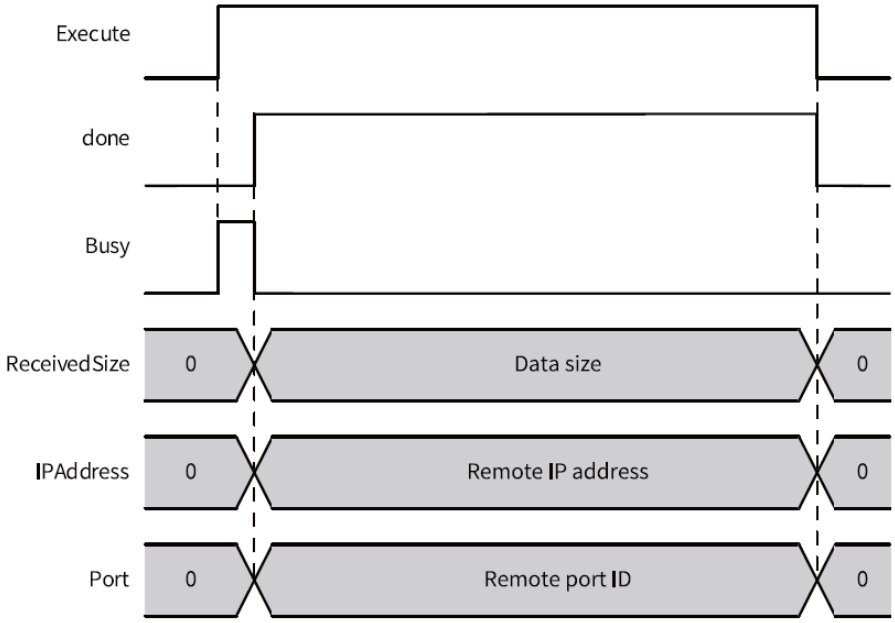
> **NOTE**
>
> [1] The X element is not supported.

Function

Message data transmitted by the remote UDP host to the local machine will be stored in the socket buffer area. The UDP_Receive instruction is used to obtain the received message data from the specified socket buffer area.

The data size (size) must be less than or equal to the actual size of the data buffer (Buffer); otherwise, there is a risk of out-of-bounds data access.

Sequence Diagram



5.18.22 UDP_Send

Description

UDP_Send: UDP data transmission

The UDP_Send instruction is used to send data to the specified remote host.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
UDP_Send	UDP data transmission	UDP_Send Execute Socket IPAddress Port Buffer Size Done Busy SentSize Error ErrorID	-

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	UDP_Send: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	IPAddress	IP address	No	-	-	DINT
S3	Port	Port	No	-	1 to 65535	DINT
S4	Buffer	Data buffer	No	-	-	BYTE[]/INT[]
S5	Size	Data size	Yes	0	0 to 32767	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	SentSize	Size of sent data	Yes	0	0 to 32767	INT
D4	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Error code	Yes	0	*2	INT

NOTE

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see 5.18.25 Error Codes of Socket Communication Instructions.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S4	-	-	-	√	√	-	-	-	-
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

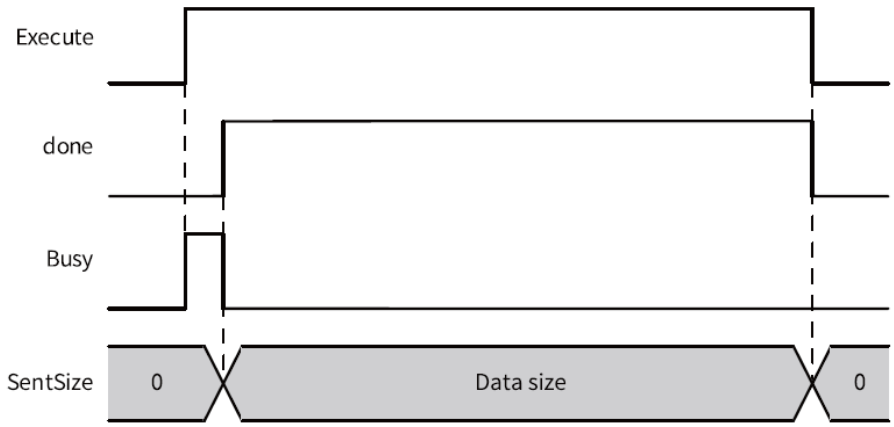
NOTE

[1] The X element is not supported.

Function

When the data size parameter (size) is 0, data is transmitted as character strings. That is, the data between the first byte and the terminator (excluding the terminator) in the data buffer (Buffer) is sent. The ASCII code of the terminator is 0.

Sequence Diagram



NOTE

If this instruction is triggered again when the data frame of the instruction is still being sent, an error is reported due to abnormal timing. To avoid this problem, it is recommended that the next transmission or instruction be triggered upon the state change of the Done or Error signal.

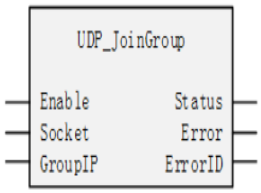
5.18.23 UDP_JoinGroup

Introduction

UDP_JoinGroup -- Add UDP port to multicast group

The UDP_JoinGroup instruction adds a motor to the specified multicast group.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
UDP_JoinGroup	Add UDP port to multicast group		-

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	UDP_JoinGroup: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	GroupIP	IP address of the multicast group to join	No	-	-	DINT and IP
D1	Status	State of the multicast group to join	Yes	OFF	ON or OFF	BOOL
D2	Error	Operation error flag of the function block	Yes	OFF	ON or OFF	BOOL
D3	ErrorID	Error code*2	Yes	0	-	INT

NOTE

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, variables of the _sSocket type are not supported, and an INT[20] array can be used instead.
- *2: For details, see 5.18.25 Error Codes of Socket Communication Instructions.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	√
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

NOTE

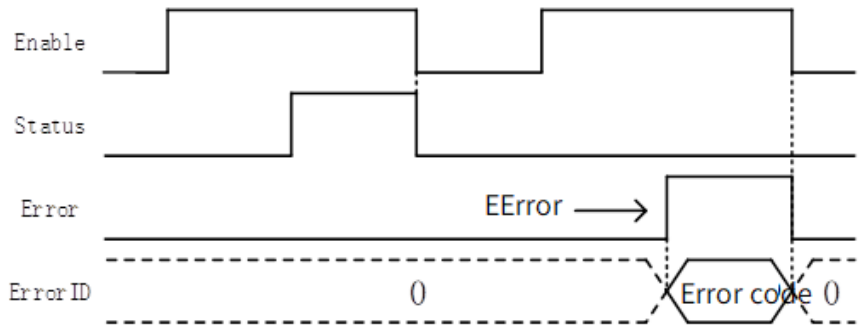
[1]: The X element is not supported.

Function and Instruction Description

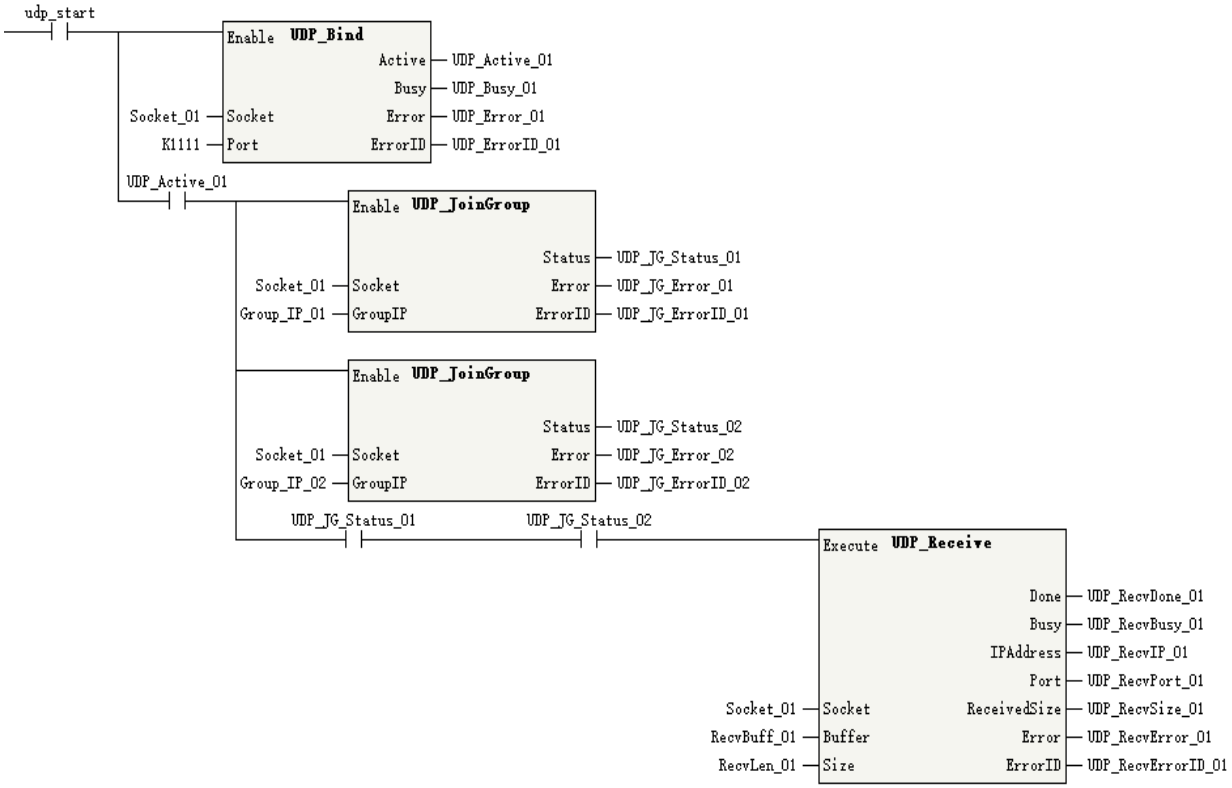
The UDP_JoinGroup instruction adds the specified network port of the motor to the multicast group of the specified IP address. After the port is added to the multicast group, the port can receive data packets from the specified multicast IP address.

This instruction can be called multiple times to add the same network port to multiple multicast groups.

Timing Diagram

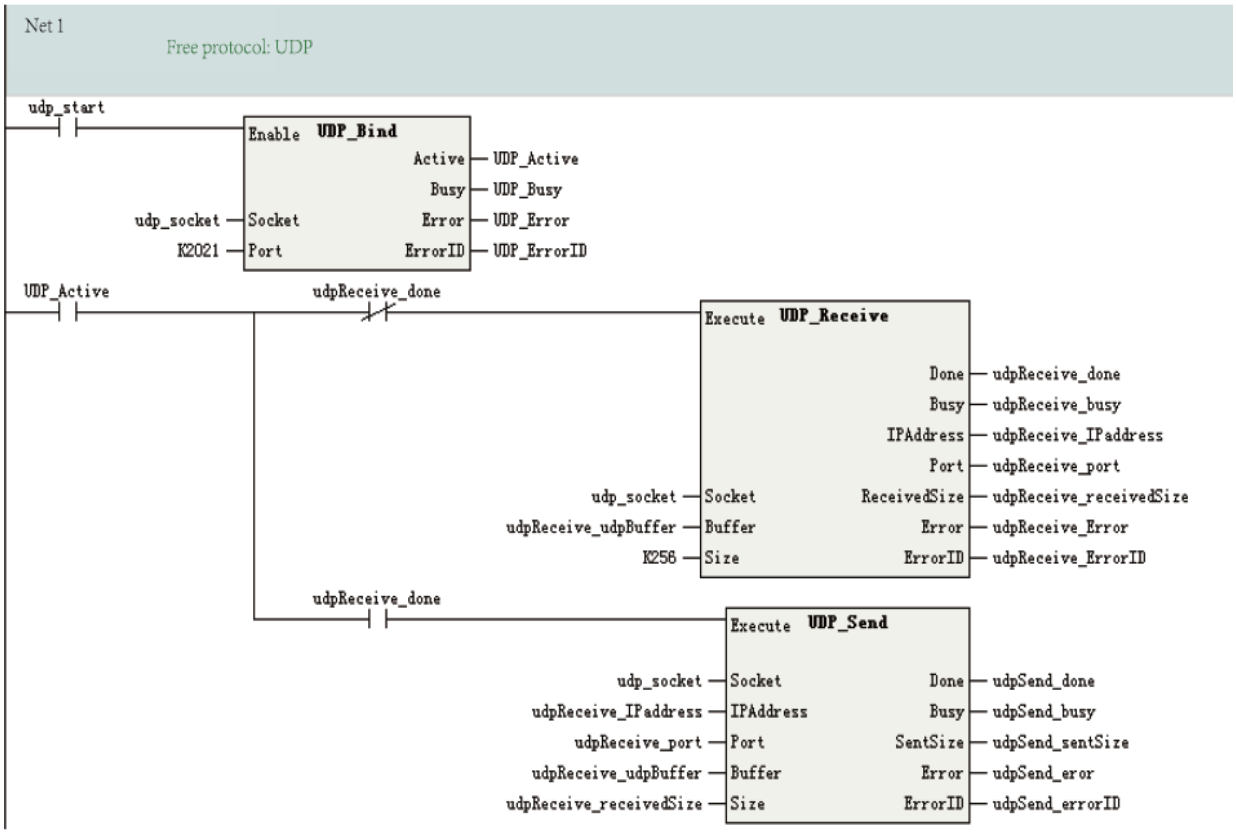


Instruction Example



5.18.24 UDP Communication Instance

When communicating with a remote motor through the UDP free protocol, the H5U uses the UDP_Bind instruction to bind the UDP port to port K2021. After receiving data from the remote motor, the UDP port sends the received data back to the remote motor. The following figure shows the program.



5.18.25 Error Codes of Socket Communication Instructions

The following table lists the error codes of socket communication instructions.

The error codes are defined as follows.

Error code	Description
1	Operation not permitted
2	No such file or directory
3	No such process
4	Interrupted system call
5	I/O error
6	No such device or address
7	Arg list too long
8	Exec format error
9	Bad file number
10	No child subprocesses
11	Try again
12	Out of memory
13	Permission denied
14	Bad address
15	Block device required
16	Device or resource busy
17	File exists
18	Cross-device link
19	Device does not exist.
20	Not a directory
21	Is a directory
22	Invalid argument
23	File table overflow
24	Too many open files
25	Not a typewriter
26	Text file busy

Error code	Description
27	File too large
28	No space left on device
29	Illegal seek
30	Read-only file system
31	Too many links
32	Broken pipe
33	Math argument out of domain of func
34	Math result not representable
35	Resource deadlock would occur
36	File name too long
37	No record locks available
38	Function not implemented
39	Directory not empty
40	Too many symbolic links encountered
41	Operation would block
42	No message of desired type
43	Identifier removed
44	Channel number out of range
45	Level 2 not synchronized
46	Level 3 halted
47	Level 3 reset
48	Link number out of range
49	Protocol driver not attached
50	No CSI structure available
51	Level 2 halted
52	Invalid exchange

Error code	Description
53	Invalid request descriptor
54	Exchange full
55	No anode
56	Invalid request code
57	Invalid slot
58	File locking deadlock error
59	Bad font file format
60	Device not a stream
61	No data available
62	Timer expired
63	Out of streams resources
64	Machine is not on the network
65	Package not installed
66	Object is remote
67	Link has been severed
68	Advertise error
69	Srmount error
70	Communication error on send
71	Protocol error
72	Multihop attempted
73	RFS specific error
74	Not a data message
75	Value too large for defined data type
76	Name not unique on network
77	File descriptor in bad state
78	Remote address changed

Error code	Description
79	Cannot access a needed shared library
80	Accessing a corrupted shared library
81	.lib section in a.out corrupted
82	Attempting to link in too many shared libraries
83	Cannot exec a shared library directly
84	Illegal byte sequence
85	Interrupted system call must be restarted
86	Streams pipe error
87	Too many users
88	Socket operation on non-socket
89	Destination address required
90	Message too long
91	Protocol wrong type for socket
92	Protocol not available
93	Protocol not supported
94	Socket type not supported
95	Operation not supported on transport endpoint
96	Protocol family not supported
97	Address family not supported by protocol
98	Address already in use
99	Cannot assign requested address
100	Network is down
101	Network is unreachable
102	Network dropped connection because of reset
103	Software caused connection abort
104	Connection reset by peer

Error code	Description
105	No buffer space available
106	Transport endpoint is already connected
107	Transport endpoint is not connected
108	Cannot send after transport endpoint shutdown
109	Too many references: cannot splice
110	Connection times out.
111	Connection refused
112	Host is down
113	No route to host
114	Operation already in progress
115	Operation now in progress
500	Transmission timing triggering error. The previous data frame is still being sent

5.18.26 SFTP_GetFile

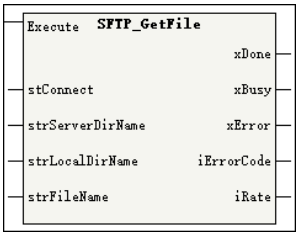
Version

- PLC: Easy series
  - The Easy 511 series supports local network ports but not expanded network ports.
  - Easy series models without Ethernet network ports do not support this instruction.
- Firmware version: V6.5.0.0 and later
- Software version: AutoShop V4.11.0.0 and later

Introduction

SFTP_GetFile -- Obtain a file

The SFTP_GetFile instruction is used to download a file from the SFTP server to your computer.

Instruction	Name	LD Expression	LiteST Expression
SFTP_GetFile	Obtain a file		SFTP_GetFile(Execute := ???, stConnect := ???, strServerDirName := ???, strLocalDirName := ???, strFileName := ???, xDone => ,

Instruction	Name	LD Expression	LiteST Expression
			xBusy => , xError => , iErrorCode => , iRate =>);

The following table describes the instruction format.

16-bit Instruction	---			
32-bit Instruction	SFTP_GetFile			
Operand	Name	Description	Range	Data Type
S1	Execute	Triggered execution	ON or OFF	BOOL
S2	stConnect	Connection information	-	SFTP_CONNECT_INFO
S3	strServerDirName	Directory name on the server	-	STRING
S4	strLocalDirName	Local directory	-	STRING
S5	strFileName	Name of the file to be uploaded	-	STRING
D1	xDone	Completion flag*1	ON or OFF	BOOL
D2	xBusy	Executing*1	ON or OFF	BOOL
D3	xError	Error flag*1	ON or OFF	BOOL
D4	iErrorCode	Error code*1	-	INT16
D5	iRate	Download progress	0 to 100	INT16

The following table lists soft elements in the instruction.

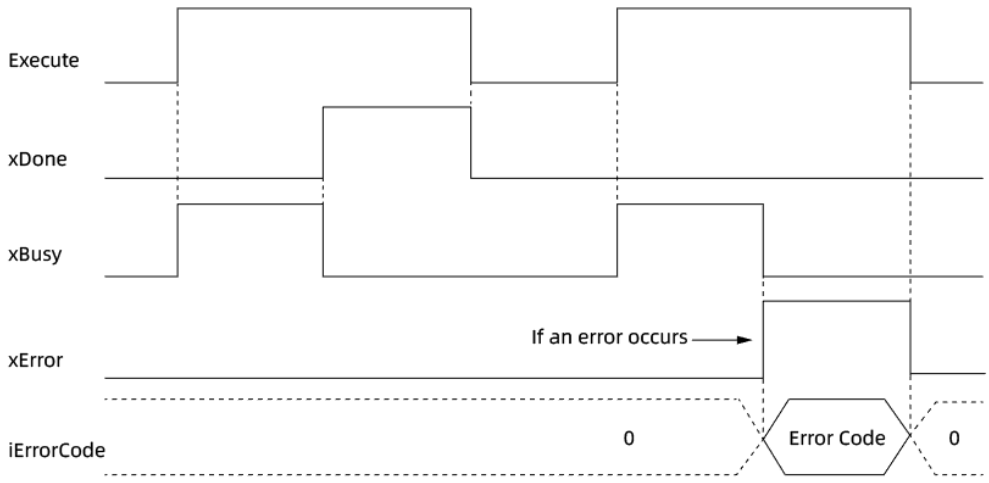
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	√	√	✓	-	-	-	-	-	-
S2	-	-	-	-	-	-	-	-	√
S3	-	-	-	-	-	-	-	-	√
S4	-	-	-	-	-	-	-	-	√

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S5	-	-	-	-	-	-	-	-	√
D1	√	√	√	-	-	-	-	-	-
D2	√	√	√	-	-	-	-	-	-
D3	√	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

Function and Instruction Description

The SFTP_GetFile instruction is used to obtain a file from the SFTP server. This instruction is executed once when the Execute signal is triggered. It establishes a connection with the server based on the information specified by stConnect. After that, it downloads a file strFileName from the server directory strServerDirName to the local directory strLocalDirName. After successful execution of the instruction, the xDone signal is set to TRUE. If the instruction fails to be executed, the xError signal is set to TURE and the instruction reports an error specified by iErrorCode.

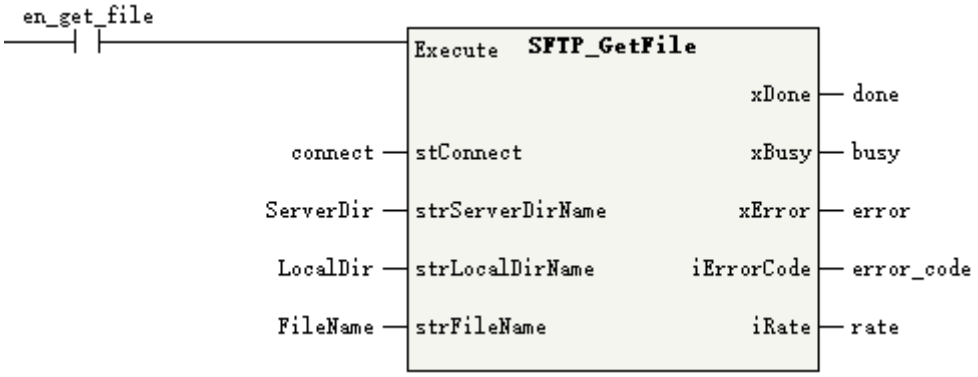
Timing Diagram



NOTE

- The local directory must be the directory of the SD card ("/" as the home directory to which the SD card is mounted). Otherwise, an error is reported.
- Before you use the SFTP_GetFile or SFTP_PutFile instruction, the SD card must be inserted.

Instruction Example



NOTE

Usage of this instruction in LiteST expression is the same as that in LD expression.

5.18.27 SFTP_PutFile

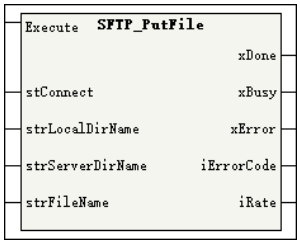
Version

- PLC: Easy series
  - The Easy 511 series supports local network ports but not expanded network ports.
  - Easy series models without Ethernet network ports do not support this instruction.
- Firmware version: V6.5.0.0 and later
- Software version: AutoShop V4.11.0.0 and later

Introduction

SFTP_PutFile -- Upload a file

The SFTP_PutFile instruction is used to upload a local file to the SFTP server.

Instruction	Name	LD Expression	LiteST Expression
SFTP_PutFile	Upload a file		SFTP_PutFile(Execute := ???, stConnect := ???, strLocalDirName := ???, strServerDirName := ???, strFileName := ???, xDone => , xBusy => , xError => , iErrorCode => , iRate =>);

The following table describes the instruction format.

16-bit Instruction	---			
32-bit Instruction	SFTP_PutFile			
Operand	Name	Description	Range	Data Type
S1	Execute	Triggered execution	ON or OFF	BOOL
S2	stConnect	Connection information	-	SFTP_CONNECT_INFO
S3	strLocalDirName	Local directory	-	STRING
S4	strServerDirName	Directory name on the server	-	STRING
S5	strFileName	Name of the file to be uploaded	-	STRING
D1	xDone	Completion flag*1	ON or OFF	BOOL
D2	xBusy	Executing*1	ON or OFF	BOOL
D3	xError	Error flag*1	ON or OFF	BOOL
D4	iErrorCode	Error code*1	-	INT16
D5	iRate	Download progress	0 to 100	INT16

The following table lists soft elements in the instruction.

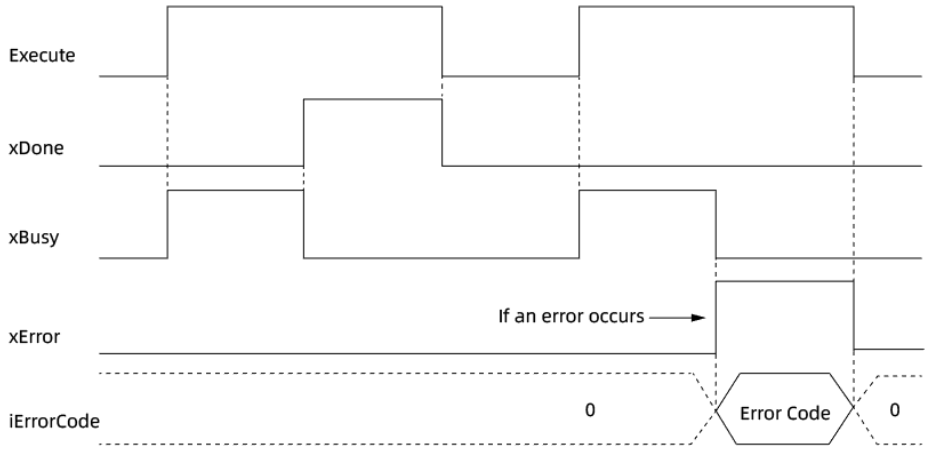
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	√	√	√	-	-	-	-	-	-
S2	-	-	-	-	-	-	-	-	√
S3	-	-	-	-	-	-	-	-	√
S4	-	-	-	-	-	-	-	-	√
S5	-	-	-	-	-	-	-	-	√
D1	√	√	√	-	-	-	-	-	-
D2	√	√	√	-	-	-	-	-	-
D3	√	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D5	-	-	-	√	√	-	-	-	-

Function and Instruction Description

The SFTP_PutFile instruction is used to upload a local file to the SFTP server. This instruction is executed when the Execute signal is triggered. It establishes a connection with the server based on the information specified by stConnect. After that, it uploads a file strFileName from the local directory strLocalDirName to the server directory strServerDirName. After successful execution of the instruction, the xDone signal is set to TRUE. If the instruction fails to be executed, the xError signal is set to TURE and the instruction reports an error specified by iErrorCode.

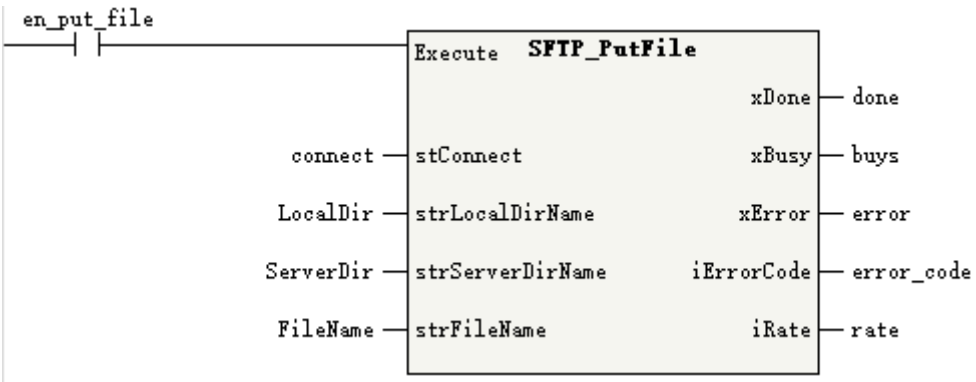
Timing Diagram



NOTE

- The local directory must be the directory of the SD card ("/" as the home directory to which the SD card is mounted). Otherwise, an error is reported.
- Before you use the SFTP_GetFile or SFTP_PutFile instruction, the SD card must be inserted.

Instruction Example



NOTE

Usage of this instruction in LiteST expression is the same as that in LD expression.

5.18.28 SFTP_GetFileList

Version

- PLC: Easy series
  - The Easy 511 series supports local network ports but not expanded network ports.
  - Easy series models without Ethernet network ports do not support this instruction.
- Firmware version: V6.5.0.0 and later
- Software version: AutoShop V4.11.0.0 and later
-

Introduction

SFTP_GetFileList -- Obtain a file list

The SFTP_GetFileList instruction is used to download all the files from the SFTP server directory.

Instruction	Name	LD Expression	LiteST Expression
SFTP_GetFileList	Obtain a file list		SFTP_GetFileList(Execute := ???, stConnect := ???, strDirName := ???, diFileListSize := ???, aFileList := ???, xDone => , xBusy => , xError => , iErrorCode => , iGetFileCount =>);

The following table describes the instruction format.

16-bit Instruction	---			
32-bit Instruction	SFTP_GetFileList			
Operand	Name	Description	Range	Data Type
S1	Execute	Triggered execution	ON or OFF	BOOL
S2	stConnect	Connection information	-	SFTP_CONNECT_INFO
S3	strDirName	Source directory name on the server	-	STRING

16-bit Instruction	---			
32-bit Instruction	SFTP_GetFileList			
Operand	Name	Description	Range	Data Type
S4	diFileListSize	Size of the file list aFileList	-	INT32
S5	aFileList	File list	-	SFTP_FILE_INFO
D1	xDone	Completion flag*1	ON or OFF	BOOL
D2	xBusy	Executing*1	ON or OFF	BOOL
D3	xError	Error flag*1	ON or OFF	BOOL
D4	iErrorCode	Error code*1	-	INT16
D5	iGetFileCount	Number of files obtained	-	INT16

The following table lists soft elements in the instruction.

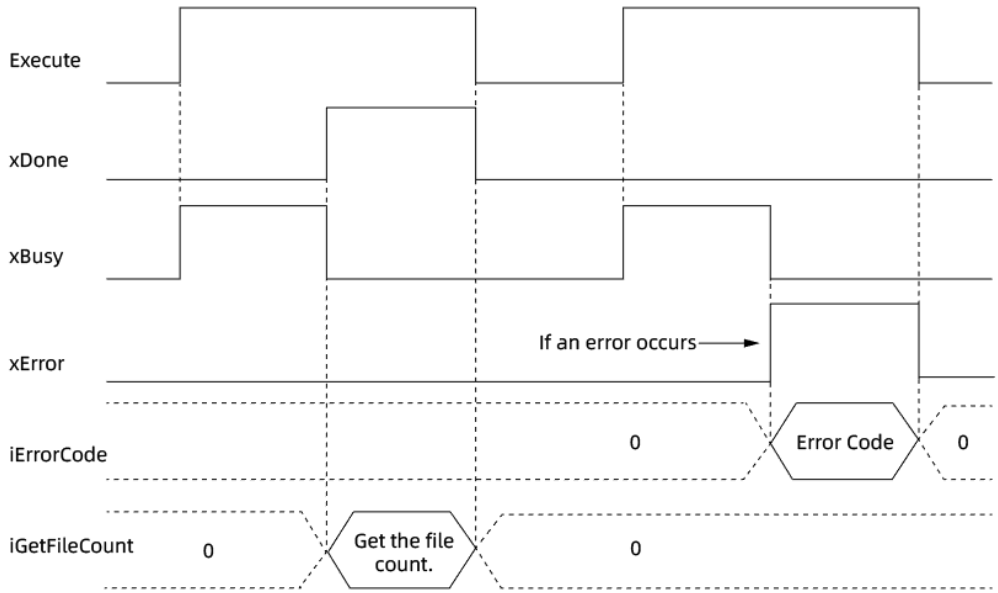
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	√	√	√	-	-	-	-	-	-
S2	-	-	-	-	-	-	-	-	√
S3	-	-	-	-	-	-	-	-	√
S4	-	-	-	√	√	-	√	√	-
S5	-	-	-	-	-	-	-	-	√
D1	√	√	√	-	-	-	-	-	-
D2	√	√	√	-	-	-	-	-	-
D3	√	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

Function and Instruction Description

The SFTP_GetFileList instruction is used to download all the files from the specified SFTP server directory. This instruction is executed once when the Execute signal is triggered. It obtains all the files from the server

directory strDirName and saves the files to the file list aFileList. After successful execution of the instruction, the xDone signal is set to TRUE. The instruction outputs the number of obtained files specified by iGetFileCount. If the instruction fails to be executed, the xError signal is set to TURE and the instruction reports an error specified by iErrorCode.

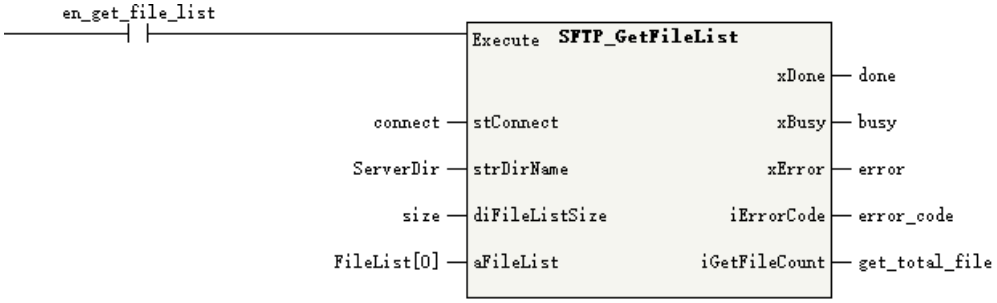
Timing Diagram



NOTE

The aFileList parameter is a structure of the _sSFTP_FILE_INF array type. The value of diFileListSize cannot be greater than the number of elements in the aFileList array. Otherwise, there is a risk of out-of-bounds during instruction execution.

Instruction Example



NOTE

Usage of this instruction in LiteST expression is the same as that in LD expression.

5.18.29 SFTP_DeleteFile

Version

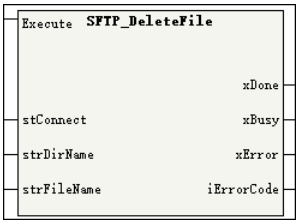
- PLC: Easy series
  - The Easy 511 series supports local network ports but not expanded network ports.

  - Easy series models without Ethernet network ports do not support this instruction.
- Firmware version: V6.5.0.0 and later
- Software version: AutoShop V4.11.0.0 and later
-

Introduction

SFTP_DeleteFile -- Delete a file or directory

The SFTP_DeleteFile instruction is used to delete a file or directory from the SFTP server.

Instruction	Name	LD Expression	LiteST Expression
SFTP_DeleteFile	Delete a file or directory		SFTP_DeleteFile(Execute := ???, stConnect := ???, strDirName := ???, strFileName := ???, xDone => , xBusy => , xError => , iErrorCode =>);

The following table describes the instruction format.

16-bit Instruction	---			
32-bit Instruction	SFTP_DeleteFile			
Operand	Name	Description	Range	Data Type
S1	Execute	Triggered execution	ON or OFF	BOOL
S2	stConnect	Connection information	-	SFTP_CONNECT_INFO
S3	strDirName	Name of the server directory where the file or directory to be deleted resides	-	STRING
S4	strFileName	Name of the file or directory to be deleted	-	STRING
D1	xDone	Completion flag*1	ON or OFF	BOOL
D2	xBusy	Executing*1	ON or OFF	BOOL
D3	xError	Error flag*1	ON or OFF	BOOL
D4	iErrorCode	Error code*1	-	INT16

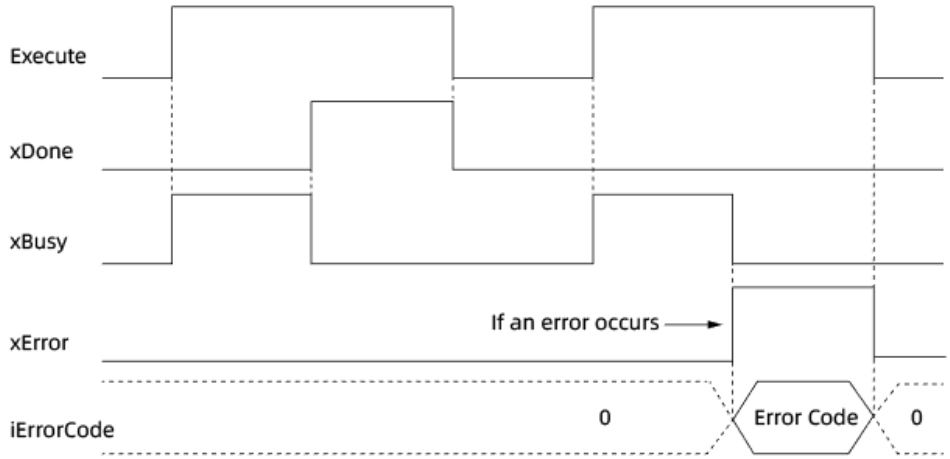
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	√	√	√	-	-	-	-	-	-
S2	-	-	-	-	-	-	-	-	√
S3	-	-	-	-	-	-	-	-	√
S4	-	-	-	-	-	-	-	-	√
D1	√	√	√	-	-	-	-	-	-
D2	√	√	√	-	-	-	-	-	-
D3	√	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-

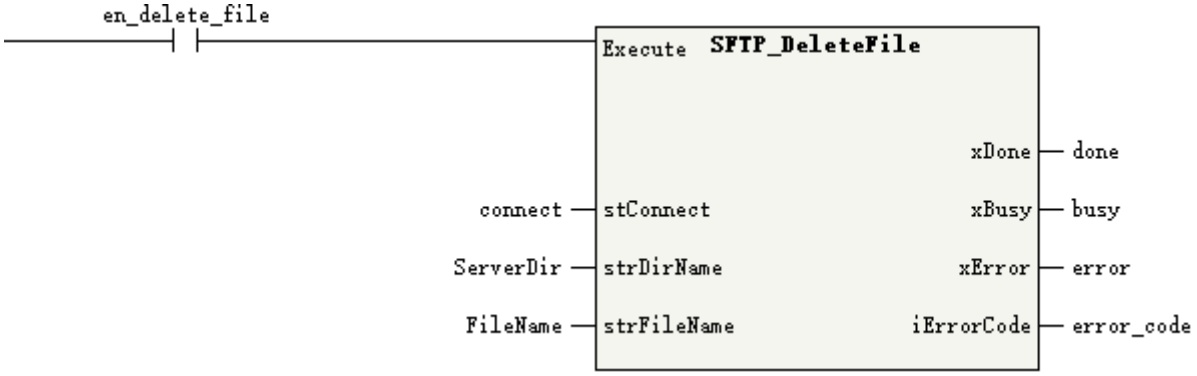
Function and Instruction Description

The SFTP_DeleteFile instruction is used to delete a file or directory from the SFTP server. This instruction is executed once when the Execute signal is triggered. It establishes a connection with the server based on the information specified by stConnect. After that, it deletes a file or directory strFileName from the server directory strDirName. After successful execution of the instruction, the xDone signal is set to TRUE. If the instruction fails to be executed, the xError signal is set to TURE and the instruction reports an error specified by iErrorCode.

Timing Diagram



Instruction Example



NOTE

Usage of this instruction in LiteST expression is the same as that in LD expression.

5.18.30 SFTP_MoveFile

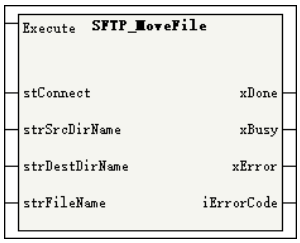
Version

- PLC: Easy series
  - The Easy 511 series supports local network ports but not expanded network ports.
  - Easy series models without Ethernet network ports do not support this instruction.
- Firmware version: V6.5.0.0 and later
- Software version: AutoShop V4.11.0.0 and later

Introduction

SFTP_MoveFile -- Move a file or directory

The SFTP_MoveFile instruction is used to move a file or directory on the SFTP server.

Instruction	Name	LD Expression	LiteST Expression
SFTP_MoveFile	Move a file or directory		SFTP_MoveFile(Execute := ???, stConnect := ???, strSrcDirName := ???, strDestDirName := ???, strFileName := ???, xDone => , xBusy => , xError => , iErrorCode =>);

The following table describes the instruction format.

16-bit Instruction	---			
32-bit Instruction	SFTP_MoveFile			
Operand	Name	Description	Range	Data Type
S1	Execute	Triggered execution	ON or OFF	BOOL
S2	stConnect	Connection information	-	SFTP_CONNECT_INFO
S3	strSrcDirName	Source directory name on the server	-	STRING
S4	strDestDirName	Target directory name on the server	-	STRING
S5	strFileName	Name of the file or directory to be moved	-	STRING
D1	xDone	Completion flag*1	ON or OFF	BOOL
D2	xBusy	Executing*1	ON or OFF	BOOL
D3	xError	Error flag*1	ON or OFF	BOOL
D4	iErrorCode	Error code*1	-	INT16

The following table lists soft elements in the instruction.

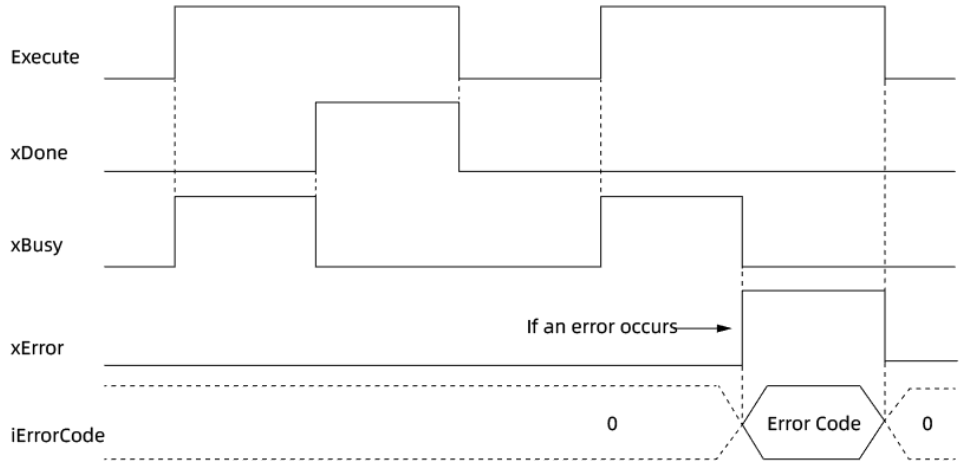
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	√	√	√	-	-	-	-	-	-
S2	-	-	-	-	-	-	-	-	√
S3	-	-	-	-	-	-	-	-	√
S4	-	-	-	-	-	-	-	-	√
S5									√
D1	√	√	√	-	-	-	-	-	-
D2	√	√	√	-	-	-	-	-	-
D3	√	√	√	-	-	-	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D4	-	-	-	√	√	-	-	-	-

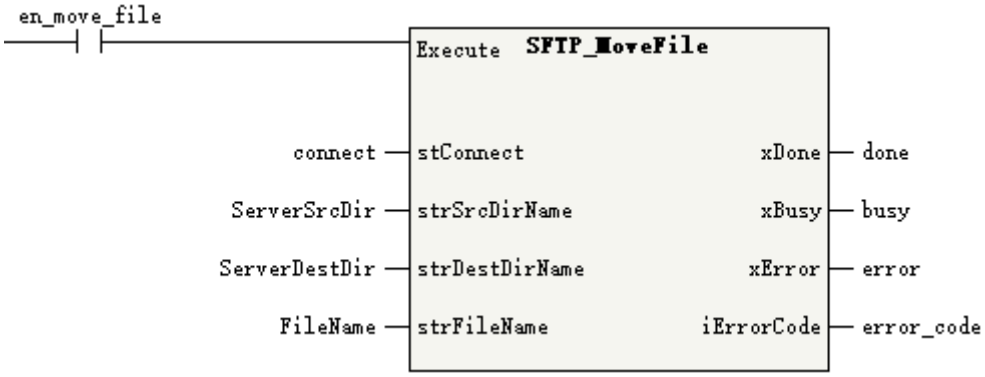
Function and Instruction Description

The SFTP_MoveFile instruction is used to move a file or directory on the SFTP server to the designated directory. This instruction is executed when the Execute signal is triggered. It establishes a connection with the server based on the information specified by stConnect. After that, it moves a file or directory strFileName from the source server directory strSrcDirName to the target directory strDestDirName. After successful execution of the instruction, the xDone signal is set to TRUE. If the instruction fails to be executed, the xError signal is set to TURE and the instruction reports an error specified by iErrorCode.

Timing Diagram



Instruction Example



NOTE

Usage of this instruction in LiteST expression is the same as that in LD expression.

5.18.31 SFTP_CreateDir

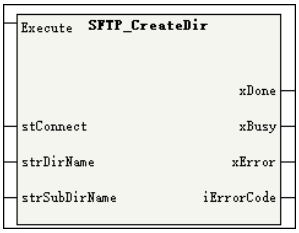
Version

- PLC: Easy series
  - The Easy 511 series supports local network ports but not expanded network ports.
  - Easy series models without Ethernet network ports do not support this instruction.
- Firmware version: V6.5.0.0 and later
- Software version: AutoShop V4.11.0.0 and later
-

Introduction

SFTP_CreateDir -- Create a directory

The SFTP_CreateDir instruction is used to create a directory on the SFTP server.

Instruction	Name	LD Expression	LiteST Expression
SFTP_CreateDir	Create a directory		SFTP_CreateDir(Execute := ???, stConnect := ???, strDirName := ???, strSubDirName := ???, xDone => , xBusy => , xError => , iErrorCode =>);

The following table describes the instruction format.

16-bit Instruction	-			
32-bit Instruction	SFTP_CreateDir			
Operand	Name	Description	Range	Data Type
S1	Execute	Triggered execution	ON or OFF	BOOL
S2	stConnect	Connection information	-	SFTP_CONNECT_INFO
S3	strDirName	Directory name on the server	-	STRING
S4	strSubDirName	Subdirectory name on the server to be created	-	STRING
D1	xDone	Completion flag*1	ON or OFF	BOOL

16-bit Instruction	-			
32-bit Instruction	SFTP_CreateDir			
Operand	Name	Description	Range	Data Type
D2	xBusy	Executing*1	ON or OFF	BOOL
D3	xError	Error flag*1	ON or OFF	BOOL
D4	iErrorCode	Error code*1	-	INT16

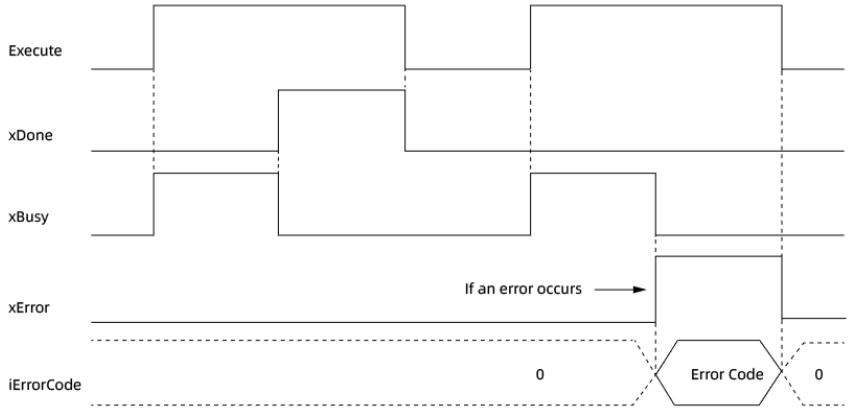
The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	√	√	√	-	-	-	-	-	-
S2	-	-	-	-	-	-	-	-	√
S3	-	-	-	-	-	-	-	-	√
S4	-	-	-	-	-	-	-	-	√
D1	√	√	√	-	-	-	-	-	-
D2	√	√	√	-	-	-	-	-	-
D3	√	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-

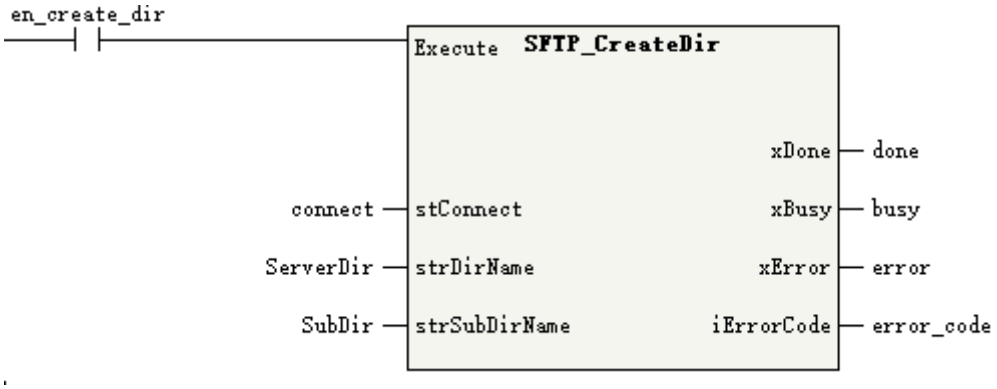
Function and Instruction Description

The SFTP_CreateDir instruction is used to create a directory on the SFTP server. This instruction is executed once when the Execute signal is triggered. It establishes a connection with the server based on the information specified by stConnect. After that, it creates a subdirectory strSubDirName in the server directory strDirName. After successful execution of the instruction, the xDone signal is set to TRUE. If the instruction fails to be executed, the xError signal is set to TURE and the instruction reports an error specified by iErrorCode.

Timing Diagram



Instruction Example



> NOTE
>
> Usage of this instruction in LiteST expression is the same as that in LD expression.

5.18.32 SFTP_RenameFile

Version

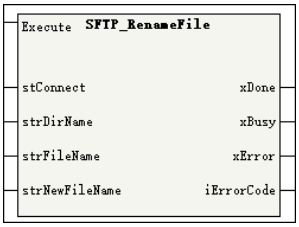
- PLC: Easy series
  - The Easy 511 series supports local network ports but not expanded network ports.
  - Easy series models without Ethernet network ports do not support this instruction.
- Firmware version: V6.5.0.0 and later
- Software version: AutoShop V4.11.0.0 and later

Introduction

SFTP_RenameFile -- Rename a file on the SFTP server

The SFTP_RenameFile instruction is used to rename a file on the SFTP server.

Instruction	Name	LD Expression	LiteST Expression
SFTP_RenameFile	Rename a file or directory		SFTP_RenameFile(Execute := ???, stConnect := ???, strDirName := ???,

Instruction	Name	LD Expression	LiteST Expression
			strFileName := ???, strNewFileName := ???, xDone => , xBusy => , xError => , iErrorCode =>);

The following table describes the instruction format.

16-bit Instruction	---			
32-bit Instruction	SFTP_RenameFile			
Operand	Name	Description	Range	Data Type
S1	Execute	Triggered execution	ON or OFF	BOOL
S2	stConnect	Connection information	-	SFTP_CONNECT_INFO
S3	strDirName	Directory name on the server	-	STRING
S4	strFileName	Original name of the file or directory	-	STRING
S5	strNewFileName	New name of the file or directory	-	STRING
D1	xDone	Completion flag*1	ON or OFF	BOOL
D2	xBusy	Executing*1	ON or OFF	BOOL
D3	xError	Error flag*1	ON or OFF	BOOL
D4	iErrorCode	Error code*1	-	INT16

The following table lists soft elements in the instruction.

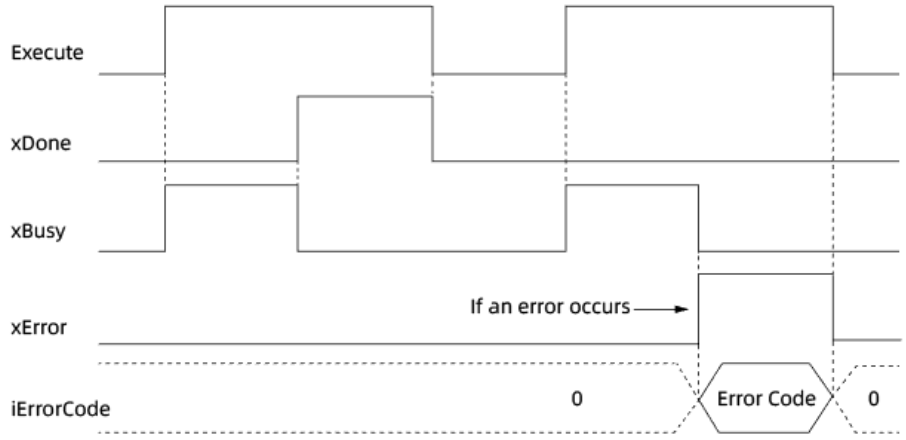
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	√	√	√	-	-	-	-	-	-
S2	-	-	-	-	-	-	-	-	√
S3	-	-	-	-	-	-	-	-	√

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S4	-	-	-	-	-	-	-	-	√
S5									√
D1	√	√	√	-	-	-	-	-	-
D2	√	√	√	-	-	-	-	-	-
D3	√	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-

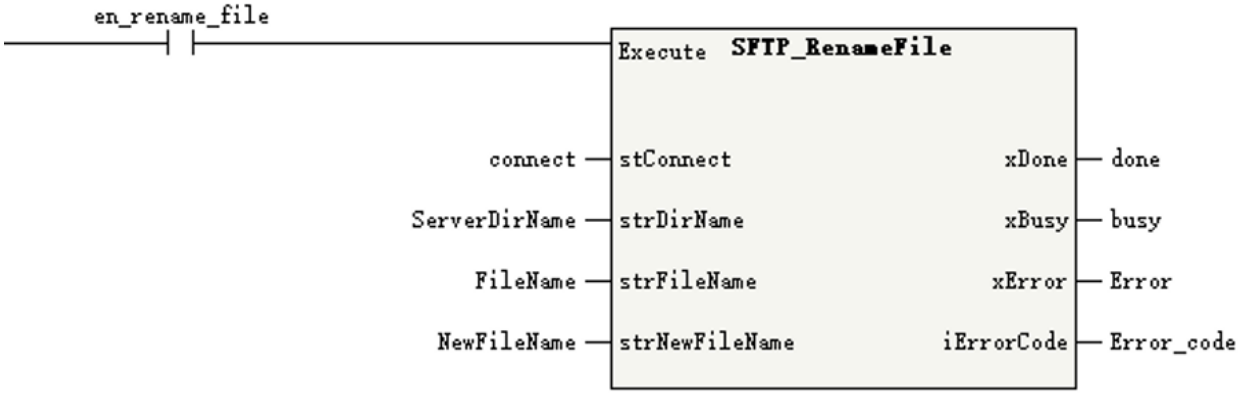
Function and Instruction Description

The SFTP_RenameFile instruction is used to rename a file or directory on the SFTP server. This instruction is executed when the Execute signal is triggered. It establishes a connection with the server based on the information specified by stConnect. After that, it changes the name of a file or directory on the server directory strDirName from strFileName to strNewFileName. After successful execution of the instruction, the xDone signal is set to TRUE. If the instruction fails to be executed, the xError signal is set to TURE and the instruction reports an error specified by iErrorCode.

Timing Diagram



Instruction Example



NOTE

Usage of this instruction in LiteST expression is the same as that in LD expression.

5.18.33 Error Codes of SFTP Instructions

The following table lists the error codes of SFTP instructions.

Error	Description
0	No error.
1	Internal error.
2	The task is not started on the SFTP client.
3	The SD card is not inserted.
4	The available space of the SD card is insufficient.
5	The directory is not supported.
20	Invalid IP address.
21	Invalid port number.
22	Invalid account.
23	Invalid password.
40	Invalid path.
41	Invalid file name.
42	Reserved
43	Invalid file list size.
1001	Internal error.
1002	Failed to create a session.

Error	Description
1003	Connection timeout.
1004	Connection error.
1005	Protocol mismatch.
1006	The server does not support password authentication.
1007	Authentication failed.
1008	Internal error.
1009	Internal error.
1030	Invalid local path.
1031	Invalid server path.
1032	Failed to transfer the file.
1033	Invalid file type.
1034	Failed to move the file.
1035	Failed to delete the file.
1036	Failed to create a directory.
1037	No permission.
1038	No available space.

5.18.34 ETC_ReadParameter_CoE

Description

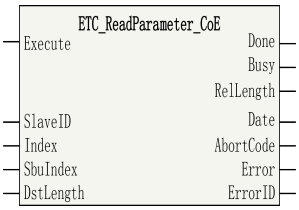
ETC_ReadParameter_CoE: Reading SDO parameters of the slave



As the CoE service of the slave is established in preop(2) state, it is recommended that the instruction be used after the slave enters preop(2) state.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ETC_ReadParameter_CoE	Reading SDO parameters of EtherCAT slave		ETC_ReadParameter_CoE(Execute := ???,

Instruction	Name	LD Expression	LiteST Expression
			SlaveID := ???, Index := ???, SubIndex := ???, DstLength := ???, Done => , Busy => , RelLength => , Data => , AbortCode => , Error => , ErrorID =>);

CAUTION

As the CoE service of the slave is established in preop(2) state, it is recommended that the instruction be used after the slave enters preop(2) state.

The instruction formats are as follows.

16-bit instruction	-			
32-bit instruction	ETC_ReadParameter_CoE: Continuous execution			
Operands	Name	Description	Value range	Data type
S1	SlaveID	Slave address	0 to 71	INT
S2	Index	Index	-	INT
S3	SubIndex	Sub-index	-	INT
S4	DstLength	Target length	1 to 4	INT
D1	Done	Completion flag	-	BOOL
D2	Busy	Busy flag	-	BOOL
D3	RelLength	Actual length of the read data, in byte	1 to 4	INT
D4	data	Read data	-	DINT
D5	AbortCode	Abortion code when reading slave object dictionary fails*1	-	DINT
D6	Error	Error flag	-	BOOL
D7	ErrorID	Fault code*2	-	INT

NOTE

- *1: For details, see "SDO AbortCode" in Instruction Codes.
- *2: For details, see "Instruction Fault Code" in 5.18.34 ETC_ReadParameter_CoE.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-
D6	√[1]	√	√	-	-	√	-	-	-
D7	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

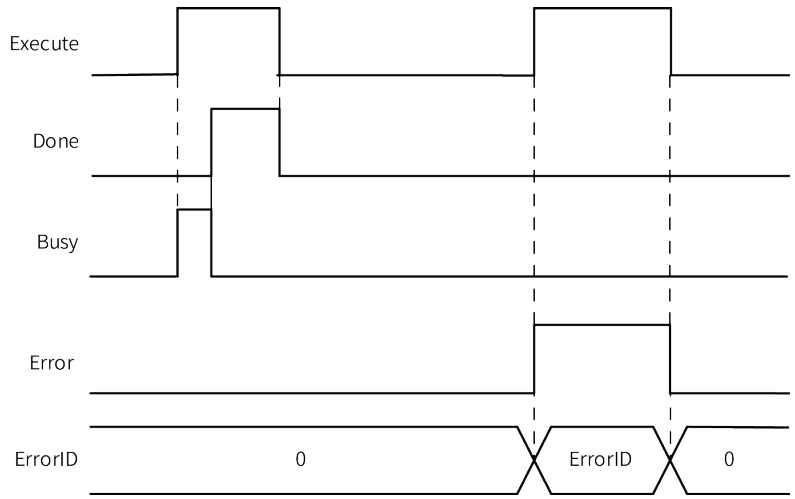
Function

This instruction is used to read the object dictionary of an EtherCAT slave. It is active on the rising edge.

- SlaveID specifies the configuration address of the EtherCAT slave.
- On the rising edge of Execute, the instruction latches the input parameters on the left and triggers reading the object dictionary specified by Index and SubIndex.
- DstLength specifies the length (in byte) of the object dictionary to read.

- When reading is successful, the Done signal becomes active. Dtate displays the read value, and RelLength displays the actual length of the read object dictionary. When reading fails, the Error output becomes active, and you can determine the cause of the failure by checking AbortCode and ErrorID.
- The parameter Data in this instruction is a DINT type parameter that occupies 4 bytes. When the read object dictionary is SINT or INT data, the reading result is stored in the low-order 8 bits or 16 bits of Data, and the unused high-order 24 bits or 16 bits are filled with 0s. For example, when the read data is the SINT-type or INT-type -8, the data actually stored in Data is 0x000000f8 or 0x0000fff8.

Sequence Diagram



Fault Code

Fault code	Cause	Solution
8001	Failed to configure the master.	Check whether the master configuration parameters are appropriate.
8002	Failed to configure the slave.	Check whether the slave configuration parameters are appropriate.
8003	Reserved	-
8200	Failed to write the slave startup parameters to the SDO.	Check whether the SDO of the startup parameter list is appropriate.

5.18.35 ETC_WriteParameter_CoE

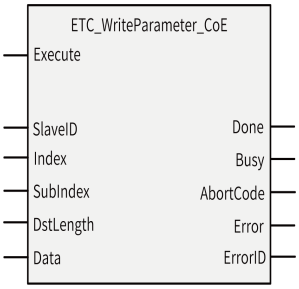
Description

ETC_WriteParameter_CoE: Writing SDO parameters of the slave

CAUTION

Since the CoE services of the slave become available only after the preoperational state (state 2) is reached, it is recommended to use this instruction when the slave is in preoperational state (state 2) or later states.

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ETC_WriteParameter_CoE	Writing SDO parameters of EtherCAT slave		ETC_WriteParameter_CoE(Execute := ???, SlaveID := ???, Index := ???, SubIndex := ???, DstLength := ???, Data := ???, Done => , Busy => , AbortCode => , Error => , ErrorID =>);

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	ETC_WriteParameter_CoE: Continuous execution

Operands	Name	Description	Value range	Data type
S1	SlaveID	Slave ID (Only the configuration address of the slave is allowed.)	0 to 71	INT
S2	Index	Index	-	INT
S3	SubIndex	Sub-index	-	INT
S4	DstLength	Target length	1 to 4	INT
S5	Data	Target data	-	DINT
D1	Done	Completion flag	-	BOOL
D2	Busy	Busy flag	-	BOOL
D3	AbortCode	Abortion code	-	DINT
D4	Error	Error flag	-	BOOL
D5	ErrorID	Fault code*1	-	INT

NOTE

*1: For details, see Instruction Codes.

The following table lists the abortion codes.

Abortion code	Description
05030000	Trigger bits not alternated
05040000	SDO protocol timeout
05040001	Invalid or unknown client/server command word
05040005	Out of memory
06010000	Access to objects not supported
06010001	Reading a write-only object
06010002	Writing to a read-only object
06020000	Object not existed in the object dictionary
06040041	The object cannot be mapped to the PDO.
06040042	Number and length of the objects to be mapped exceeding PDO length
06040043	General parameters incompatible
06040047	General device incompatible internally
06060000	Access to object fails due to a hardware error
06070010	Data type mismatch: service parameter length mismatch
06070012	Data type not match, service parameter length too long
06070013	Data type not match, service parameter length too short
06090011	Sub-index nonexistent
06090030	Invalid parameter value
06090031	Parameter value written too large
06090032	Parameter value written too small
06090036	Maximum value is less than minimum value.
08000000	General error
08000020	Data cannot be transmitted or stored to the application.

Abortion code	Description
08000021	Data cannot be transmitted or stored to the application due to local control.
08000022	Data cannot be transmitted or stored to the application due to current device status.
08000023	Object dictionary error occurred or object dictionary not existed

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

NOTE

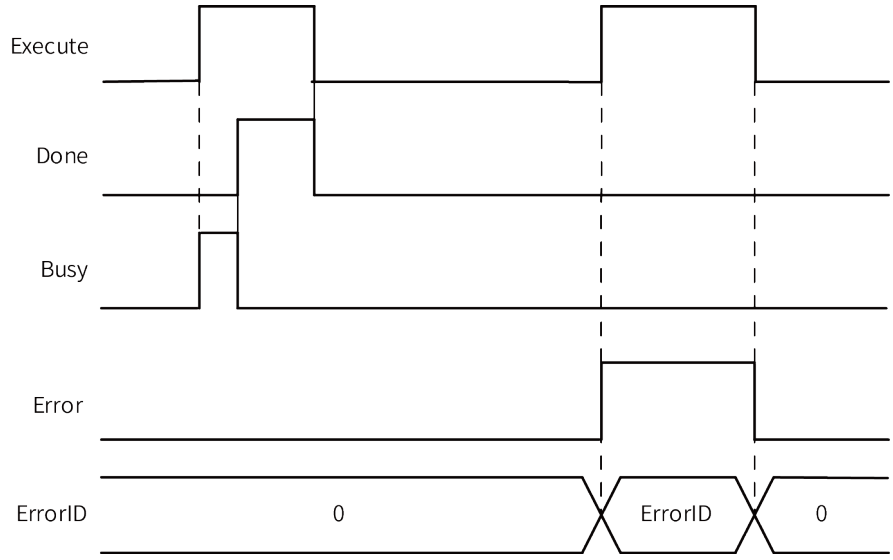
[1] The X element is not supported.

Function and Instruction Description

This instruction is used to write to the object dictionary of an EtherCAT slave. It is active on the rising edge.

- SlaveID specifies the configuration address of the EtherCAT slave.
- On the rising edge of Execute, the instruction latches the input parameters on the left and writes the data in Data to the object dictionary specified by Index and SubIndex.
- DstLength specifies the length (in byte) of the object dictionary to write to.
- When writing is successful, the Done signal becomes active. When writing fails, the Error output becomes active, and you can determine the cause of the failure by checking AbortCode and ErrorID.

Sequence Diagram



Fault Code

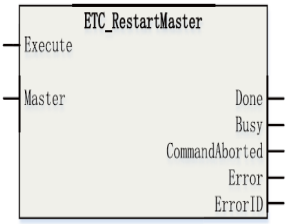
Fault code	Cause	Solution
8001	Failed to configure the master.	Check whether the master configuration parameters are appropriate.
8002	Failed to configure the slave.	Check whether the slave configuration parameters are appropriate.
8003	Reserved	-
8200	Failed to write the slave startup parameters to the SDO.	Check whether the SDO of the startup parameter list is appropriate.

5.18.36 ETC_RestartMaster

Introduction

ETC_RestartMaster -- Restart EtherCAT master

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
ETC_RestartMaster	Restart EtherCAT master		ETC_RestartMaster(Execute := ???, Master := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

16-bit Instruction	ETC_RestartMaster: Continuous execution
32-bit Instruction	-

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Master	EtherCAT master	Yes	-	-	-
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON or OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D5	ErrorID	Error code*1	Yes	0	-	INT16

NOTE

*1: For details, see Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

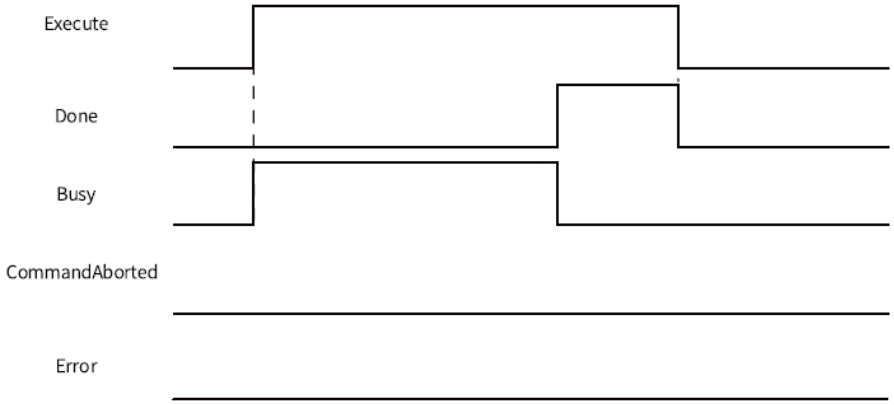
> **NOTE**
>
> [1]: The X element is not supported.

Function and Instruction Description

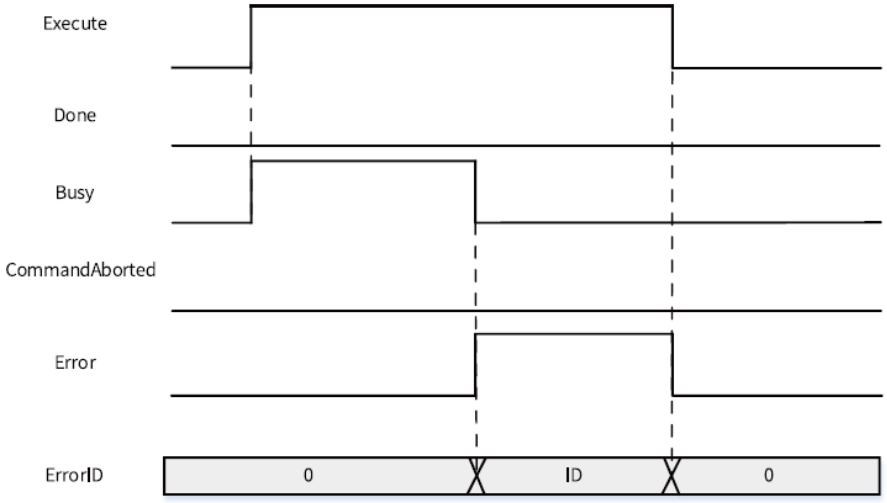
The ETC_RestartMaster instruction restarts the EtherCAT bus.

Timing Diagram

- If the bus is restarted successfully, the Done output becomes active.

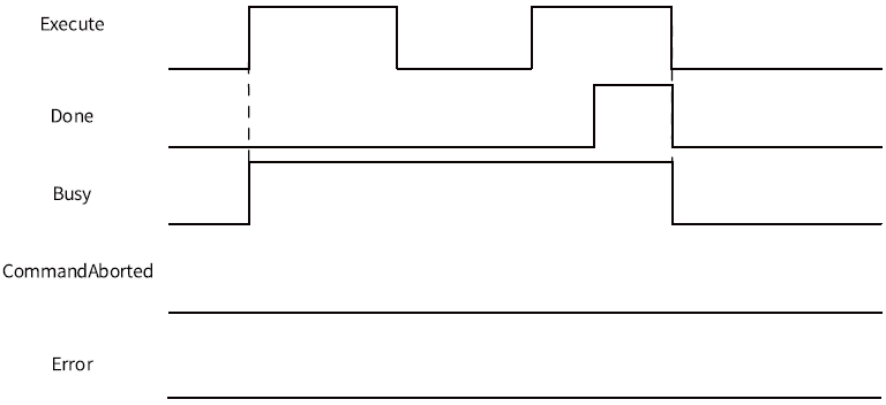


- If the bus fails to restart, the Error output becomes active, and ErrorID indicates the error code.



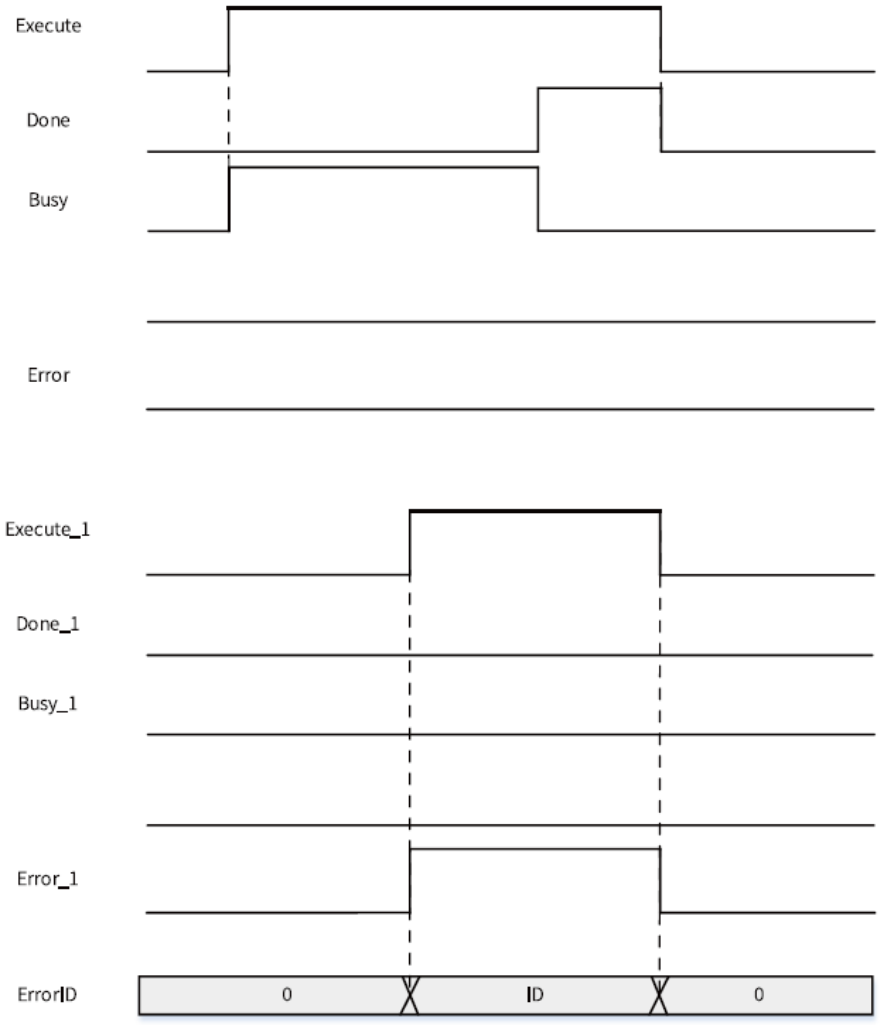
- Re-execution

  The same restart instruction can be re-triggered.



- Multi-execution

This instruction does not support multi-execution. If a second instruction is triggered during execution of the first one, the second instruction reports an error, and the first instruction continues execution.



5.18.37 Instruction Code Information

SDO AbortCode

Value	Description
0x05 03 00 00	Toggle bit not alternated
0x05 04 00 00	SDO protocol timed out.
0x05 04 00 01	Client/server command specifier not valid or unknown
0x05 04 00 05	Out of memory
0x06 01 00 00	Unsupported access to an object
0x06 01 00 01	Attempt to read a write only object.
0x06 01 00 02	Attempt to write a read only object.
0x06 02 00 00	Object does not exist in the object dictionary.

Value	Description
0x06 04 00 41	Object cannot be mapped to the PDO.
0x06 04 00 42	The number and length of the objects to be mapped would exceed PDO length.
0x06 04 00 43	General parameter incompatibility reason
0x06 04 00 47	General internal incompatibility in the device.
0x06 06 00 00	Access failed due to a hardware error.
0x06 07 00 10	Data type does not match; length of service parameter does not match.
0x06 07 00 12	Data type does not match; length of service parameter too high.
0x06 07 00 13	Data type does not match; length of service parameter too low.
0x06 09 00 11	Sub-index does not exist.
0x06 09 00 30	Value range of parameter exceeded (only for write access).
0x06 09 00 31	Value of parameter written too high
0x06 09 00 32	Value of parameter written too low
0x06 09 00 36	Maximum value is less than minimum value.
0x08 00 00 00	General error
0x08 00 00 20	Data cannot be transferred or stored to the application.
0x08 00 00 21	Data cannot be transferred or stored to the application because of local control.
0x08 00 00 22	Data cannot be transferred or stored to the application because of the present device state.
0x08 00 00 23	Object dictionary dynamic generation fails or no object dictionary is present.

Instruction Fault Codes

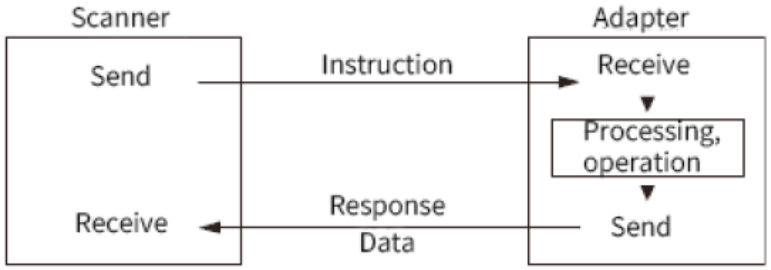
Fault code	Cause	Solution
0x0500	The master is not found.	Check whether EtherCAT bus communication is enabled.
0x0501	The slave is not found.	Check whether the slave exists in configuration.
0x0502	The length of the SDO to read or write to is 0 or greater than 4.	Check whether the length specified in the SDO read or write function is correct.
0x0503	The master is not found.	Check whether the master configuration parameters are correct.

Fault code	Cause	Solution
0x0504	Reading or writing fails. 1. The SDO read or write operation times out. 2. The SDO does not exist. 3. Reading or writing to the SDO is not allowed by the slave state. 4. The length of the SDO to read or write to is incorrect.	Check whether the SDO operation is allowed by the slave state machine. Check whether the SDO to read or write to exists. Check whether the length of the SDO to read or write to is correct.
0x0505	Failed to request the memory.	1. Check whether the PLC memory runs out. 2. Contact the manufacturer.
0x0506	The master is in Stopping state.	Do not call this instruction when the master is in Stopping state.

5.18.38 EIP_Generic_Service

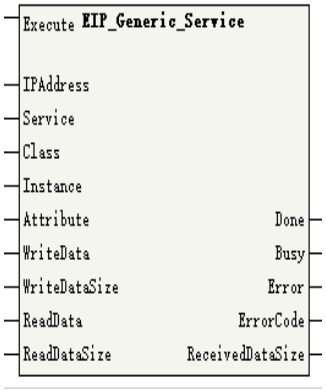
Description

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic block

Instruction	Name	LD Expression	LiteST Expression
EIP_Generic_Service	Calling the “Generic” service of a specific instance of the EtherNet/IP object		-

Instruction	Name	LD Expression	LiteST Expression
			

For instruction formats, see the following table.

16-bit instruction	-
32-bit instruction	EIP_Generic_Service: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Service	Service code	No	-	0 to 255	INT
S3	Class	Class code	No	-	-	DINT
S4	Instance	Instance code	No	-	-	DINT
S5	Attribute	Attribute code	No	-	-	DINT
S6	WriteData	Written data buffer area	No	-	-	BYTE[]/INT[]
S7	WriteDataSize	Written data size (in bytes)	No	-	0 to 494	INT
S8	ReadData	Read data buffer area	No	-	-	BYTE[]/INT[]
S9	ReadDataSize	Read data size (in bytes)	No	-	0 to 494	INT

D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT
D5	ReceivedDataSize	Received data size	Yes	0	0-1502	INT

NOTE

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	-	-	-	-
S7	-	-	-	√	√	√	√	-	-
S8	-	-	-	√	√	-	-	-	-
S9	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

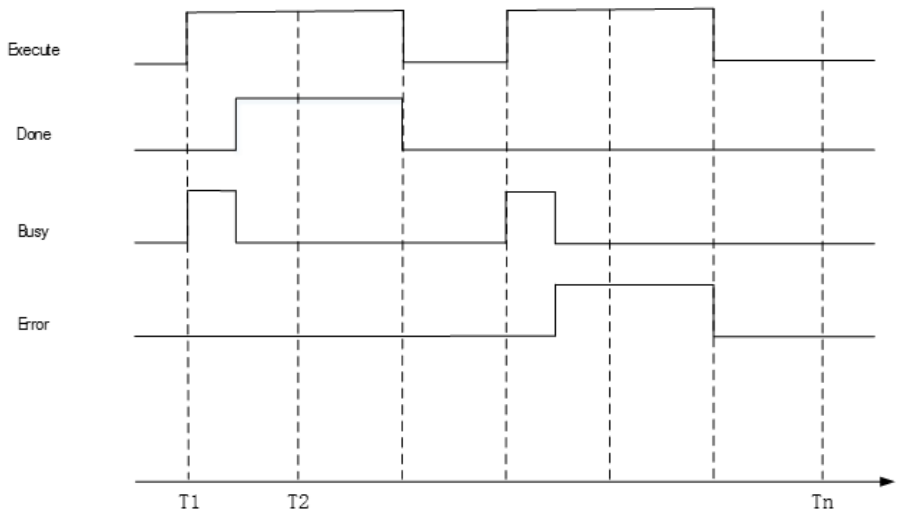
NOTE

[1] The X element is not supported.

Function

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values, and the ReadData buffer area is cleared.
- When execution of the instruction is successful, Done is set to ON, the size of the service response data is saved to ReceivedDataSize, and the service response data is written to the ReadData buffer area.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

Sequence Diagram

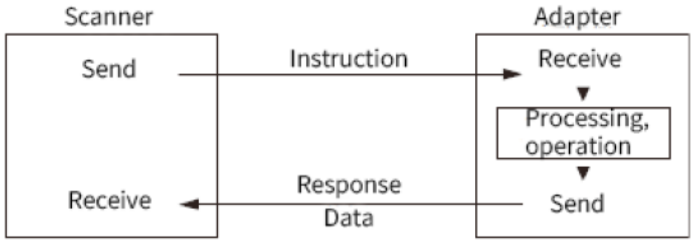


5.18.39 EIP_Get_Attributes_All

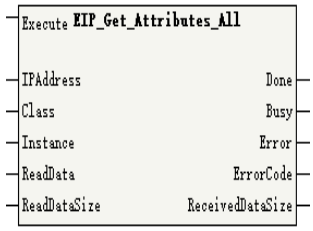
Description

EIP_Get_Attributes_All: Calling the “Get_Attributes_All” service for a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block

Instruction	Name	LD Expression	LiteST Expression
EIP_Get_Attributes_All	Calling the "Get_Attributes_All" service for a specific instance of the EtherNet/IP object		-

The list of instructions is shown below.

16-bit instruction	-					
32-bit instruction	EIP_Get_Attributes_All: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
S4	ReadData	Read data buffer area	No	-	-	BYTE[]/INT[]
S5	ReadDataSize	Read data size (in bytes)	No	-	0 to 494	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL

D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT
D5	ReceivedDataSize	Received data size	Yes	0	0 to 494	INT

NOTE

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress must be set to 192.168.1.88.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	-	-	-	-
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

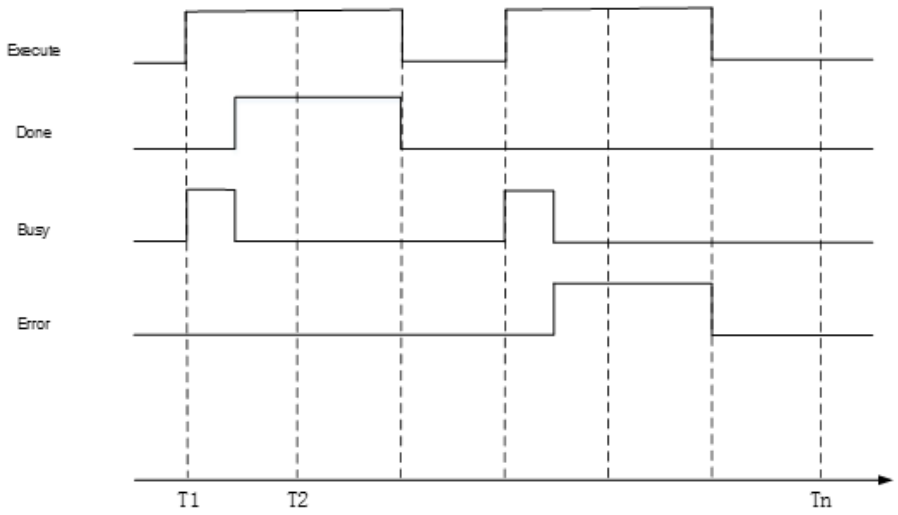
NOTE

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values, and the ReadData buffer area is cleared.
- When execution of the instruction is successful, Done is set to ON, the size of the service response data is saved to ReceivedDataSize, and the service response data is written to the ReadData buffer area.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

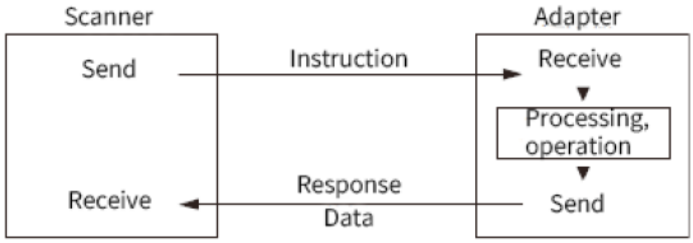
Sequence Diagram



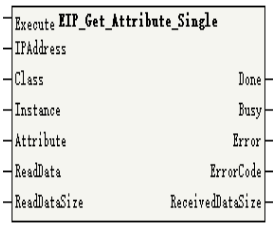
5.18.40 EIP_Get_Attribute_Single

Description

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block

Instruction	Name	LD Expression	LiteST Expression
EIP_Get_Attribute_Single	Calling the “Get_Attribute_Single” service for a specific instance of the EtherNet/IP object		-

The list of instructions is shown as follows.

16-bit instruction	-					
32-bit instruction	EIP_Get_Attribute_Single: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
S4	Attribute	Attribute code	No	-	-	DINT
S5	ReadData	Read data buffer area	No	-	-	BYTE[]/INT[]
S6	ReadDataSize	Read data size (in bytes)	No	-	0 to 494	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT
D5	ReceivedDataSize	Received data size	Yes	0	0 to 494	INT

NOTE

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress must be set to 192.168.1.88.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	-	-	-	-
S6	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

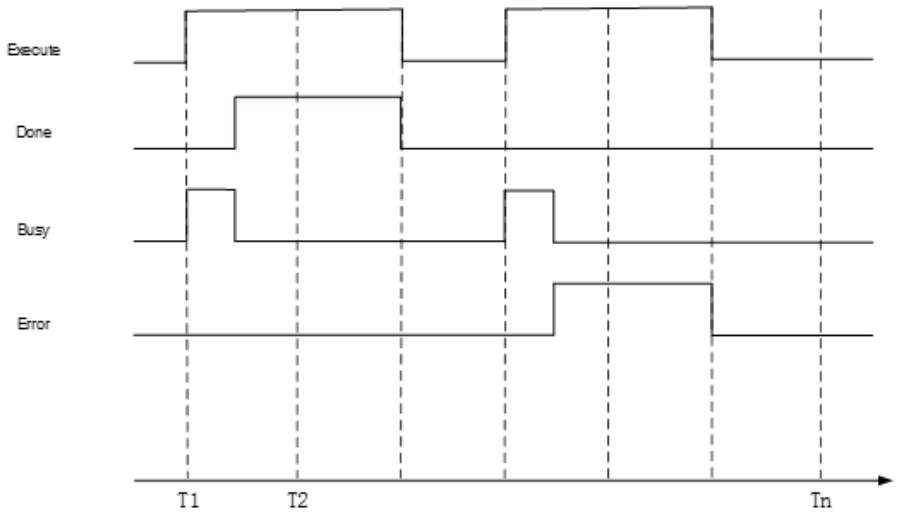
NOTE

[1] The X element is not supported.

Function

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values, and the ReadData buffer area is cleared.
- When execution of the instruction is successful, Done is set to ON, the size of the service response data is saved to ReceivedDataSize, and the service response data is written to the ReadData buffer area.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

Sequence Diagram

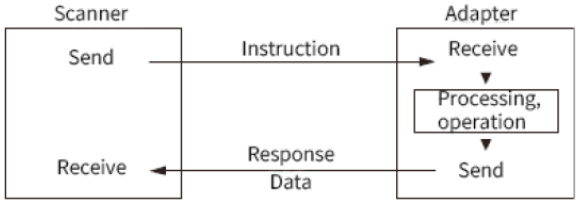


5.18.41 EIP_Set_Attributes_All

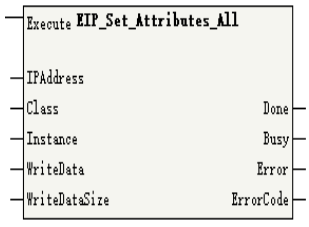
Description

EIP_Get_Attribute_Single: Calling the "Get_Attribute_Single" service for a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block

Instruction	Name	LD Expression	LiteST Expression
EIP_Set_Attributes_All	Calling the "Set_Attributes_All" service for a specific instance of the EtherNet/IP object		-

The list of instructions is shown below.

16-bit instruction	-
	EIP_Set_Attributes_All: Continuous execution

32-bit instruction						
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
S4	WriteData	Written data buffer area	No	-	-	BYTE[]/INT[]
S5	WriteDataSize	Written data size (in bytes)	No	-	0 to 1502	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

NOTE

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress must be set to 192.168.1.88.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	-	-	-	-

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

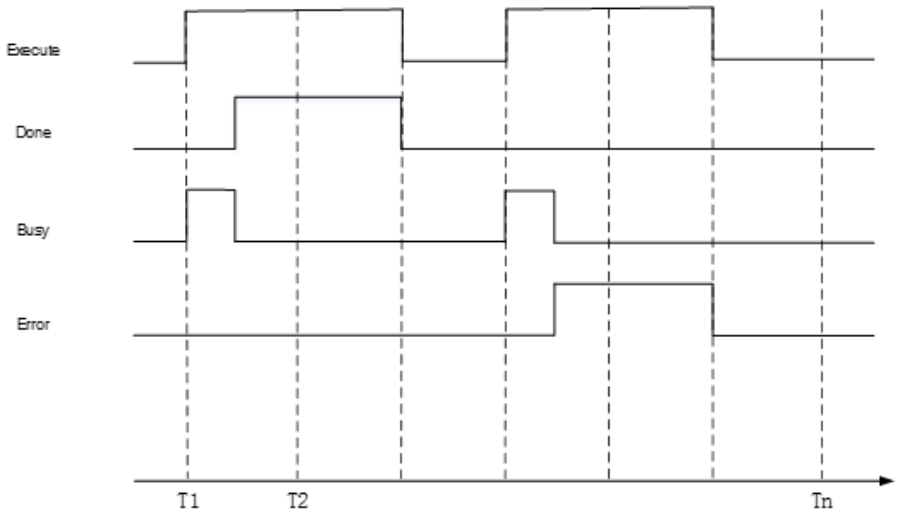
NOTE

[1] The X element is not supported.

Function

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

Sequence Diagram

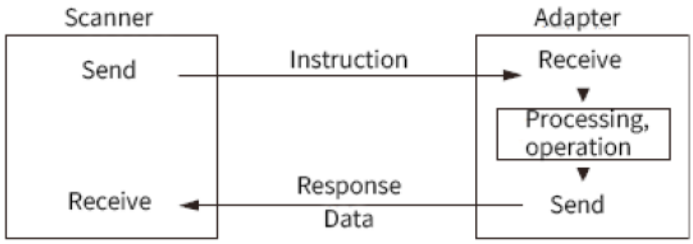


5.18.42 EIP_Set_Attribute_Single

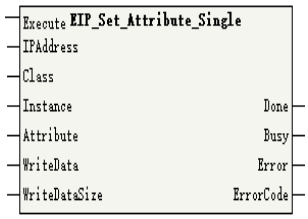
Description

EIP_Set_Attributes_All: Calling the "Set_Attributes_All" service for a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block

Instruction	Name	LD Expression	LiteST Expression
EIP_Set_Attribute_Single	Calling the "Set_Attribute_Single" service for a specific instance of the EtherNet/IP object		-

The list of instructions is shown as follows.

16-bit instruction	-					
32-bit instruction	EIP_Set_Attribute_Single: Continuous execution					
Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
S4	Attribute	Attribute code	No	-	-	DINT
S5	WriteData		No	-	-	BYTE[]/INT[]

		Written data buffer area				
S6	WriteDataSize	Written data size (in bytes)	No	-	0 to 494	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

NOTE

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress must be set to 192.168.1.88.

For soft elements, see the following table.

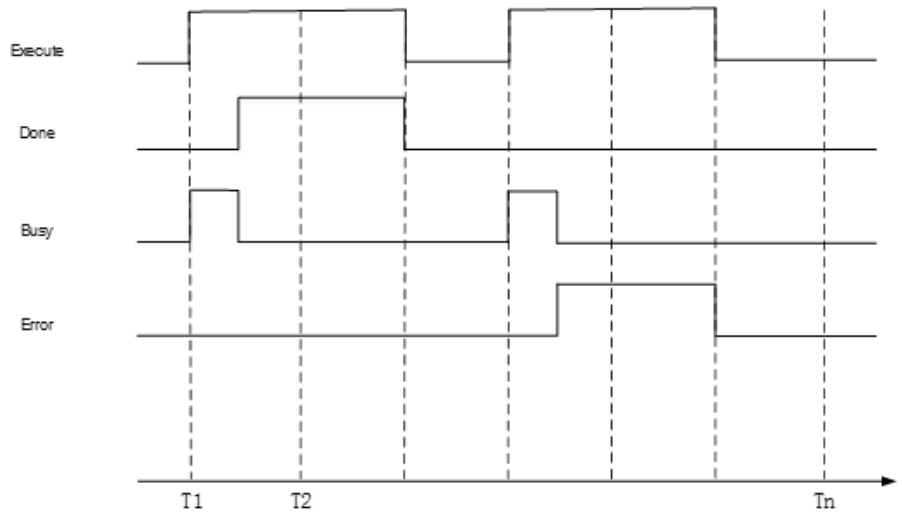
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	-	-	-	-
S6	-	-	-	√	√	√	√	-	-
D1	√ [1]	√	√	-	-	√	-	-	-
D2	√ [1]	√	√	-	-	√	-	-	-
D3	√ [1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

> **NOTE**
>
> [1] The X element is not supported.

Function

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

Sequence Diagram



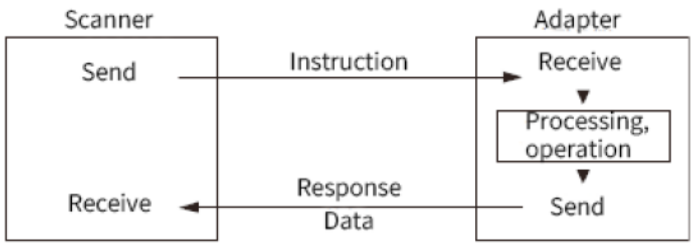
5.18.43 EIP_Apply_Attributes

Description

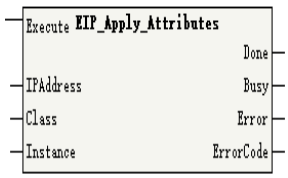
EIP_Apply_Attributes: Calling the "Apply_Attributes" service for a specific instance of the EtherNet/IP object

The adapter adopts and saves attributes set by "Get_Attribute_Single" or "Get_Attribute_All".

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block

Instruction	Name	LD Expression	LiteST Expression
EIP_Apply_Attributes	Calling the "Apply_Attributes" service for a specific instance of the EtherNet/IP object		-

The list of instructions is shown as follows.

16-bit instruction	-
32-bit instruction	EIP_Apply_Attributes: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

NOTE

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress must be set to 192.168.1.88.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

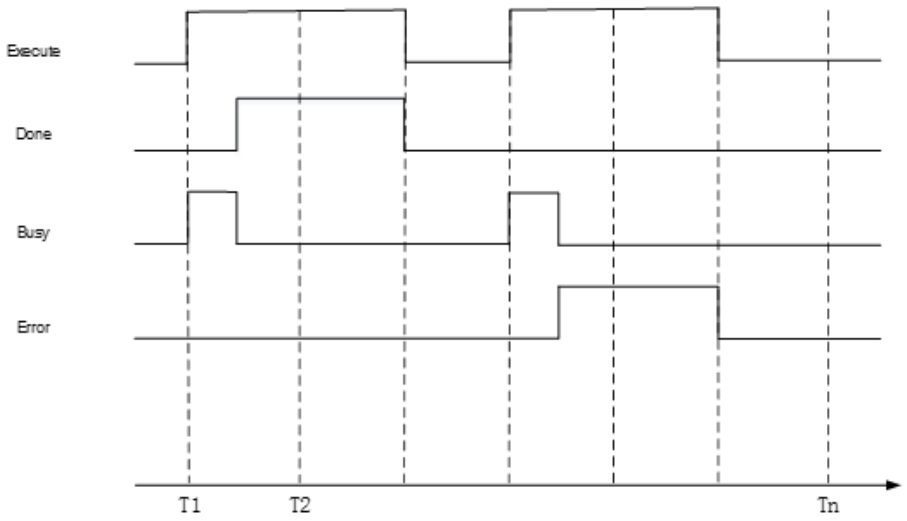
NOTE

[1] The X element is not supported.

Function

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

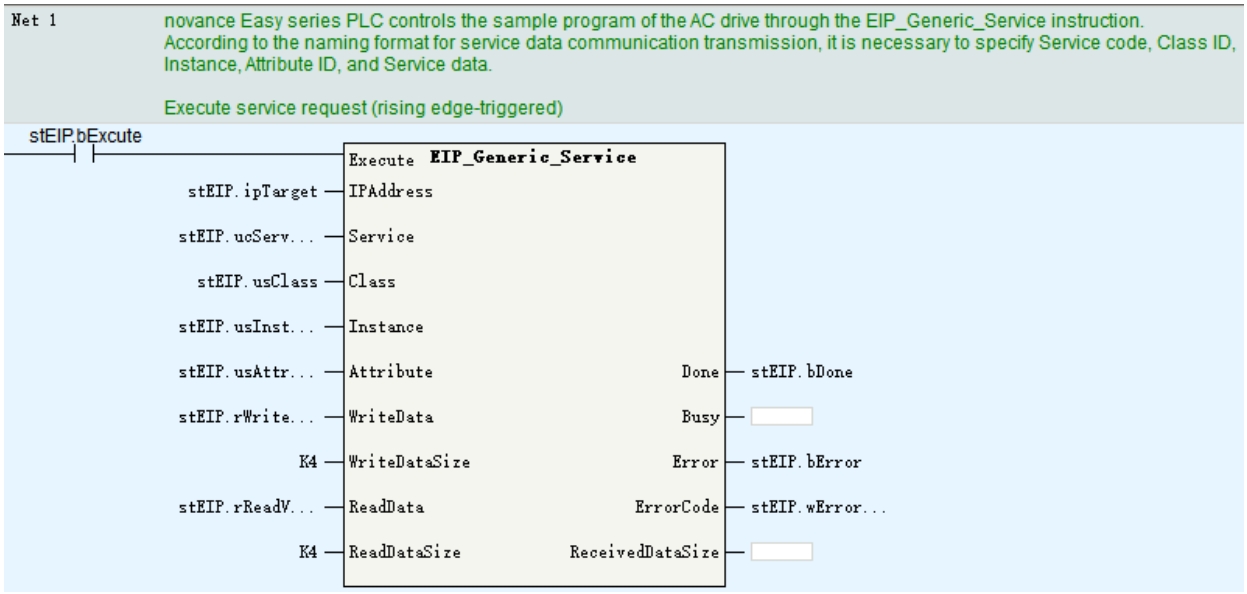
Sequence Diagram



5.18.44 EIP_Generic_Service通信实例

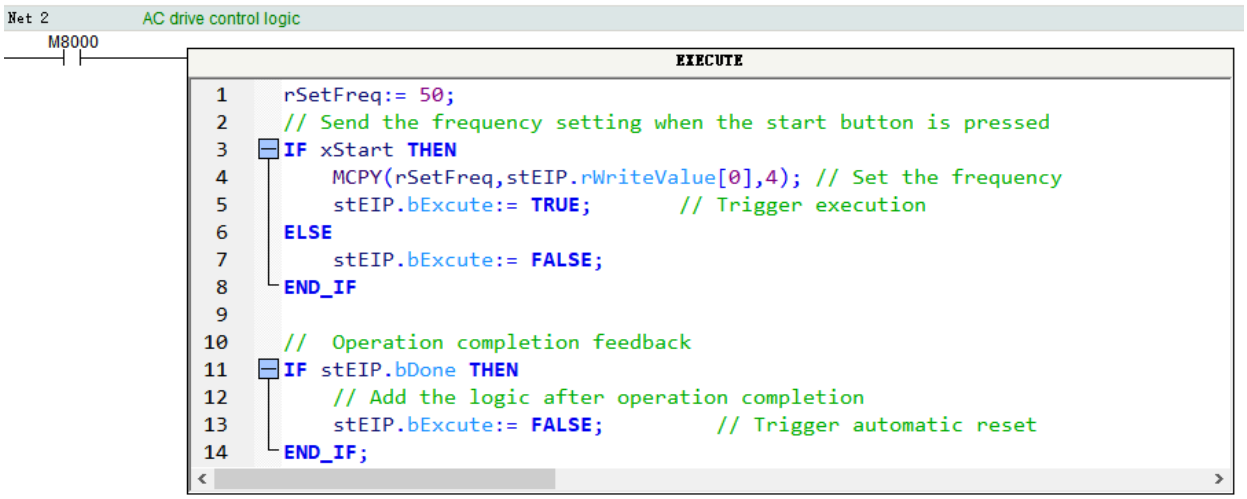
Inovance Easy series PLCs use the EIP_Generic_Service instruction to control sample programs of the AC drive. According to the naming format of the service data sending command, you must specify the following parameters: Service, Class, Instance, and Attribute. This instruction executes service requests (triggered on the rising edge).

The following figure shows the sample program.



The following table describes the examples of variable values.

Variable Name	Sample Input	Description
IPAddress	stEIP.ipTarget	IP address of the AC drive (such as 192.168.1.100)
Service	stEIP.ucServices	Service code (0x10 = Set_Attribute_Single)
Class	stEIP.usClass	Class ID of the AC drive (The default value is 0x00F2 for Inovance MD500.)
Instance	stEIP.usInstance	Instance ID (1 = Host)
Attribute	stEIP.usAttribute	Attribute ID (3 = Frequency reference)
WriteData	stEIP.rWriteValue	-
WriteDataSize	K4	-
ReadData	stEIP.rReadValue	Read value (in Hz).
ReadDataSize	K4	-

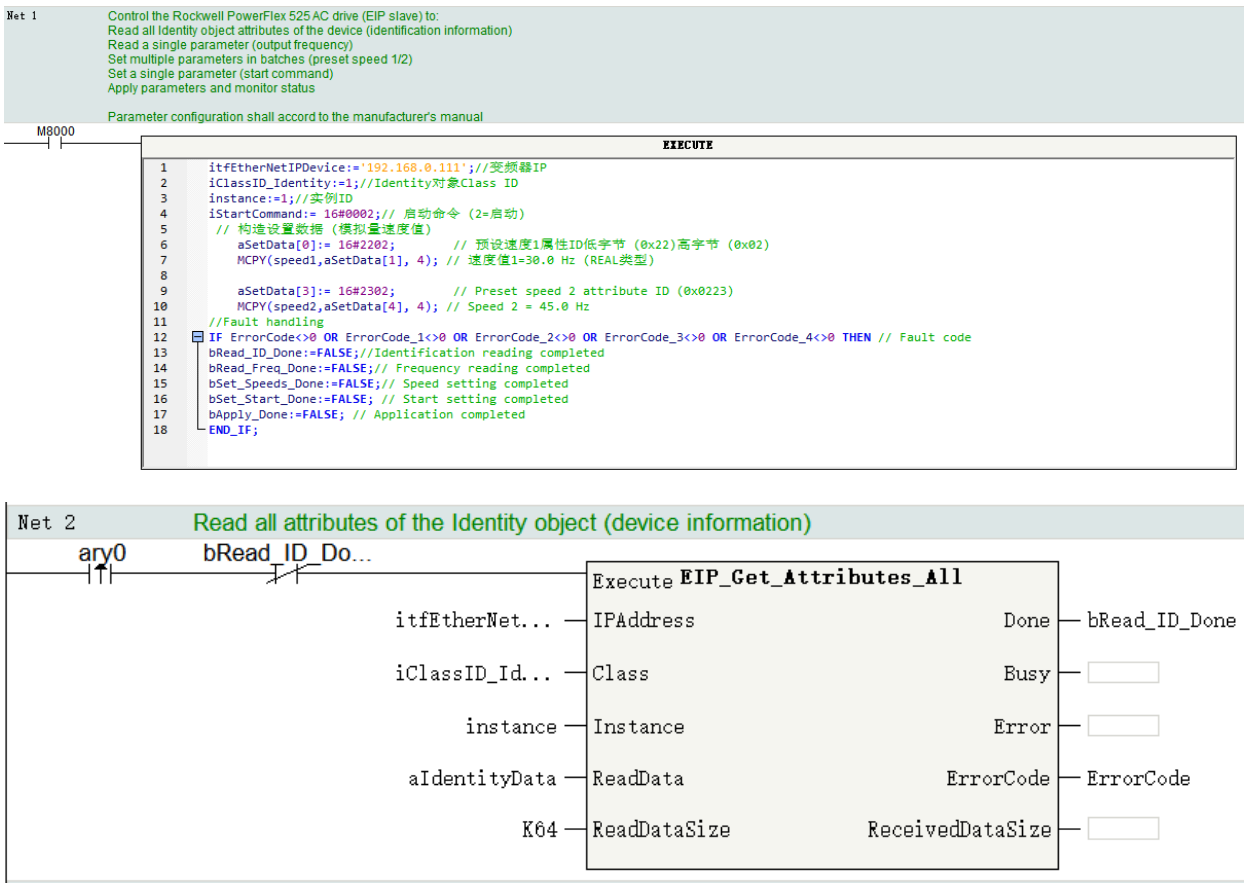


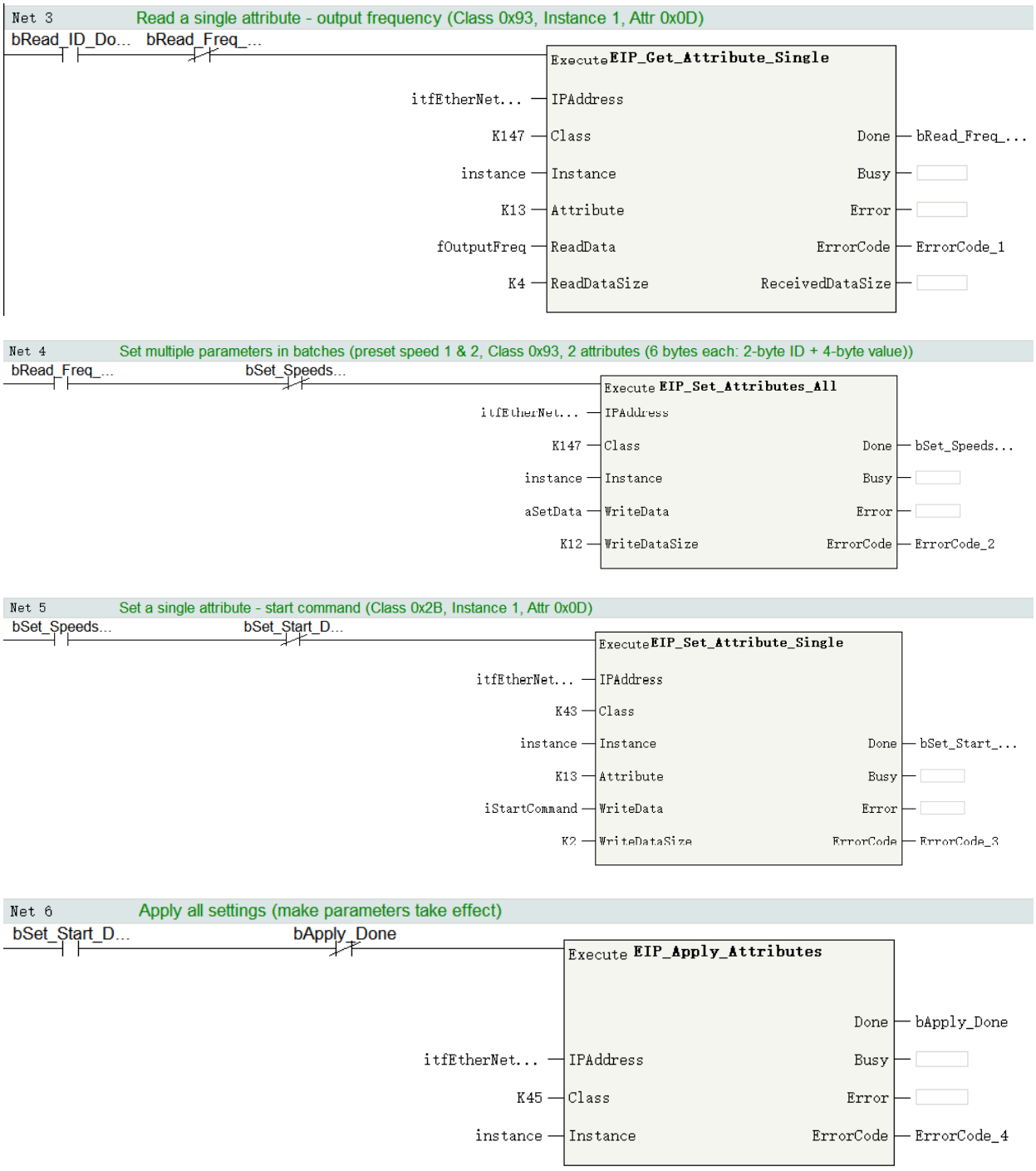
5.18.45 EIP Combined Application Instance

To control a Rockwell PowerFlex525 AC drive (EIP slave), you need to set the corresponding parameters based on the manufacturer manual to implement the following functions:

- Read all Identity object attributes (identification information) of the device.
- Read a single parameter (output frequency).
- Batch set multiple parameters (preset velocity 1/2).
- Set a single parameter (start command).
- Apply applications and monitor their status.

The following figure shows the sample program.



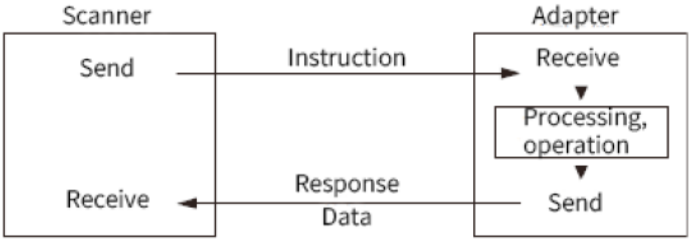


5.18.46 EIP_NOP

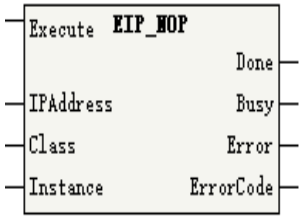
Description

It is often used to check whether the adapter is still available in the network.

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block

Instruction	Name	LD Expression	LiteST Expression
EIP_NOP	Calling the "NOP" (No Operation) service for a specific instance of the EtherNet/IP object		-

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	EIP_NOP: Continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	IPAddress	IP address *1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

> **NOTE**
>
> *1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress must be set to 192.168.1.88.

For soft elements, see the following table.

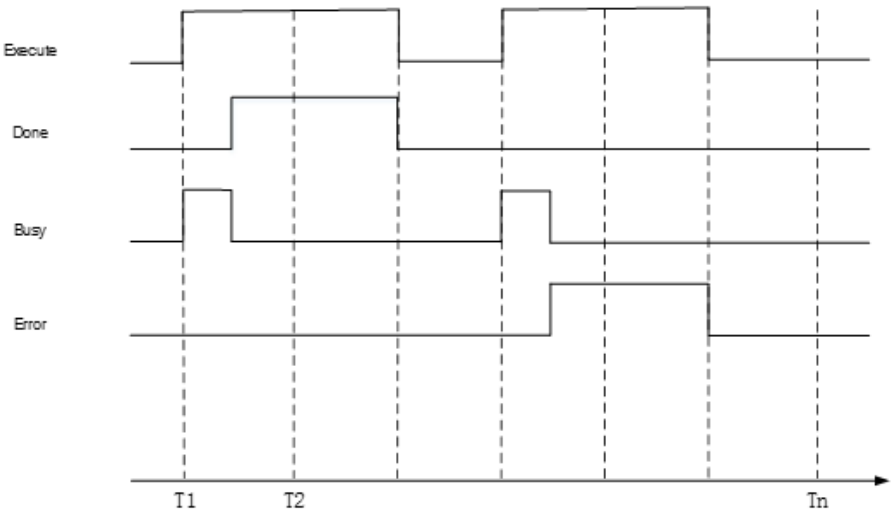
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	

> **NOTE**
>
> [1] The X element is not supported.

Function

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

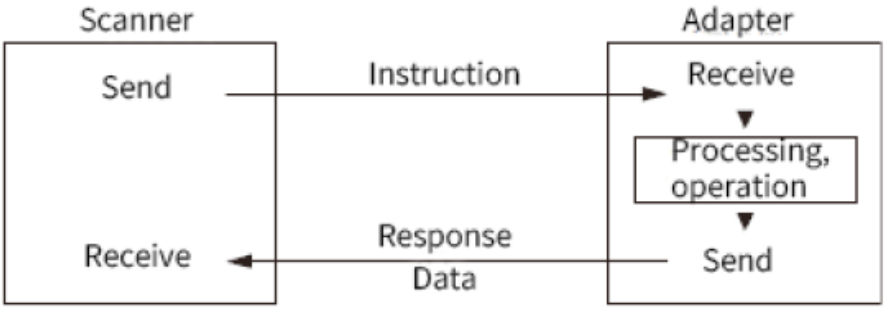
Sequence Diagram



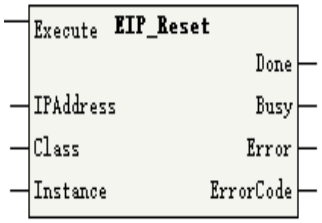
5.18.47 EIP_Reset

Description

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block

Instruction	Name	LD Expression	LiteST Expression
EIP_Reset	Calling the “Reset” service of a specific instance of the EtherNet/IP object		-

The instruction formats are as follows.

16-bit instruction	-					
32-bit instruction	EIP_Reset: Continuous execution					
Operands	Name	Description		Default	Value range	Data type

			Empty allowed			
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

NOTE

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress must be set to 192.168.1.88.

For soft elements, see the following table.

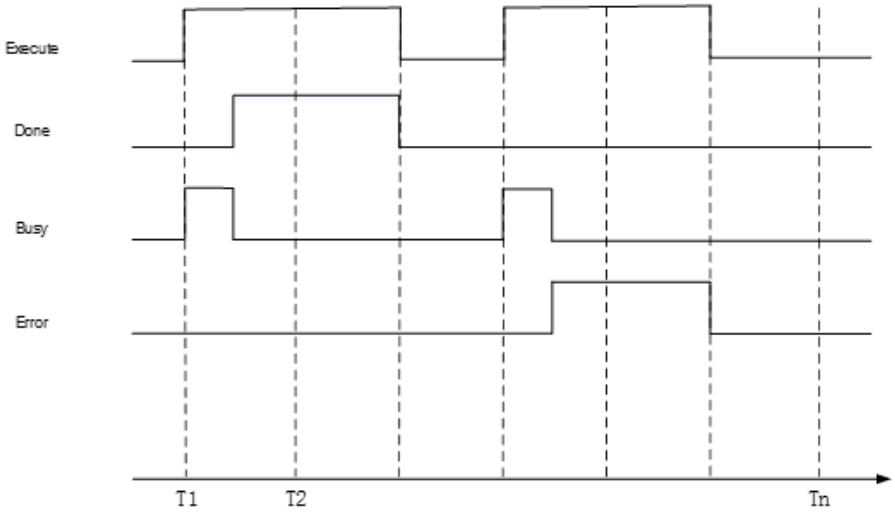
Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

> **NOTE**
>
> [1] The X element is not supported.

Function

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

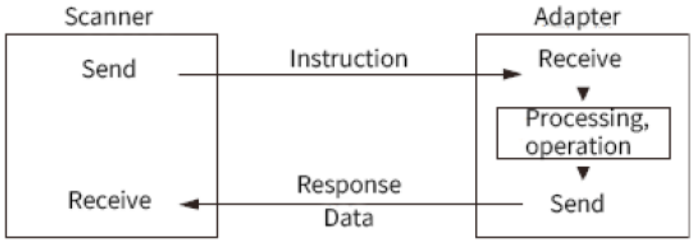
Sequence Diagram



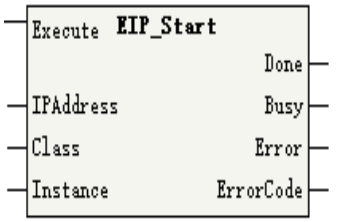
5.18.48 EIP_Start

Description

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block

Instruction	Name	LD Expression	LiteST Expression
EIP_Start	Calling the "Start" service of a specific instance of the EtherNet/IP object		-

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	EIP_Start continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

NOTE

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

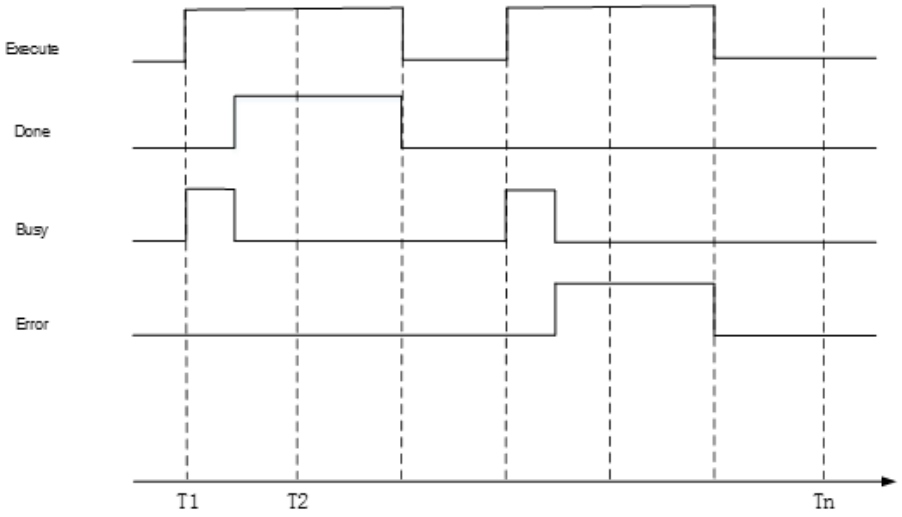
NOTE

[1] The X element is not supported.

Function

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

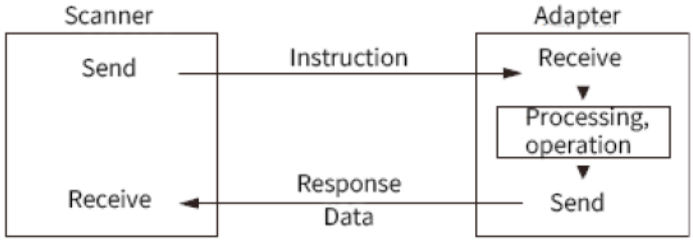
Sequence Diagram



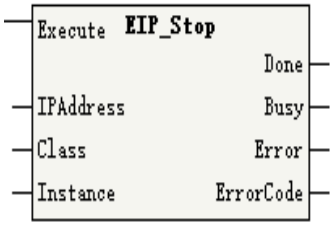
5.18.49 EIP_Stop

Description

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block

Instruction	Name	LD Expression	LiteST Expression
EIP_Stop	Calling the “Stop” service of a specific instance of the EtherNet/IP object		-

The instruction formats are as follows.

16-bit instruction	-
32-bit instruction	EIP_Stop continuous execution

Operands	Name	Description	Empty allowed	Default	Value range	Data type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

NOTE

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress must be set to 192.168.1.88.

For soft elements, see the following table.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

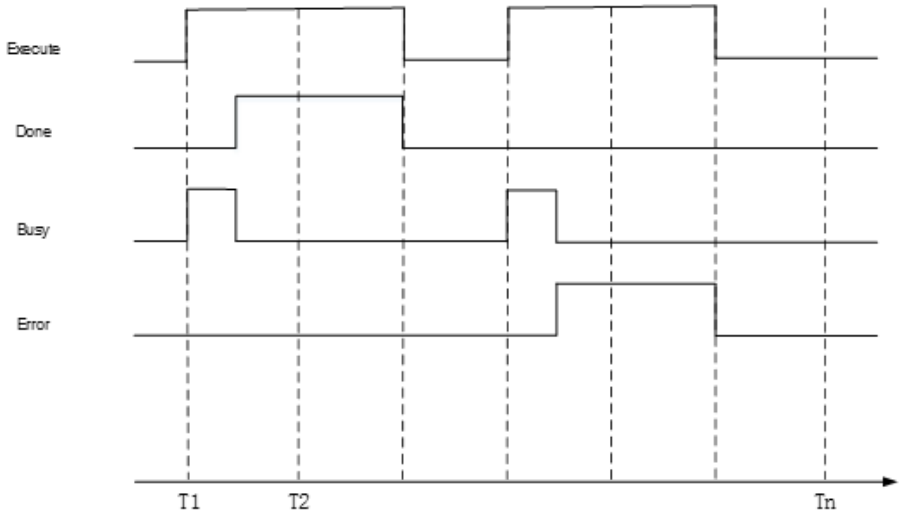
NOTE

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

Sequence Diagram



5.18.50 EIP_OpenWithDataSize

Introduction

EIP_OpenWithDataSize -- Create an EIP class 3 connection

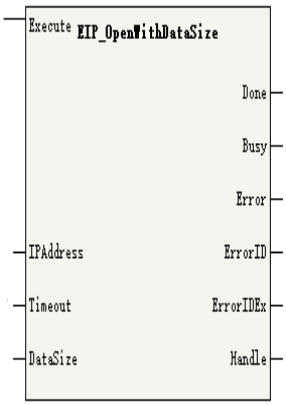
The EIP_OpenWithDataSize instruction creates an EIP class 3 connection to the remote device and returns the connection handle. This connection can read and write class 3 label data in the specified label path.

Version

- Firmware version: V6.4.0.0 and later
- Software version: AutoShop V4.10.2.0 and later

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
EIP_OpenWithDataSize	Create an EIP class 3 connection		-

Instruction	Name	LD Expression	LiteST Expression
			

The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	EIP_OpenWithDataSize: Triggered execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	DINT and IP
S2	Timeout	Timeout period	No	-	20 to 50000	DINT
S3	DataSize	Data length	No	-	1 to 8192	DINT
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D4	ErrorID	Error code*2	Yes	0	-	DINT
D5	ErrorIDEx	Extension error code	Yes	0	-	DINT
D6	Handle	Connection handle	No	-	-	DINT

NOTE

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress must be set to 192.168.1.88.

*2: For details, see 5.18.55 Error Codes of EIP Communication Instructions.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-
D6	-	-	-	√	√	√	-	-	-

NOTE

[1]: The X element is not supported.

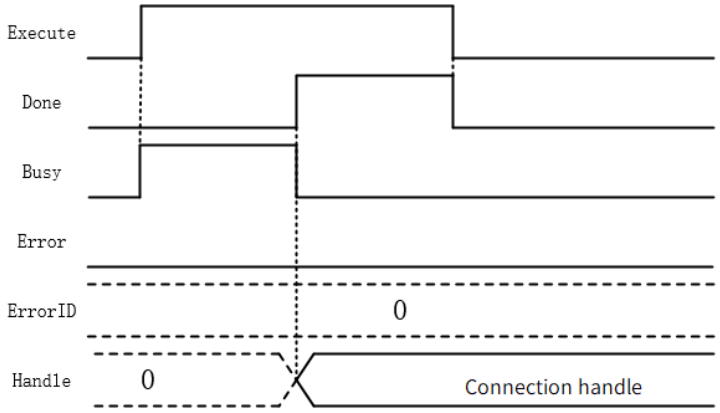
Function and Instruction Description

- **Function Description**
  - On the rising edge of the flow, the EIP_OpenWithDataSize instruction parses parameters, resets the Busy flag, sends a connection request to the slave station, and waits for response. If the slave station returns Connection success within the timeout period, the instruction resets the Done flag. Otherwise, the instruction resets the Error flag and outputs the corresponding error code.
  - On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, ErrorID, and ErrorIDEx are set to default values.
  - This instruction supports a maximum of 32 class 3 connections.

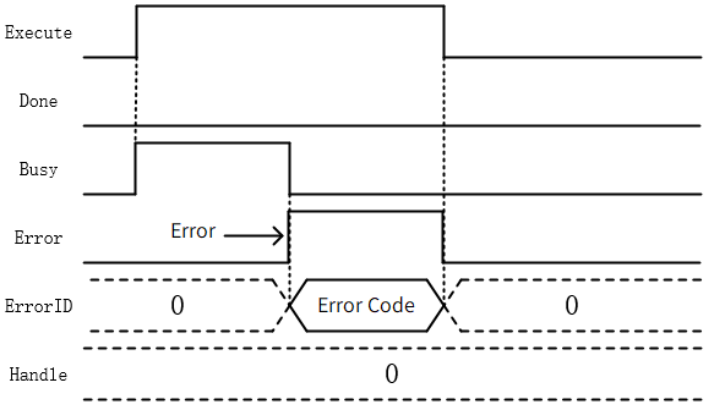
- **Parameter Description**
  - S2: Timeout period, in ms. Counting starts when the instruction is triggered to send a connection request.
  - S3: Data length, in bytes. This parameter specifies the maximum size of read/write labels of the corresponding connection.

Timing Diagram

The instruction is normally executed.



The instruction reports an error.



5.18.51 EIP_Close

Introduction

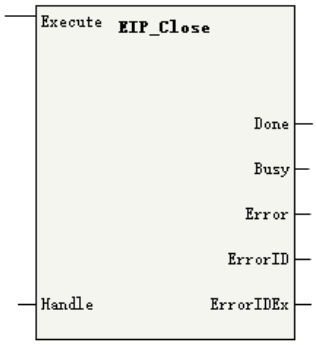
EIP_Close -- Close EIP class 3 connection

The EIP_Close instruction closes the EIP class 3 connection to the remote device.

Version

- Firmware version: V6.4.0.0 and later
- Software version: AutoShop V4.10.2.0 and later

Graphic Block



The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	EIP_Close: Triggered execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Handle	Connection handle	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D4	ErrorID	Error code*1	Yes	0	-	DINT
D5	ErrorIDEx	Extension error code	Yes	0	-	DINT

NOTE

*1: For details, see 5.18.55 Error Codes of EIP Communication Instructions.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	√
D1	√[1]	√	√	-	-	√	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

NOTE

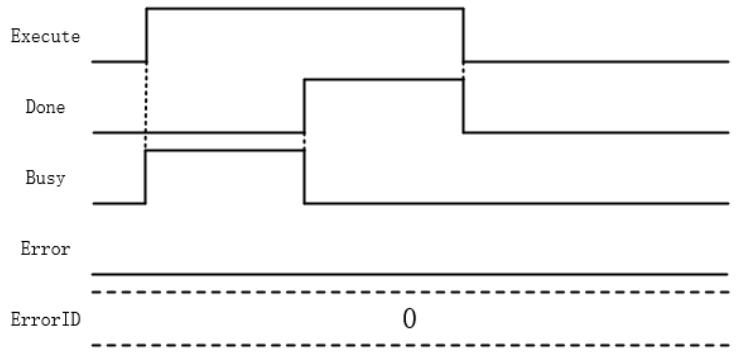
[1]: The X element is not supported.

Function and Instruction Description

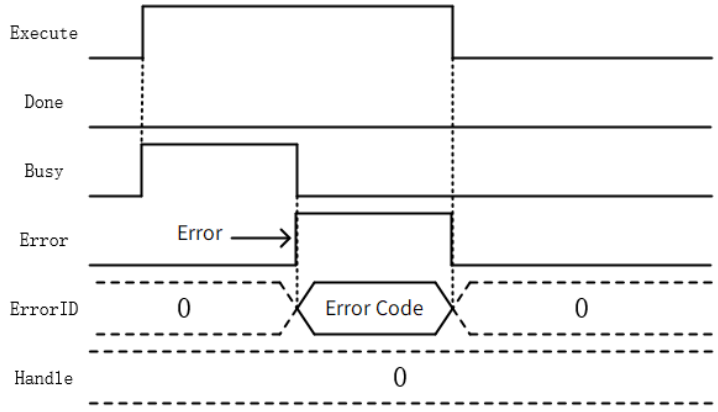
- **Function description**
  - On the rising edge of the flow, the EIP_Close instruction parses parameters, resets the Busy flag, closes a connection request to the slave station, and waits for response. If the slave station returns Closing success, the instruction resets the Done flag. Otherwise, the instruction resets the Error flag and outputs the corresponding error code.
  - On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, ErrorID, and ErrorIDEx are set to default values.
- **Parameter description**
  - S1: Connection handle, which is obtained by using the EIP_OpenWithDataSize instruction.

Timing Diagram

- The instruction is normally executed.



- The instruction reports an error.



5.18.52 EIP_Read

Introduction

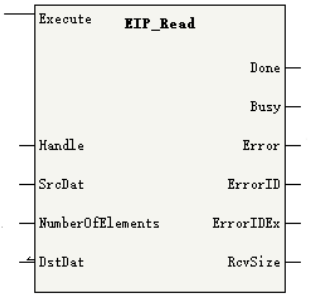
EIP_Read -- Read CIP class 3

The EIP_Read instruction reads label variable values (class 3 information) of a remote device on the EIP network.

Version

- Firmware version: V6.4.0.0 and later
- Software version: AutoShop V4.10.2.0 and later

Graphic Block



The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	EIP_Read: Triggered execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Handle	Connection handle	No	-	-	DINT
S2	SrcDat	Variable name of the external device	No	-	-	STRING

S3	NumberOfElements	Number of elements to read	No	-	-	DINT
S4	DstDat	Read data	No	-	-	BOOL, BYTE, INT, DINT, REAL, string, and STRUCT*1
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D4	ErrorID	Error code*2	Yes	0	-	DINT
D5	ErrorIDEx	Extension error code	Yes	0	-	DINT
D6	RcvSize	Size of read data	Yes	0	-	DINT

NOTE

*1: STRUCT indicates a custom structure type.

*2: For details, see 5.18.55 Error Codes of EIP Communication Instructions.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	-	-	-	-	-	√
S3	-	-	-	√	√	√	√	-	-
S4	√[1]	√	√	√	√	√	-	-	√
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D6	-	-	-	√	√	√	-	-	-

NOTE

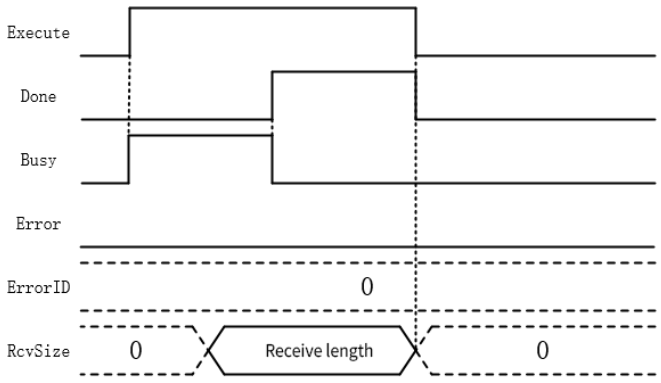
[1]: The X element is not supported.

Function and Instruction Description

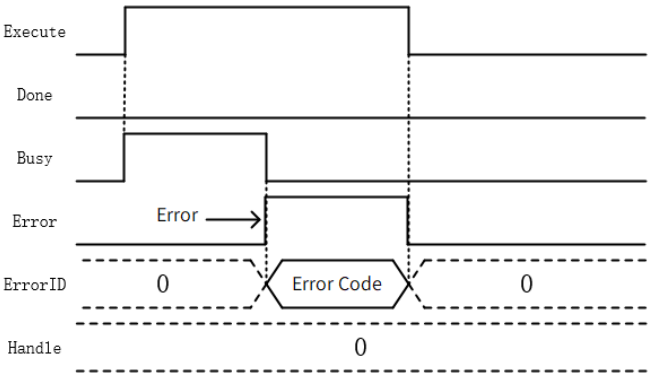
- **Function description**
  - On the rising edge of the flow, the EIP_Read instruction parses parameters, resets the Busy flag, parses the type of local variables, sends a read request to the slave station, and waits for response. If the slave station returns Success, the instruction parses the label data returned from the slave station. If the data is correct, the instruction writes the label data into local variables and resets the Done flag. Otherwise, the instruction resets the Error flag and outputs the corresponding error code.
  - On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, ErrorID, and ErrorIDEx are set to default values.
  - This instruction cannot read label data of the STRING and array types.
  - This instruction cannot access array label variables as a whole. (The label path is an array and the number of requested element is 1.)
- **Parameter description**
  - S1: Connection handle, which is obtained by using the EIP_OpenWithDataSize instruction.
  - S2: Variable name of the external device. The variable names are read from the slave station. It supports the array subscript ([n]) and structure path (struct.member).
  - S4: Read data. The variable type of this parameter must be the same as that of the slave station from which data is read.
  - D6: Size of the read data. This parameter specifies the size of label data in EIP communication data, which may differ from the memory size used by the variables.

Timing Diagram

- The instruction is normally executed.



- The instruction reports an error.



5.18.53 EIP_Write

Introduction

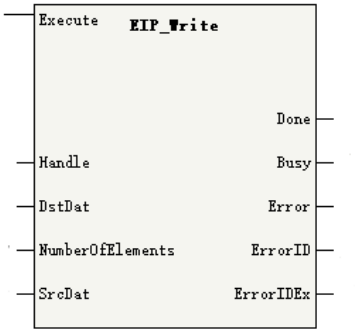
EIP_Write -- Write CIP class 3

The EIP_Write instruction writes variable values to label variables (class 3 information) of a remote device on the EIP network.

Version

- Firmware version: V6.4.0.0 and later
- Software version: AutoShop V4.10.2.0 and later

Graphic Block



The following table describes the instruction format.

16-bit Instruction	-
32-bit Instruction	EIP_Write: Triggered execution

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Handle	Connection handle	No	-	-	DINT
S2	DstDat	Variable name of the external device	No	-	-	STRING
S3	NumberOfElements	Number of elements to write	No	-	-	DINT
S4	SrcDat	Written data	No	-	-	BOOL, BYTE, INT, DINT, REAL, string, and STRUCT*1
D1	Done	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON or OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON or OFF	BOOL
D4	ErrorID	Error code*2	Yes	0	-	DINT
D5	ErrorIDEx	Extension error code	Yes	0	-	DINT

NOTE

*1: STRUCT indicates a custom structure type.

*2: For details, see 5.18.55 Error Codes of EIP Communication Instructions.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S2	-	-	-	-	-	-	-	-	√
S3	-	-	-	√	√	√	√	-	-
S4	√[1]	√	√	√	√	√	-	-	√
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

NOTE

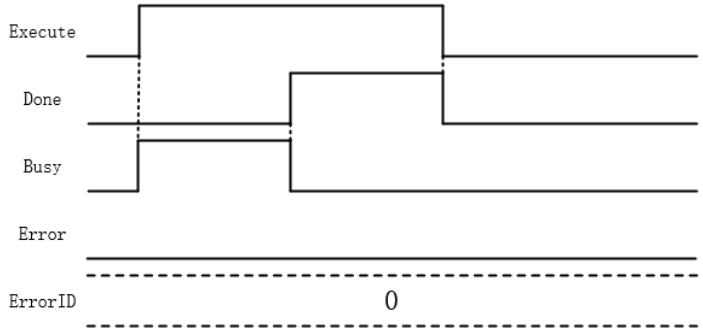
[1]: The X element is not supported.

Function and Instruction Description

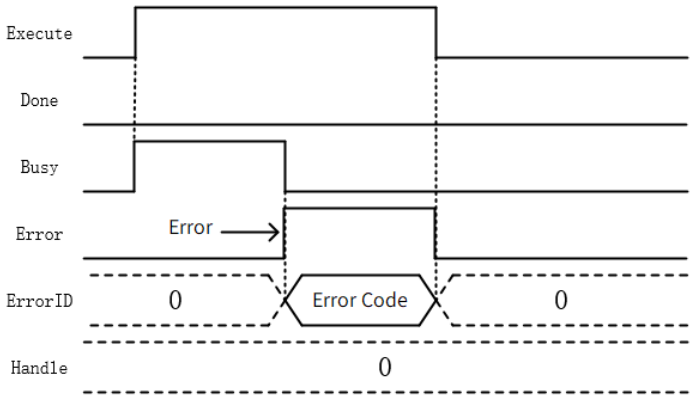
- **Function Description**
  - On the rising edge of the flow, the EIP_Write instruction parses parameters, resets the Busy flag, parses the value of local variables, sends a write request and label data to the slave station, and waits for response. If the slave station returns Success, the instruction resets the Done flag. Otherwise, the instruction resets the Error flag and outputs the corresponding error code.
  - On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, ErrorID, and ErrorIDEx are set to default values.
  - This instruction cannot read label data of the STRING and array types.
  - This instruction cannot access array label variables as a whole. (The label path is an array and the number of requested element is 1.)
- **Parameter Description**
  - S1: Connection handle, which is obtained by using the EIP_OpenWithDataSize instruction.
  - S2: Variable name of the external device. This parameter specifies variable names of the slave station to be written. It supports the array subscript ([n]) and structure path (struct.member).
  - S4: Written data. The variable type of this parameter must be the same as that of the slave station whose data is written.

Timing Diagram

- The instruction is normally executed.

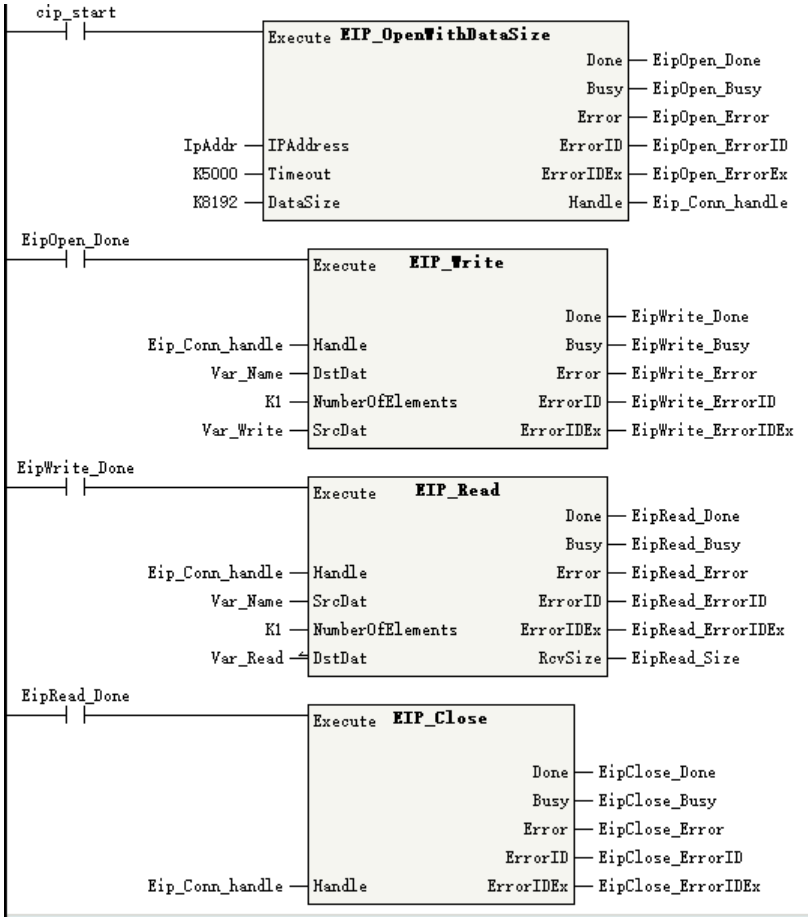


- The instruction reports an error.



5.18.54 EIP Class3 Label Communication Instance

The Easy series PLC functions as an EIP scanner, and the EIP_OpenWithDataSize instruction is used to create a class 3 connection. After the connection is successful, use the EIP_Write instruction to write the values of local variables to the remote adapter labels, and then use the EIP_Read instruction to read the remote adapter labels to local variables. After data read and write are complete, use the EIP_Close instruction to close the connection.



5.18.55 Error Codes of EIP Communication Instructions

NOTE

- Firmware version: V6.4.0.0 and later
- Software version: AutoShop V4.10.2.0 and later

The following table lists the error codes of EIP communication instructions.

Error Code	Description
6602	The connection path is invalid.
6603	The remote slave station does not respond.
6604	The remote slave station responds incorrectly.
6605	Response from the remote slave station times out.
6634	The data type of the response from the remote slave station is different from that of the local variable.
6635	The data length of the response from the remote slave station is different from that of the local variable.
6660	The EIP protocol is not running.
6661	The control block is not found.

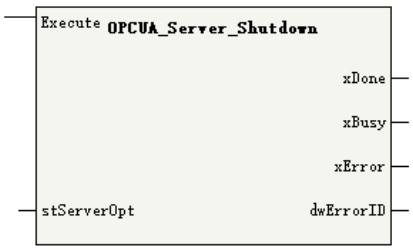
Error Code	Description
6662	Failed to create a control block.
6663	Failed to call the EIP protocol stack interface.
6664	Invalid input parameter.
6665	The connection does not exist.
6666	The connection is in use.
6667	Unknown event.
6668	The requested label name is invalid.
6669	Invalid local label.
6670	System error.
6671	The number of connections exceeds the limit.
6672	Internal error.
6673	The maximum size of the label variable that can be read or written exceeds the preset value.
6688	External error. For example, the specified label does not exist on the remote slave station.

5.18.56 OPCUA_Server_Shutdown

Introduction

OPCUA_Server_Shutdown -- Shut down OPC UA server

Graphic Block



The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	OPCUA_Server_Shutdown: Triggered execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type

S1	stServerOpt	Server attribute	No	-	-	INT
D1	xDone	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	xBusy	Executing	Yes	OFF	ON or OFF	BOOL
D3	xError	Error flag	Yes	OFF	ON or OFF	BOOL
D4	dwErrorID	Error code*1	Yes	0	-	INT

NOTE

*1: For details, see the error description below.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

NOTE

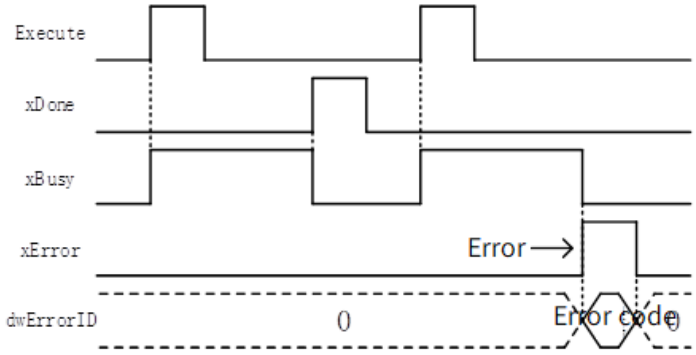
[1]: The X element is not supported.

Function and Instruction Description

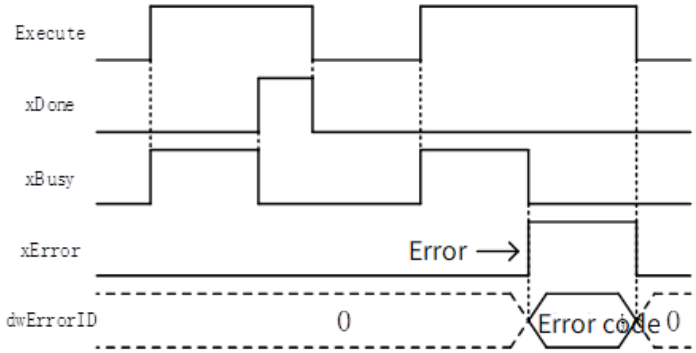
After the OPCUA_Server_Shutdown instruction is triggered, it shuts down functions of the OPC UA server.

Timing Diagram

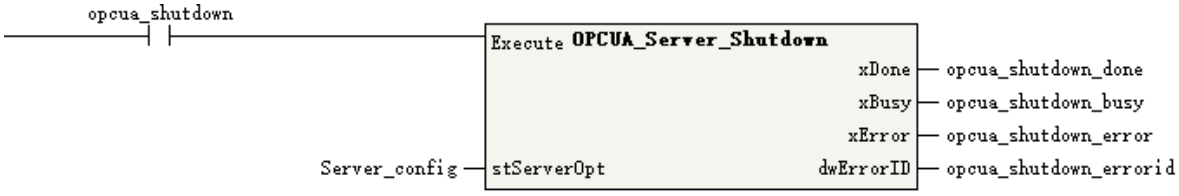
- The flow is triggered for one cycle.



- The flow remains active (ON).



Instruction Example



Error Description

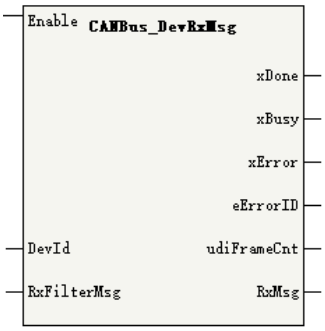
The following table describes error codes of the OPCUA_Server_Shutdown instruction.

Error Code	Description
1	The OPC UA server is disabled.
2	The OPC UA server is being shut down.
3	Shutting down the OPC UA server times out.

5.18.57 CANBus_DevRxMsg

CANBus_DevRxMsg -- Receive CAN free protocol data

Graphic Block



The following table describes the instruction format.

16-bit Instruction	-
32-bit Instruction	CANBus_DevRxMsg: Level execution

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	DevId	CAN network ID	No	-	0 to 63	INT
S2	RxFilterMsg	Setting of information reception and filtering	No	-	-	_sCAN_Rx_Filter*1
D1	xDone	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	xBusy	Executing	Yes	OFF	ON or OFF	BOOL
D3	xError	Error flag	Yes	OFF	ON or OFF	BOOL
D4	eErrID	Error code*2	Yes	0	-	INT
D5	udiFrameCnt	Frame counter. The number of received CAN frames is counted only when the conditions are met.	Yes	0	-	DINT
D6	RxMsg	CAN frame received last	Yes	-	-	_sCAN_MSG*1

NOTE

*1: For details, see the structure description below.

*2: For details, see the error description below.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	-	-	-	-	-	√
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-
D6	-	-	-	-	-	-	-	-	√

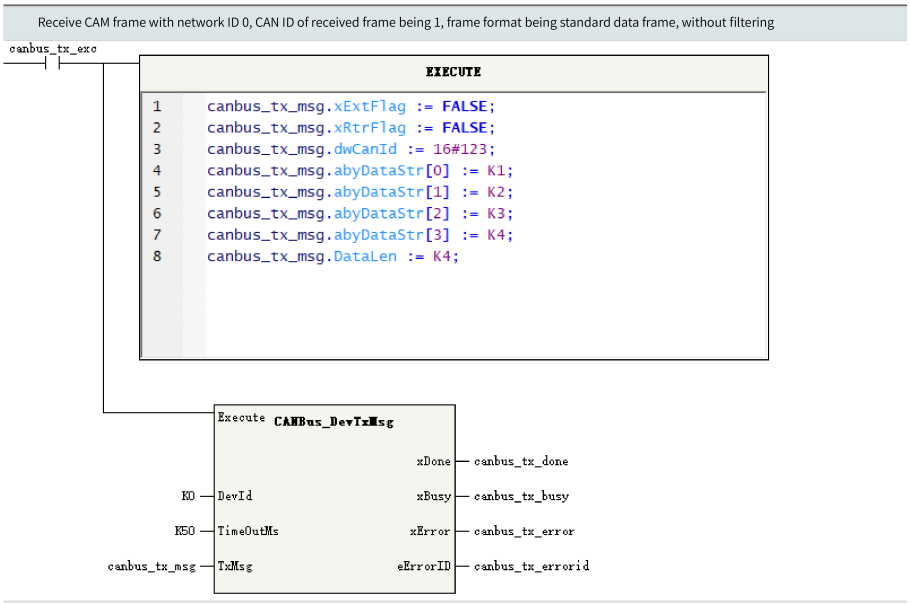
NOTE

[1]: The X element is not supported.

Function and Instruction Description

- When the flow in the CANBus_DevRxMsg instruction is active, the PLC receives CAN frames from the CAN network with the network ID specified by DevId. The reception conditions are specified by the RxFilterMsg pin.
- Before you use a CAN bus instruction, you must set the CAN network protocol to a CAN free protocol and configure the correct baud rate.
- Up to 128 instantiated receive function blocks can be enabled at a time.

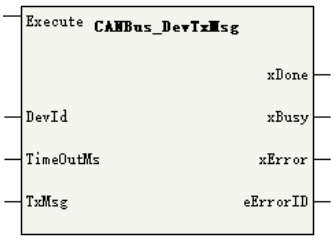
Instruction Example



5.18.58 CANBus_DevTxMsg

CANBus_DevTxMsg -- Send CAN free protocol data

Graphic Block



The following table describes the instruction format.

16-bit Instruction	-					
32-bit Instruction	CANBus_DevTxMsg: Triggered execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	DevId	CAN network ID	No	-	0 to 63	INT
S2	TimeOutMs	Timeout period, in ms. If this parameter is set to 0, the detection timeout period is 50 ms.	Yes	-	0 to 65,535	INT
S3	TxMsg	CAN frame that is sent	Yes	-	-	

						_sCAN_MSG*1
D1	xDone	Completion flag	Yes	OFF	ON or OFF	BOOL
D2	xBusy	Executing	Yes	OFF	ON or OFF	BOOL
D3	xError	Error flag	Yes	OFF	ON or OFF	BOOL
D4	eErrID	Error code*2	Yes	0	-	INT

NOTE

*1: For details, see the structure description below.

*2: For details, see the error description below.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	-	-	-	-	-	√
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

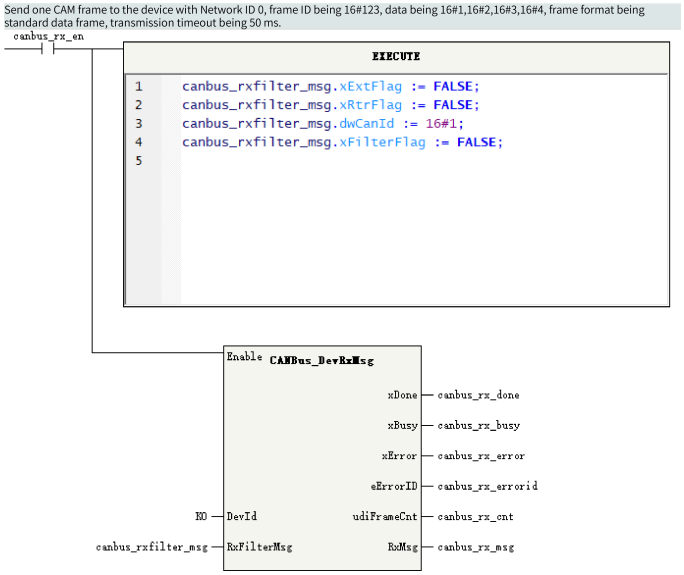
NOTE

[1]: The X element is not supported.

Function and Instruction Description

- When the CANBus_DevTxMsg is triggered on the rising edge, the PLC sends a CAN frame to the CAN network with the network ID specified by DevId. The sent content is specified by the TxMsg pin.
- Before you use a CAN bus instruction, you must set the CAN network protocol to a CAN free protocol and configure the correct baud rate.
- Only one CANBus_DevTxMsg function block can be triggered at a time. Otherwise, a frame sending error is reported.

Instruction Example



5.18.59 Description of CANBUS Structure

_sCAN_Rx_Filte

The following table describes variables in the _sCAN_Rx_Filter structure that defines information reception and filtering for CAN bus.

Variable Name	Data Type	Minimum	Maximum	Default	Description
xExtFlag	BOOL	OFF	ON	ON	Extension frame flag OFF: Standard frame ON: Extension frame
xRtrFlag	BOOL	OFF	ON	ON	Remote flag OFF: Data frame ON: Remote frame
dwCanId	DINT	0	65535	0	CAN network ID

Variable Name	Data Type	Minimum	Maximum	Default	Description
					Lower 11 bits are active for standard frames. Higher 29 bits are active for extension frames.
xFilterFlag	DINT	OFF	ON	ON	Filtering mechanism enable flag OFF: Disable filtering and receive by single ID ON: Enable filtering
xFilterExt	BOOL	OFF	ON	ON	Extension frame filtering OFF: Disable ON: Enable
xFilterRtr	BOOL	OFF	ON	ON	Remote frame filtering OFF: Disable ON: Enable
dwFilterCanId	DINT	0	65535	0	CAN network ID filtering by bit bitn = 0: Disable bitn = 1: Enable

_sCAN_MSG

The following table describes variables in the _sCAN_MSG structure that defines frame information for CAN bus.

Variable Name	Data Type	Minimum	Maximum	Description
xExtFlag	BOOL	OFF	ON	Extension frame flag OFF: Standard frame ON: Extension frame

Variable Name	Data Type	Minimum	Maximum	Description
xRtrFlag	BOOL	OFF	ON	Remote flag OFF: Data frame ON: Remote frame
dwCanId	DINT	0	65535	CAN network ID Lower 11 bits are active for standard frames. Higher 29 bits are active for extension frames.
DataLen	BYTE	0	8	Data length Data frame: 0 to 8 Remote frame: 0
abyDataStr	BYTE[8]	-	-	Data area The maximum number of bytes is 8 for the data area of CAN frames.

5.18.60 CANBUS Error Codes

The CANBUS error codes are defined as follows.

Error code	Description
0	The function block runs normally or has no error.
1	The function block failed to run.
2	The device ID is invalid because this ID is not enabled.
3	The device is faulty.
4	The input parameters of the function block are incorrect.
5	The function block is busy.
6	An error occurs when the function block is aborted.
7	The function block times out.
8	The function block pointer is null.
9	The function block buffer is used up.

5.19 User Program Backup Instructions

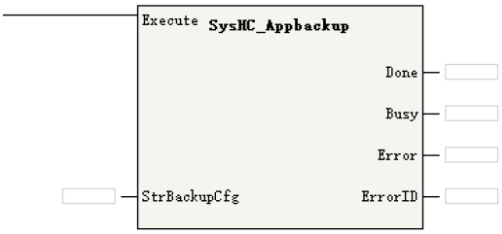
5.19.1 SysHC_Appbackup

Introduction

The SysHC_Appbackup instruction backs up the programs running on the PLC to the SD card.

SysHC_Appbackup -- User program backup

Graphic Block



The following table describes the instruction format.

16-bit Instruction	SysHC_Appbackup: Triggered execution				
32-bit Instruction	-				
Operand	Name	Description	Default	Range	Data Type
S0	StrBackupCfg	Whether to back up the project source code, non-volatile data, and backup path	Back up the project source code and non-volatile data to the SD card	INT[0] specifies the backup options. 0: Back up the project source code and non-volatile data 1: Back up the project source code only 2: Back up the non-volatile data 3: Do not back up the project source code and non-volatile data INT[1-31] is reserved.	INT[32]
D0	Done	Completion flag	OFF	ON or OFF	BOOL
D1	Busy	Busy flag	OFF	ON or OFF	BOOL
D2	Error	Error flag	OFF	ON or OFF	BOOL
D3	ErrorID	Error code	0	For details, see the "Program	INT

				Backup Error Codes" section.	

NOTE

- This instruction backs up the programs running on the PLC (without login password) to the SD card when the PLC is running.
- You do not need to create the backup path in advance. By default, the backup data is stored in the BackupProgram folder of the SD card.
- If you set StrBackupCfg to 1 but the project source code is not downloaded, backup fails. You need to download the source project again.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S0	-	-	-	-	√[2]	-	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

NOTE

[1]: The X element is not supported.

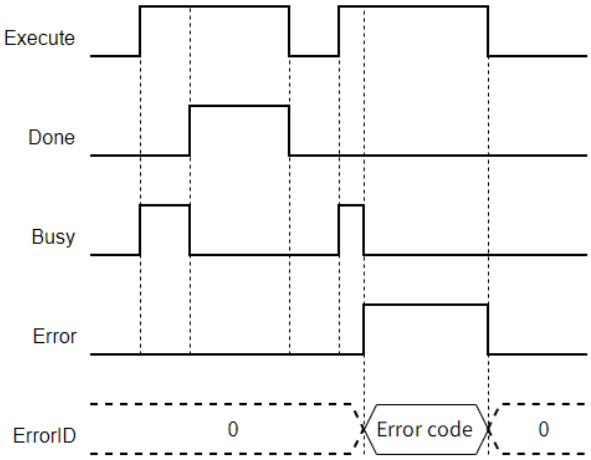
[2]: Only custom variables of the INT[32] type are supported.

Function and Instruction Description

The SysHC_Appbackup instruction backs up programs running on the PLC to the SD card.

- If a password is required to log in to the PLC, this instruction cannot be executed and an error is reported. If no password is required, this instruction can be normally executed.
- After this instruction is triggered, the PLC automatically backs up programs running on the PLC, project source code (if selected), and non-volatile data (if selected) to the SD card.

Timing Diagram



Program Backup Error Codes

Error codes of the SysHC_Appbackup instruction

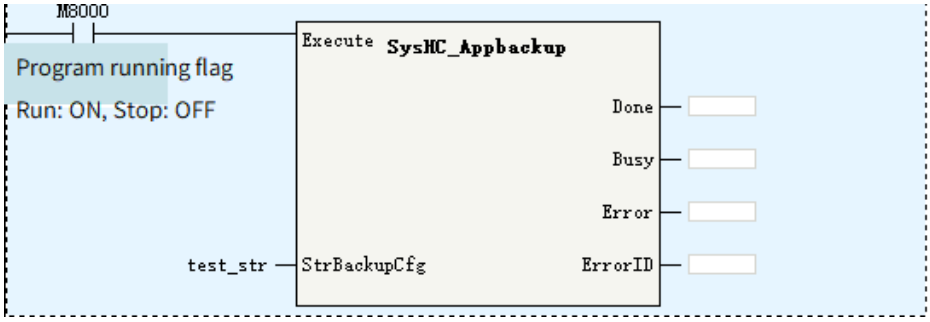
Error Code	Description
0	No fault occurs.
0x1	A password is required to log in to the PLC, and copy is not allowed.
0x11	The source project is not downloaded to the PLC, so the source project cannot be backed up.
0x13	The instruction is called repeatedly within 10 seconds.
0x31	Failed to create the backup directory before backup execution.
0x32	Failed to pack the backup files during the backup process.
0x33	No corresponding backup file is generated after backup ends.
0xAA	Invalid backup parameter.
0xEE	No SD card or USB flash disk is detected.
0xDD	The storage space is insufficient (less than 256 MB).

NOTE

- The servo does not respond if the instruction is called repeatedly within 10 seconds.
- The space of the SD card is 256 MB to 32 GB, and the remaining space is less than 256 MB.

Instruction Example

When the instruction is active on the M8000, the first rising edge executes the program backup instruction. During the execution process, the Busy signal is ON, the Done signal is ON after successful backup, the Error signal is ON if backup fails, and ErrorID displays the error code.



Restoring the Backup User Program

H5U

Load the SD card with the backup user program to the master module of the PLC. Switch the PLC to Stop state. Press and hold the MFK key on the PLC panel for 3 seconds to enter the **Sd** menu. Press the MFK key again to start programming the user program in the SD card into the PLC host. The LED display shows the programming progress (00 to 99), and shows PP after the programming is completed.



Easy Series

1. Insert the SD card with the backup user program into a TF expansion card, and then install the TF expansion card onto the PLC.
2. Power on the PLC again. The PLC starts loading the user program from the SD card, and the RUN indicator blinks quickly at 4 Hz during the programming process.
3. After successful programming, the RUN indicator blinks slowly at 1 Hz. Remove the SD card after the PLC enters the Stop state.
4. Power off and then power on the PLC and check whether it restores to normal.

- If the backup user program fails to be restored on the H5U series PLC, the error code is displayed on the LED, which is the same as that in the Down file.

Code	Name	Description
	E1	PLC is in a non-safe state (running or downloading), operation prohibited.
	E2	SD card or programming file not detected.
	E3	Multiple programming files detected on the SD card.
	E4	Programming file data is abnormal or device model is incompatible.
	E5	Password verification failed.

- If the ERR indicator on the Easy series PLC blinks slowly, programming fails. Check whether the programming file fits the PLC model and whether the correct login password of the PLC is entered.

5.20 Other Instructions

5.20.1 List of Instructions

The following table lists the pointer instructions.

Category	Version	Name	Function
PID instruction	Firmware version: V6.3.0.3 and later	PID	PID calculation
		PID_AT	Universal PID controller
		PID_ATC	Bipolar temperature PID controller

5.20.2 PID

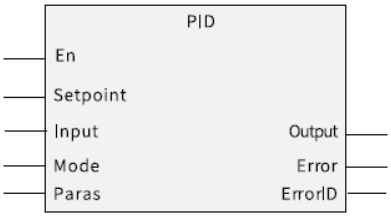
Description

The PID instruction performs PID calculation to control the parameters of a close-loop control system.

PID: PID calculation

Firmware version: V6.3.0.3 or above.

Graphic Block



The instruction formats are as follows.

16-bit instruction	PID: Continuous execution			
32-bit instruction	-			
Operands	Name	Description	Value range	Data type
S1	Setpoint	Set target control value. Unit: 0.1°C	-	INT16
S2	Input	Indicates the measured feedback value (in 0.1°C). The user program needs to read the actual value of the device and update this parameter.	-	INT16
S3	Mode	PID mode, Mode=6, Auto-tuning PID	-	INT16
S4	Paras	Operation parameter, the start unit for storing the operation result. It occupies consecutive D elements. S4 is used for user parameter setting and control.	-	INT16, array*100

D1	Output	Analog output PID analog output, 0 to 20000 by default	-	INT16
D2	Error	Error flag	-	BOOL
D3	ErrorID	Error code	-	INT16

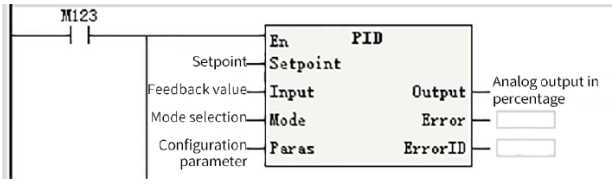
See the following table for soft elements.

Operands	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of word element	Custom bit variable	D, R, W	Custom word variable	Pointer variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	-	-	-	-
D1	-	-	-	√	√	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

NOTE

[1] The X element is not supported.

Instruction Description



S3 specifies the PID mode, which is described as follows.

Parameter	Mode	Description
S3	0	Incremental PID
	1	Position-type PID
	2	Special PID
	3	Temperature control PID

Parameter	Mode	Description
	4	MPC control (used for process control in the air compressor industry)
	5	Large-inertia temperature control PID (used for process control in the injection molding machine industry)
	6	Auto-tuning PID

Mode 0: Incremental PID Instruction

Unit	Function	Description
S4	Sampling time (TS)	The maximum sampling time is 132767 ms and the sampling time must be greater than the PLC's scan cycle.
S4+1	Action direction (ACT)	Bit 0 = 0: forward action; bit 0 = 1: reverse action Bit 1 = 0: input variable alarm disabled; bit 1 = 1: input variable alarm enabled Bit 2 = 0: output variable alarm disabled; bit 2 = 1: output variable alarm enabled Bit 3: unavailable Bit 4 = 0: auto-tuning not executed; bit 4 = 1: auto-tuning executed (The current version does not provide auto-tuning for the moment.) Bit 5 = 0: upper/lower output limits invalid; bit 5 = 1: upper/lower output limits valid Bit6 to bit15: unavailable Do not set both bits 5 and 2 to ON.
S4+2	Input filter constant (α)	Value range: 0 to 99, in percent. When it is set to 0, no input filter is processed.
S4+3	Proportional gain (Kp)	Value range: 1 to 32767, in percent.
S4+4	Integral time (T1)	Value range: 0 to 32767 (x 100 ms). When it is set to 0, it is processed as ∞ (no integral).
S4+5	Differential gain (Kd)	Value range: 0 to 100, in percent. When it is set to 0, no differential gain is processed.
S4+6	Differential time (Td)	Value range: 0 to 32767 (x 10 ms). When it is set to 0, no differential is processed.
S4+(7-19)	Occupied by the internal processing of PID calculation. Clear these units before initial running.	
When bit 1 = 1 and bit 2 or bit 5 = 1 in <ACT>, S4+(20-24) are occupied and defined as follows:		
S4+20		Value range: 0 to 32767. This parameter is valid when bit 1 = 1 in <ACT>.

Unit	Function	Description
	Input variable (incremental) alarm value	
S4+21	Input variable (decremental) alarm value	Value range: 0 to 32767. This parameter is valid when bit 1 = 1 in <ACT>.
S4+22	Output variable (incremental) alarm value	Value range: 0 to 32767. This parameter is valid when bit 2 = 1 and bit 5 = 0 in <ACT>.
		Value range of the upper output limit: -32768 to +32767. This parameter is valid when bit 1 = 0 and bit 5 = 1 in <ACT>.
S4+23	Output variable (decremental) alarm value	Value range: 0 to 32767. This parameter is valid when bit 2 = 1 and bit 5 = 0 in <ACT> of S4+1.
		Value range of the lower output limit: -32768 to +32767. This parameter is valid when bit 1 = 0 and bit 5 = 1 in <ACT>.
S4+24	Alarm output	Bit 0 input variable (incremental) overflow Bit 1 input variable (decremental) overflow Bit 2 output variable (incremental) overflow Bit 3 output variable (decremental) overflow This parameter is valid when bit 1 = 1 or bit 2 = 1 in <ACT>.
S4+(25-100)	Occupied by the internal processing of PID calculation	

Mode 1: Position PID Instruction

The following table lists the functions and setting methods of the parameters in each unit.

Address	Name	Value range	Meaning
S4 + 0	Sampling cycle	1 to 32767, in ms	PID calculation cycle; default value: 10
S4 + 1	Control mode	-	0: Forward (default) 1: Reverse
S4 + 2	Proportional gain Kp1	0 to 32767, in percent	Proportional gain; default value: 0
S4 + 3	Integral gain Ki1	0 to 32767, in percent	Integral gain; default value: 0
S4 + 4	Differential gain Kd1	0 to 32767, in percent	Differential gain; default value: 0
S4 + 5		0 to 32767	

Address	Name	Value range	Meaning
	Deviation dead zone		0: Disabled Non-0: Deviation is zero if the deviation value is less than the specific value. Default value: 0 Default value: 0
S4 + 6	Upper output limit	-32768 to +32767	Maximum output value
S4 + 7	Lower output limit	-32768 to +32767	Minimum output value
S4 + 8	Upper integral limit	-32768 to +32767	Maximum cumulative integral value ※1
S4 + 9	Lower integral limit	-32768 to +32767	Minimum cumulative integral value ※1
S4 + 10	Cumulative integral	-	32-bit floating-point number
S4 + 11			
S4 + 12	Last output	-32768 to +32767	Used for differential calculation
S4 + 13	Kp2	0 to 32767, in percent	Default value: 0
S4 + 14	Ki2	0 to 32767, in percent	Default value: 0
S4 + 15	Kd2	0 to 32767, in percent	Default value: 0
S4 + 16	Parameter switching condition	-	0: No switching 1: Switching based on deviation 2: User-defined ※2
S4 + 17	Lower deviation limit E1	-32768 to +32767	Deviation starting point or user-defined switching starting point
S4 + 18	Upper deviation limit E2	-32768 to +32767	Deviation end point or user-defined switching end point
S4 + 19	User-defined switching reference	-32768 to +32767	Switching reference when the parameter switching condition is set to 2
S4 + (20 -100)	Occupied by internal operation	-	-

- 1: When the upper and lower integral limits are set to 0, the upper limit +32,737 and lower limit -32,768 take effect.

- 2: When (S4+16) = 0, (S4+17) to (S4+19) are invalid.

Principle of Position-type PID Calculation

PID calculation formula:

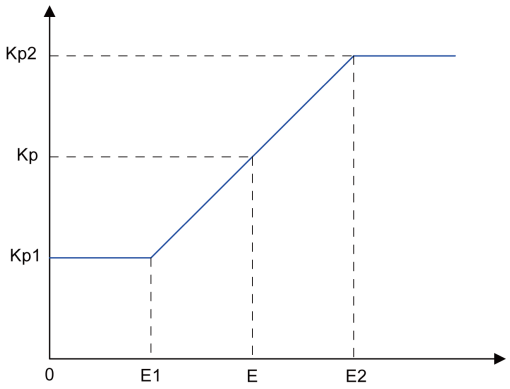
$u(k) = Kp \times e(k) + Ki \times T \times \sum e(i) + (Kd/T) \times [Pv(k) - Pv(k-1)]$

u(k)	Current output value	Pv(k-1)	Feedback value at the last time point
e(k)	Current deviation	T	Sampling time
∑e(i)	Current cumulative integral	Kp	Proportional gain
Sv(k)	Current value	Ki	Integral gain
Pv(k)	Current feedback value	Kd	Differential gain

Forward direction: e(k) = Sv(k) - Pv(k)

Reverse direction: e(k) = Pv(k) - Sv(k)

Principles of parameter switching (proportional gain Kp used an as example)



Kp1	(S4 + 2)
Kp2	(S4 + 13)
E1	(S4 + 17)
E2	(S4 + 18)
E	Switching reference

When E ≤ E1, Kp = Kp1.

When E1 < E < E2, Kp = (Kp2 - Kp1) x E/(E2 - E1).

When E ≥ E2, Kp = Kp2.

S4 + 16	0	No switching
	1	E = |Sv - Pv|

	2	E = S4+19

Mode 2: Special PID Instruction (Operation Principles Same as Those of Customized Inovance 307 Series AC Drive)

The following table lists the functions and setting methods of the parameters in each unit.

Address	Name	Value range	Meaning	AC drive parameter	Winding parameter	Unwinding parameter	Wire drawing machine parameters
S4 + 0	Sampling time	1 to 32767, in ms	PID calculation cycle	-	10	10	10
S4 + 1	Mode	-	0: Positive 1: Reverse	-	-	-	-
S4 + 2	Default parameter selection	-	0: No initialization 1: Wire take-up parameters 2: Dynamic wire take-off 3: Wire drawing machine parameters After the instruction is triggered, the address is reset, indicating that the set parameter has been written to the address.	-	1	2	3
S4 + 3	Feedback range setting	0 to 32767	AND feedback range setting	FA-04	1000	1000	1000
S4 + 4	Output range	0 to 32767	Output range	-	10000	10000	10000
S4 + 5	Reverse maximum output	0 to 32767	Reverse maximum output ※1	-	10000	10000	10000
S4 + 6	Output range selection	-	0: Relative to the maximum range 1: Relative to the main output (D+1)	F0-05	0	0	1
S4 + 7	Auxiliary output range	0 to 32767, in percent	Valid when (S4+6) = 1	F0-06	-	-	70

Address	Name	Value range	Meaning	AC drive parameter	Winding parameter	Unwinding parameter	Wire drawing machine parameters
S4 + 8	Proportional gain Kp1	0 to 32767 (0.1%)	Proportional gain; default value: 0	FA-05	100	150	45
S4 + 9	Integral time Ti1	0 to 32767 (0.01s)	Integral gain; default value: 0	FA-06	120	130	200
S4 + 10	Differential time Td1	0 to 32767 (0.001s)	Differential gain; default value: 0	FA-07	150	0	0
S4 + 11	Deviation limit	0 to 32767 (0.1%)	Maximum calculation deviation	FA-09	0	0	0
S4 + 12	Differential limit	0 to 32767 (0.01%)	Maximum differential limit	FA-10	50	-	-
S4 + 13	PID reference change time	0 to 32767, in ms	When this parameter is enabled, the reference value will equal the setting after the specified time has elapsed.	FA-11	5000	0	0
S4 + 14	Proportional gain Kp 2	0 to 32767 (0.1%)	Default value: 0	FA-15	-	-	-
S4 + 15	Integral time Ti 2	0 to 32767 (0.01s)	Default value: 0	FA-16	-	-	-
S4 + 16	Differential time Td2	0 to 32767 (0.001s)	Default value: 0	FA-17	-	-	-
S4 + 17	Parameter switching condition	-	0: Not switched 1: Switching based on deviation 2: User-defined ※2	FA-18	-	-	-
S4 + 18	Lower deviation limit	0 to 32767 (0.1%)	Deviation starting point or user-defined switching starting point	FA-19	-	-	-

Address	Name	Value range	Meaning	AC drive parameter	Winding parameter	Unwinding parameter	Wire drawing machine parameters
S4 + 19	Upper deviation limit	0 to 32767 (0.1%)	Deviation end point or user-defined switching end point	FA-20	-	-	-
S4 + 20	User-defined switching reference	0 to 32767 (0.1%)	Switching reference when the parameter switching condition is set to 2	-	-	-	-
S4 + 21	Initial output	0 to 32767 (0.1%)	Initial value after PID startup	FA-21	0	0	0
S4 + 22	Initial output hold time	0 to 32767, in ms	Time during which the initial value remains unchanged	FA-22	0	0	0
S4 + 23	Output deviation limit	0 to 32767 (0.1%)	Range of every deviation change	-	0	0	0
S4 + (24-100)	Internal operation	-	-	-	-	-	-

Address	Name	Meaning
D1 + 0	Total output	PID calculation component + (D1+1)
D1 + 1	Main output	User-designated main output (AC drive dominant frequency) This value is set to 0 for pure PID.

- ※1: Maximum negative value of PID output. Example: if this parameter is set to 100, the maximum negative output value is -100.
- ※2: See the parameter switching principle of the position-type PID instruction.

PID calculation formula

$u(k) = Kp \{e(k) + T/Ti \times \sum e(i) + Td/T \times [e(k) - e(k-1)]\}$

u(k)	Current output value	∑e(i)	Current cumulative integral

Kp	Proportional gain	T	Sampling time
e(k)	Current deviation	Ti	Integral time
e(k-1)	Deviation at the last time point	Td	Differential time
Sv(k)	Current value	Ki	Integral gain
Pv(k)	Current feedback value	Kd	Differential gain

Forward direction: e(k) = Sv(k) – Pv(k); reverse direction: e(k) = Pv(k) – Sv(k)

For details about parameter switching, see the position-type PID description.

Main output application

When (S4+6) is 0, (D1+1) is forcibly set to 0.

When (S4 + 6) = 1, (S4 + 7) is enabled. The maximum PID component is equal to (S4+7) percent of (D1+1).

Final (D1+0) = PID component + Main output (D1+1)

Mode 3: Temperature Control PID Instruction

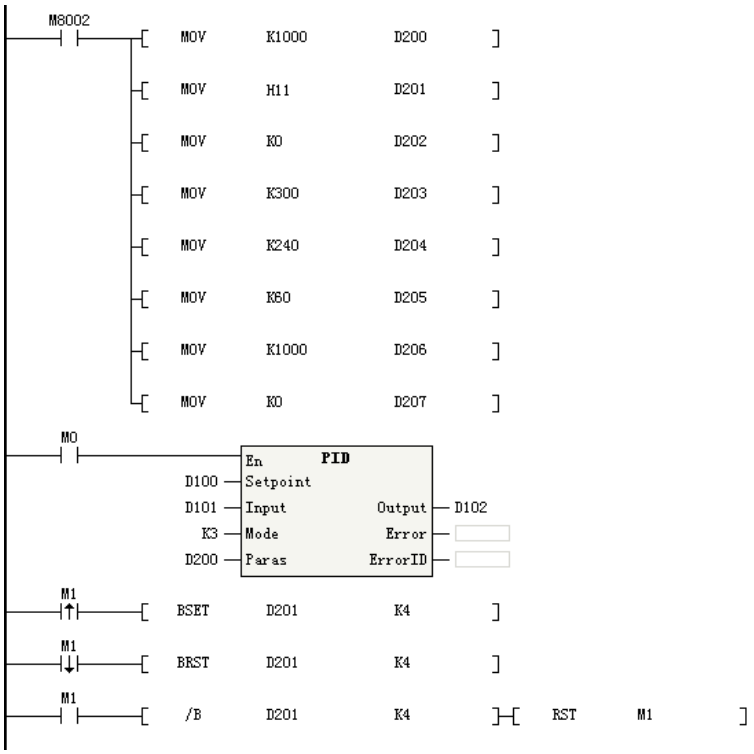
The following table lists the functions and setting methods of the parameters in each unit.

Unit	Function	Description
S4	Sampling period	The sampling time ranges from 1 ms to 32,767 ms and must be greater than the PLC's scan cycle.
S4+1	Mode	Bits 15 to 8: unavailable Bits 7 to 5: unavailable Bit4 = 0: auto-tuning not executed; bit4 = 1: auto-tuning executed, auto reset after auto-tuning is completed Bit1 to bit3: unavailable Bit0 = 0: forward action; bit0 = 1: reverse action
S4+2	Auto-tuning rule	0: common mode, moderate overshoot 1: Slow mode, small overshoot but slow temperature rise 2: Fast mode, fast temperature rise but large overshoot
S4+3	Scaling band	Auto-tuning result scaling band output. The value range is 1 to 32767. The smaller the value of the scaling band, the stronger the scaling effect. After self-tuning, the scaling band will be automatically adjusted to the value after self-tuning.
S4+4	Integral time	Auto-tuning result integral band output. The value range is 0 to 32767, in second. 0 indicates no integral processing.
S4+5	Differential time	Auto-tuning result differential band output. The value range is 0 to 32767, in second. 0 indicates no differential processing.

Unit	Function	Description
S4+6	Upper output limit	The upper output limit ranges from -32768 to +32767 and must be greater than the lower output limit.
S4+7	Lower output limit	The lower output limit ranges from -32768 to +32767 and must be less than sign than the upper output limit.
S4+8	Reserved	Occupied by internal processing of PID calculation
S4+9	Proportional output	Current scaling output
S4+10	Integral output	Current integral output
S4+11	Differential output	Current differential output
S4+(12-100)	Reserved	Occupied by internal processing of auto-tuning calculation

Instruction example:

M0: Temperature control enable/disable; M1: Auto-tuning start/stop (auto reset after auto-tuning is completed)



Mode 4: Special PID Instruction for Air Compressors

CAUTION

- Only one special PID is supported.
- You need to control the relevant logic and process of the air compressor by using the user program, such as determining whether to enable certain functions and controlling parameter modification (the instruction flow needs to be re-controlled) and air compressor feedback.
- Before calling the instruction, you must preset the parameters related to the special PID, including the default values of PID parameters. For example, you can use M8002 to assign values for PID parameters, AC drive parameters, and air compressor parameters. Basically, the control effect is good when the default values are used.

Parameter description:

S1 specifies the set target value of the PID (in 0.01 Mpa).

S2 is the measured feedback value (in 0.01 Mpa). The user program needs to read the actual value of the device and update this parameter.

S4 is the start unit for storing the operation result. It occupies 60 consecutive D elements or 16-bit word variable arrays. S4 is used for user parameter setting and control.

D1 specifies the PID output value (in 0.01 Hz).

The following table lists the functions and setting methods of the parameters in each unit.

Category	Address	Value range	Default	Unit	Description
PID parameter setting, 30 differential parameters	S4+0	-	-	-	For system use, not writable!
	S4+1	-	-	-	For system use, not writable!
	S4+2	0 to 1	1	-	Whether there is an air tank 0: No 1: Yes
	S4+3	0 to 1	0	-	Model coefficient calculation enable 0: Hold 1: Re-calculate
	S4+4	0 to 1	0	-	PID direction 0: Forward 1: Reverse
	S4+5	0 to 10000	3000	ms	Control pressure filter time, setting not required
	S4+6	20 to 60	45	%	

Category	Address	Value range	Default	Unit	Description
					Model switching frequency percentage
	S4+7	100 to 30000	1000	ms	PID control calculation interval
	S4+8	0 to 3000	6	s	Motor acceleration/ deceleration and pipe lag time constant
	S4+9	-	-	-	For system use, not writable!
	S4+10	-32767 to +32767	150	0.01	Model proportion coefficient
	S4+11	-32767 to +32767	300	s	Model time constant
	S4+12	0 to 50	25	1	Fast prediction step 1
	S4+13	0 to 30000	1	1	Fast output weighting factor 1
	S4+14	0 to 50	25	1	Fast prediction step 2
	S4+15	0 to 30000	2	1	Fast output weighting factor 2
	S4+16	0 to 50	35	1	Fast prediction step 3
	S4+17	0 to 30000	5	1	Fast output weighting factor 3
	S4+18	0 to 50	40	1	Fast prediction step 4
	S4+19	0 to 30000	15	1	

Category	Address	Value range	Default	Unit	Description
					Fast output weighting factor 4
	S4+20	-32767 to +32767	150	0.01	Non-air tank model scale factor
	S4+21	-32767 to +32767	30	s	Air tank model time constant
	S4+22	0 to 50	9	1	Non-air tank control step
	S4+23	0 to 30000	5	1	Non-air tank output weighting factor
	S4+24	0 to 50	9	1	Slow non-air tank control step
	S4+25	0 to 30000	5	1	Gas tank-less output weighting factor (slow speed)
	S4+26	-	-	-	For system use, not writable!
	S4+27	-	-	-	For system use, not writable!
	S4+28	-	-	-	For system use, not writable!
	S4+29	-	-	-	For system use, not writable!
Air compressor and AC drive parameter setting (30 differential parameters)	S4+30	1 to 30000	-	0.01 Hz	Maximum frequency of the AC drive
	S4+31	1 to 30000	-	0.01 Hz	Upper output frequency of the AC drive

Category	Address	Value range	Default	Unit	Description
	S4+32	0 to 30000	-	0.01 Hz	Lower output frequency of the AC drive
	S4+33	0 to 30000	-	0.01 Hz	Pre-run frequency of the AC drive
	S4+34	-	-	0.01 Hz	Current running frequency of the AC drive, updated in real time. The user program needs to read the AC drive and update this parameter.
	S4+35	0 to 1000	160	0.01 Mpa	Maximum pressure of the air compressor, used for calibration. The output frequency must be less than the value of this parameter.
	S4+36	-	-	-	For system use, not writable!
	S4+37	-	-	-	For system use, not writable!
	S4+38	-	-	-	For system use, not writable!
	S4+39	-	-	-	For system use, not writable!
	S4+40	-	-	-	For system use, not writable!
	S4+41	-	-	1	Target calibration value

Category	Address	Value range	Default	Unit	Description
	S4+42	-	-	-	For system use, not writable!
	S4+43	-	-	1	Feedback calibration value
	S4+44	-	-	-	For system use, not writable!
	S4+45	-	-	-	For system use, not writable!
	S4+46	0 to the maximum value	-	0.01 Mpa	Pressure feedback value
	S4+47	0 to the maximum value	-	0.01 Mpa	Pressure protection value
	S4+48	0 to the maximum value	-	0.01 HZ	Actual value of the output result
	S4+49	-	-	1	Calibration value of the output result
	S4+50	-	-	-	For system use, not writable!
	S4+51	-	-	-	For system use, not writable!
	S4+52	-	-	-	For system use, not writable!
	S4+53	-	-	-	For system use, not writable!
	S4+54	-	-	-	For system use, not writable!
	S4+55	-	-	-	For system use, not writable!
	S4+56	-	-	-	For system use, not writable!
	S4+57	-	-	-	For system use, not writable!

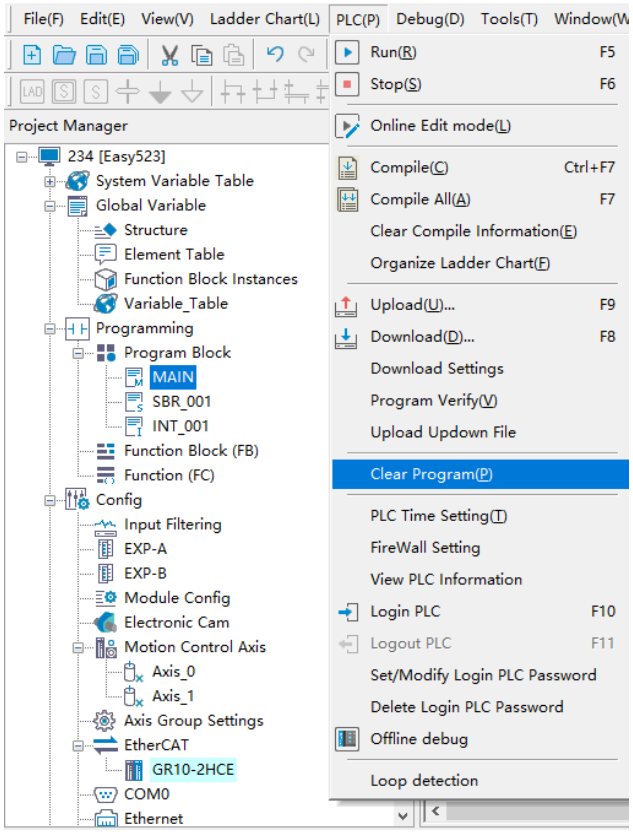
Category	Address	Value range	Default	Unit	Description
	S4+58	-	-	-	For system use, not writable!
	S4+59	-	-	-	For system use, not writable!

Do not occupy 60 consecutive elements in the program.

Mode 5: Large-inertia Temperature Control PID Instruction

Note:

- This instruction is applicable to long-term temperature control applications with large inertia, in which the control cycle is long and the heating time is more than 100s (typically 200s to 500s).
- You are advised to enable this instruction all the time. Before the first startup, ensure that the temperature difference is 100 degrees. You are advised to clear the user program and re-download and start the target program.
  To clear the user program, see the following figure.



- Before calling this instruction, you need to preset relevant parameters. The M8002 is recommended for value assignment. The parameters to configure include the ambient temperature (S4+13) (which is 0 if it is not specified) and output control parameters such as the output status word (S4+7) (which needs to be implemented by the program). As some states (such as the auto-tuning state) and parameters need to be saved and take effect all the time, the S4 parameter area will not be completely cleared.
- Note in the program that the S4 parameters occupy 90 word elements. Do not reuse these 90 word elements in the program.
- Bit0 of the parameter in S4+6 is used by the user to enable and disable temperature control. Pay attention to it in the program.

- S4+7 stores the output bits. You need to associate the output bits with the external temperature control I/O by programming.
- Retentive registers or variables are strongly recommended to store the S4 parameters.

Parameter description:

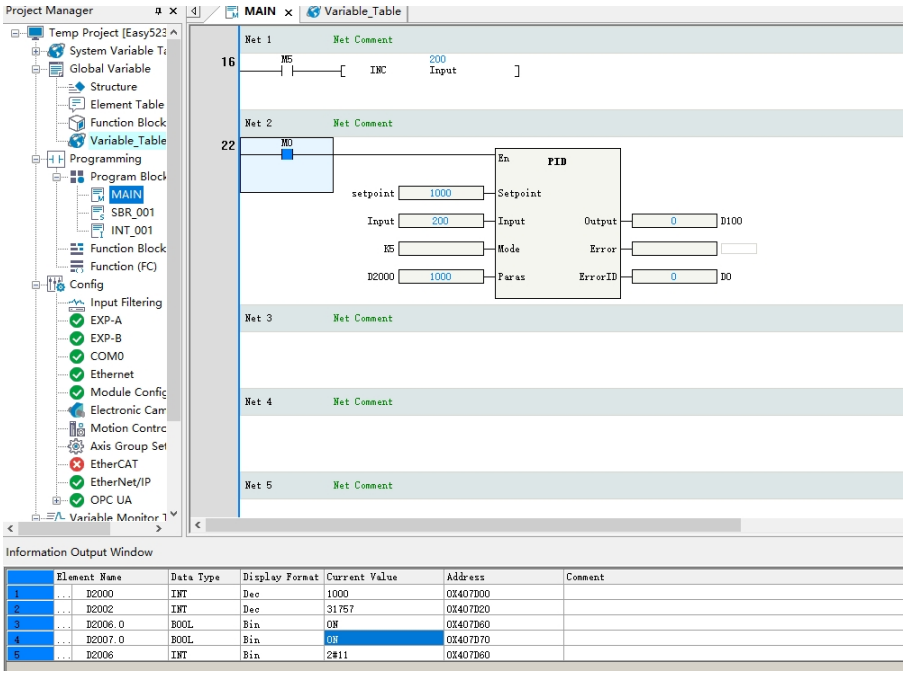
- S1 specifies the set target value of the PID (in 0.1 °C).
- S2 is the measured feedback value (in 0.1 °C). The user program needs to read the actual value of the device and update this parameter.
- S4 is the start unit for storing the operation result. It occupies 90 consecutive D elements or 16-bit word variable arrays. S4 is used for user parameter setting and control.
- D1 specifies the PID output value (in ms).

The following table lists the functions and setting methods of the parameters in each unit.

Category	Address	Value range	Default	Description
PID parameter setting	S4+0	-	-	For system use, not writable!
	S4+1	-	-	For system use, not writable!
	S4+2	-	5000	Sampling time
	S4+3	0 to 4	0	Sampling cycle count
	S4+4	-	-	Updated target value
	S4+5	-	-	Running phase
	S4+6	-	-	Status word: Bit0: Temperature control mode enable (user control required) Bit1 to bit4: State control Bit8: Heat charging area (no control required) Bit9: Linear area (no control required) Bit10: Heat discharging area (no control required) Bit12: Auto-tuning completed Bit13: Auto-tuning in progress Bit14: Heating auto-tuning in progress Bit15: Heat dissipation auto-tuning in progress Other bits: For system use, not writable!
	S4+45	-	-	
	S4+46	-	-	

	S4+47	-	-	
	S4+48	-	-	
	S4+49	-	-	
	S4+50	32-bit data	-	
	S4+52	32-bit data	-	
	S4+54	32-bit data	-	
	S4+56	32-bit data	-	
	S4+58	32-bit data	-	
	S4+60	32-bit data	-	Start time of PID heat dissipation compensation auto-tuning
	S4+62	32-bit data	-	Heating start time during temperature prediction
	S4+64	32-bit data	-	Parameter self-test cycle start time
	S4+66	32-bit data	-	Heat dissipation start time during parameter self-test
	S4+68	32-bit data	-	Reserved
	S4+(70-79)	-	-	PID calculation
	PID parameter output	S4+80	-	ms
		S4+81	-	ms
		S4+82	-	ms
		S4+83	-	ms
S4+84		-	-	Total switch ON duration
S4+85		-	-	Timer for digital control
S4+(86-89)		-	-	Reserved

The following figure shows PID parameter settings for sampling time.



PID errors in different modes are shown in the following table.

Error code	Description
10	Sampling time (TS) less than 0 or out of range
11	Auto-tuning failed
12	Input filter constant object illegal
13	Input scaling coefficient illegal
14	Integral time illegal
15	Differential gain illegal
16	Differential time illegal
17	Range setting error
18	Output period range error
20	Abnormal PID result
21	Offset illegal
22	Scaling item illegal
23	Integral item illegal
24	Differential item illegal
100	Current PID mode not supported

Mode 6: Auto-tuning PID instruction

Note:

- Before calling the instruction, you must pre-set the relevant parameters. It is recommended that you use values assigned by M8002. Main parameters include:
  - PID mode selection
  - Output period
  - Sampling time
  - Max./Min. output percentage
  - Auto-tuning coefficient
  - Scaling coefficient
  - Target temperature
- S4 parameters occupy 100 word elements. Do not reuse these 90 word elements in the program.
- Bit0 of the parameter in S4+2 is used by the user to enable and disable temperature control. Pay attention to it in the program.
- For S4+25, compile a program to associate the output bit with the external temperature control output I/O.
- S4+26 is the percentage of PID digital output. That is, the ratio of output ON time to total output period in percentage.
- For the Setpoint, Mode, and S4 parameters, it is strongly recommended that you use registers retentive upon power failure, such as D components after D200 and R components.

The following table lists the functions and setting methods of the parameters in each unit.

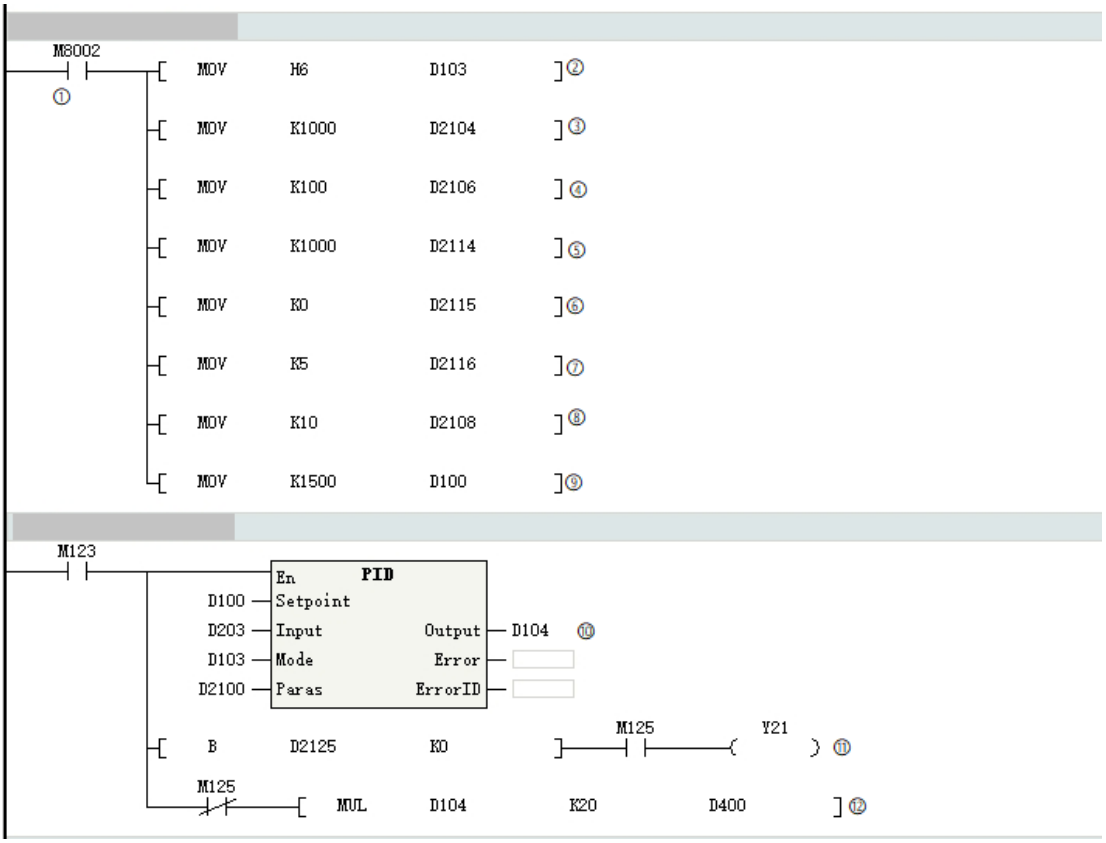
Category	Address	Value range	Default	Unit	Description
PID parameter setting	S4+0	-	-	-	For system use, not writable!
	S4+1	-	-	-	Not used yet
	S4+2	0 to 1	0		Used to start or stop PID. 1: Start, 0: Stop.
	S4+3	0 to 1	0	-	Used to start or stop auto-tuning. 1: Start, 0: Stop.
	S4+4	32-bit data 1 to 1000000	1000	ms	Output period in ms
	S4+6	32-bit data 1 to 1000000	100	ms	Sampling time in ms
	S4+8		10	-	Scaling coefficient Kp, with one decimal place

Category	Address	Value range	Default	Unit	Description
		32-bit data 1 to 1000000			
	S4+10	32-bit data 0 to 1000000	-	0.1s	Integral time Ti, with one decimal place
	S4+12	32-bit data 0 to 1000000	-	0.1s	Derivative time Td, with one decimal place
	S4+14	0 to 1000	100	0.1%	PID maximum output percentage, with one decimal place. If there is no special requirement, it is recommended that you set this parameter to 1000, that is, 100.0%.
	S4+15	0 to 1000	0	0.1%	PID minimum output percentage, with one decimal place. If there is no special requirement, it is recommended that you set this parameter to 0, that is, 100.0%.
	S4+16	1 to 10	5	-	PID auto-tuning coefficient, with one decimal place. The value range is 0.1 to 1.0. The larger the value, the faster the tuning, and the overshoot may be large. If there are no special requirements, it is recommended that you set this parameter to 5, that is, 0.5.
	S4+17	0 to 1	0	-	Used to enable or disable the manual mode. TRUE: Manual mode FALSE: Automatic mode
	S4+18	0 to 1000	0	0.1%	Output percentage in manual mode, with one decimal place.
	S4+19	0	-	0	Bit0=0: Integral separated (default) Bit0=1: Integral not separated Bit1=0: Analog output S4+79, range of 0 to 20000 Bit1=1: Analog output S4+79, range of 0 to 27648

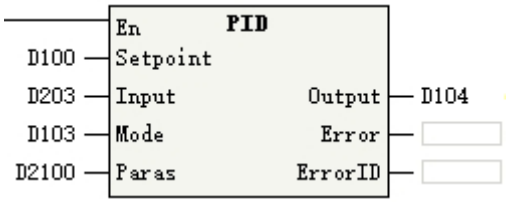
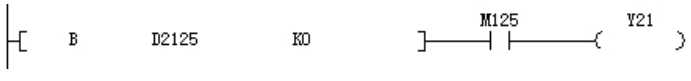
Category	Address	Value range	Default	Unit	Description
					Bit2=0: Fuzzy self-adaption disabled Bit2=1: Fuzzy self-adaption enabled Bit3=0: Default integral settings in auto-tuning Bit3=1: Specified integral settings in auto-tuning Bit4=0: Integral cutting (default) Bit4=1: Integral cutting disabled Bit5=0: Maximum integral limit 75% Bit5=1: Maximum integral limit 100%
	S4+20	0 to 2	0	-	Control mode: 0: Unipolar reverse control (heating control); 1: Unipolar forward control (cooling control); 2: Bipolar control
	S4+21	0 to 10000	-	0.1℃/S	Temperature rise speed per second, fuzzy control used. Obtained through auto-tuning or manual input
	S4+22	0 to 1000	0	0.1%	Cooling AO in percentage (Control mode = bipolar active)
	S4+23	0 to 1000	0	0.1%	Cooling DO in percentage (Control mode = bipolar active)
	S4+24	0 to 1	-	-	DO state: 1: Cooling enabled; 0: Cooling disabled; (Control mode = bipolar active)
PID parameter output	S4+25	0 to 1	-	-	DO state: 1: Heating enabled, 0: Heating disabled
	S4+26	0 to 1000	-	0.1%	DO percentage
	S4+27	-	-	0.1°	Measured value
	S4+28	-	-	0.1°	Setpoint
	S4+29	0 to 1000	-	0.1%	AO percentage
	S4+30	32-bit data	-	0.1	Integral coefficient Ki
	S4+32	32-bit data	-	0.1	Differential coefficient Kd
	S4+(34-35)	-	-	-	Reserved

Category	Address	Value range	Default	Unit	Description
	S4+36	32-bit data	-	0.1	Scaling calculation result
	S4+38	32-bit data	-	0.1	Integral calculation result
	S4+40	32-bit data	-	0.1	Differential calculation result
	S4+42	-	-	0.1	Deviation
	S4+43	-	-	0.1	Previous deviation
	S4+44	-	-	-	Reserved
	S4+45	-	-	-	Max. Pv
	S4+46	32-bit data	-	-	Ti time
	S4+79	16-bit data	-	-	Analog output: When Bit1 of S4+19 is 0: range of 0 to 20000 When Bit1 of S4+19 is 1: range of 0 to 27648
	S4+(50–100)	-	-	-	For internal use

Program Example



For program description, see the following table.

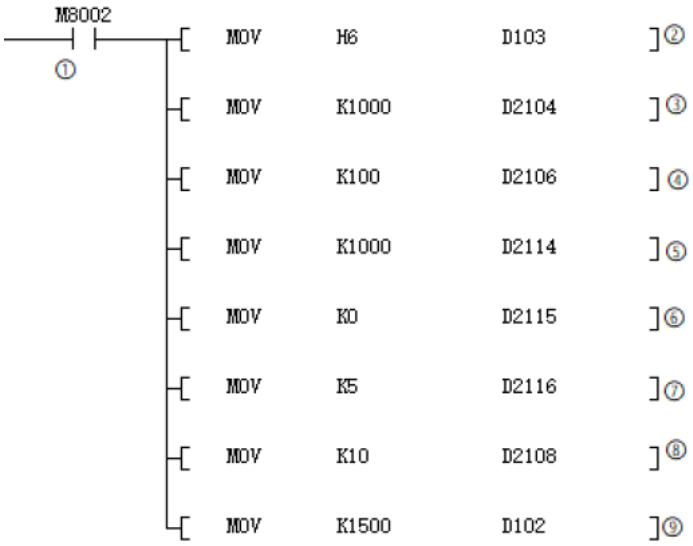
No.	Program	Description
①	M8002	Initialization
②	MOV H6 D103	PID mode selection
③	MOV K1000 D2104	Output period
④	MOV K100 D2106	Sampling time
⑤	MOV K1000 D2114	Maximum output percentage
⑥	MOV K0 D2115	Minimum output percentage
⑦	MOV K5 D2116	Auto-tuning coefficient
⑧	MOV K10 D2108	Scaling coefficient *Kp*
⑨	MOV K1500 D100	Target temperature
⑩		Analog output (AO)
⑪		DO signal assignment
⑫		AO signal assignment

No.	Program	Description
	M125 [MUL D104 K20 D400]	

For the default of Kp, you can first set a value greater than 0. After parameter auto-tuning is completed, set appropriate defaults for Kp, Ti and Td according to the setting results.

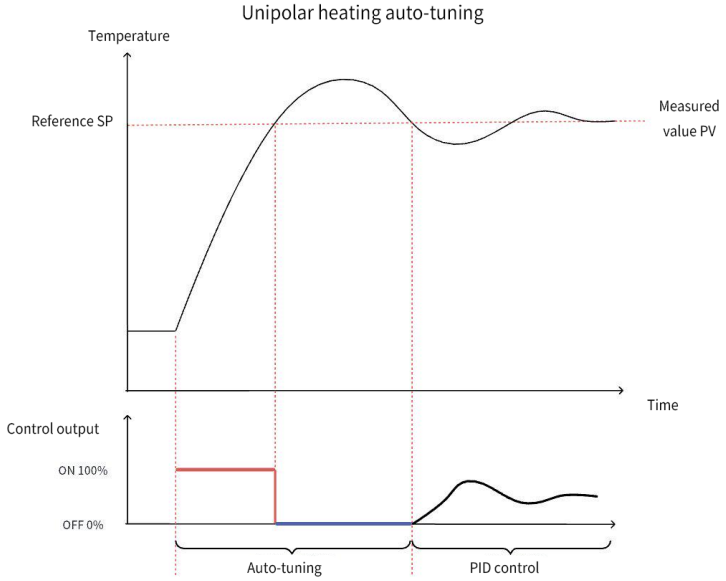
Function Description

- Before running the PID, check the defaults to ensure that they are reasonable, otherwise the program may encounter a running error.



> **NOTE**
>
> For program description, see the preceding tabl

- Enable the auto-tuning function at room temperature (S4+3=1). Enable S4+2 (S4+2=1). When auto-tuning is completed, S4+3 is reset to 0, and you can obtain the set Kp, Ti and Td values. Let the program continue to run, and the controlled object is getting into the steady state. The following figure shows the auto-tuning process.

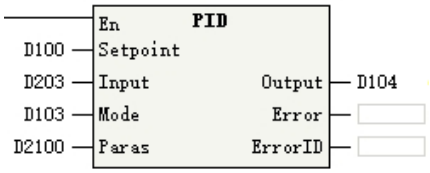


- If the current usage needs are met after the controlled object reaches steady state, record the values of Kp, Ti, and Td, and write these values into the program as defaults.
- You can manually adjust the Kp, Ti, and Td values to meet your usage needs.
- Note that if you use the DO output, associate the value of S4 +25 with the DO output.



- If you use the AO output, associate the output with the AO output. Sets the analog output range for bit1 of S4+19. If bit1 is 0, that indicates Inovance analog module output is selected by default with a range of 0 to 20000. If bit1 is 1, that indicates Siemens analog module is selected with a range of 0 to 27648. If other analog ranges have to be linked, use S4+29 for conversion.

S4+29	0 to 1000	-	0.1%	AO percentage



- The larger the auto-tuning coefficient, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature can be reached, and the temperature curve will be very smooth, without overshoot or with very small overshoot. Unless otherwise required, use the default 5.
  If the customer requires to reach the set temperature faster, increase the auto-tuning coefficient S4+16 and reset it.

  If the customer needs a smooth curve and small overshoot, decrease the auto-tuning coefficient S4+16 and reset it.

  Auto-tuning is not supported for control of pressure, flow, and velocity. You can set bit0 of S4+19 to 1 to disable integral separation. Set Kp to 10, integral to 50, and derivative to 10. You can manually adjust the PID parameters according to the control effect.
- Set bit2 of S4+19 to 1 to enable the fuzzy auto-tuning function.

CAUTION

1. Before running the PID, check whether the parameters are correct.
2. It is recommended that the auto-tuning process starts at the room temperature.
3. If the common temperature range for the customer is 150°C to 200°C, tuning can be carried out at 175°C. It is important to select a common temperature for more accurate tuning parameter.
4. If the parameter of control mode is set to 0 or 1, the output ports are S4+25 and S4+29. The cooling output (S4+24) is valid only when the parameter of control mode is set to 2 (bipolar control).

Error code	Description
10	Sampling time (TS) less than 0 or out of range

Error code	Description
13	Input scaling coefficient illegal
14	Integral time illegal
16	Differential time illegal
17	Parameters setting incorrect

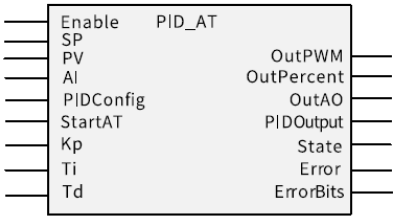
5.20.3 PID_AT

Introduction

The PID_AT instruction performs auto-tuning PID calculation to control the parameters of a close-loop control system.

PID_AT -- Universal PID controller

Graphic Block



The following table describes the instruction format.

Operand	Name	Description	Range	Data Type
16-bit Instruction	-			
32-bit Instruction	Universal PID controller			
S1	SP	Setpoint, in ℃	-	REAL
S2	PV	Feedback value, in ℃	-	REAL
S3	AI	Analog input value	-	DINT
S4	PIDConfig	PID parameter configuration structure	-	_sPID_AT_Config
S5	StartAT	PID parameter auto-tuning enable. After auto-tuning is completed, internal auto reset occurs.	-	BOOL

Operand	Name	Description	Range	Data Type
		TRUE: Enable auto-tuning FALSE: Disable auto-tuning		
S6	Kp	Scale	0.01 to 1000	REAL
S7	Ti	Integral time, in seconds A larger value indicates less integral effect. When the value is 0, there is no integral action.	0.00 to 10,000	REAL
S8	Td	Differential time, in seconds A larger value indicates larger differential effect. When the value is 0, there is no differential action.	0.00 to 10,000	REAL
D1	OutPWM	PWM output TRUE: Enabled FALSE: Disabled	-	BOOL
D2	OutPercent	Output percentage (selected based on output types)	0.00 to 100	REAL
D3	OutAO	Analog output value	-	DINT
D4	PIDOutput	Output variable structure	-	_sPID_Output
D5	State	Operation state PID operation state Options: 0: Disable 1: Manual 2: Auto	-	DINT

Operand	Name	Description	Range	Data Type
		3: Interlock 4: Auto-tuning 5: Error value output 6: Alternative output for the incorrect security value 7: Master tracking		
D6	Error	Error flag	-	BOOL
D7	ErrorBits	Error code*1 Bit 0: PV overlimit bit1: Parameter error	-	DINT

NOTE

*1 For details, see 5.20.5 Error Codes.

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	-	-	√	-	-	√
S5	√	√	√	-	-	-	-	-	-
S6	-	-	-	√	√	√	√	-	-
S7	-	-	-	√	√	√	√	-	-
S8	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D2	-	-	-	√	√	√	-	√	-
D3	-	-	-	√	√	√	√	-	-
D4	-	-	-	-	-	√	-	-	√
D5	-	-	-	√	√	√	√	-	-
D6	√[1]	√	√	-	-	-	-	-	-
D7	-	-	-	√	√	√	√	-	-

NOTE

[1]: The X element is not supported.

Instruction Description

CAUTION

- Before running the PID, check whether the parameters are correct.
- It is recommended that the auto-tuning process starts at the room temperature.
- If the common temperature range that you use is 150°C to 200°C, tuning can be carried out at 175°C. It is important to select a common temperature for more accurate parameter tuning.
- In general, auto-tuning can be carried out first to obtain the tuned PID parameters. Then, you can adjust the PID parameters as needed based on the control effect.
- After the project calls and downloads the instruction to the PLC, the project periodically checks whether the input parameters are correct even if the instruction is not triggered.

Structure Description

The following table lists structures of the PID_AT_Config.

Variable Name	Type	Range	Default	Unit	Description
GeneralConfig	PID_GeneralConfig	-	-	-	PID general configuration
InputConfig	PID_InputConfig	-	-	-	PID input configuration
OutputConfig	PID_OutputConfig	-	-	-	PID output configuration

Variable Name	Type	Range	Default	Unit	Description
SuperConfig	PID_SuperConfig	-	-	-	PID advanced configuration

The following table describes variables in the PID_GeneralConfig structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
SampleTime	DINT	1	1,000,000	100	ms	Heating sampling period
CycleTime	DINT	1	1,000,000	1000	ms	Heating output period
PIDType	DINT	0	3	0	-	0: PID 1: PI 2: PD 3: ID
ControlMode	DINT	0	1	0	-	0: Reverse (unipolar heating) 1: Forward (unipolar cooling)
ManualEnable	BOOL	FALSE	TRUE	FALSE	-	Manual mode switch
ManualValue	REAL	OutMin	OutMax	0.0	%	Manual mode output. The value ranges from -100 to +100 when cooling is enabled.
ATCoefficient	REAL	0.0	1.0	0.5	-	Auto-tuning coefficient
DelayStartUp	DINT	0	1,000,000	0	ms	Delayed start time. PID is started after delay of the set time for off-peak start in case of multi-channel temperature control on the device.
DelayShutdown	DINT	0	1,000,000	0	ms	Delayed shutdown time. PID is shut down after delay of the set time.
CAS_PV (input)	REAL	-	-	0	-	Process value of slave circuit When the slave circuit is detected as non-cascade automatic mode, the output of the master circuit tracks the process value of the slave circuit.
CAS_OVE (input)	INT	0	2	0	-	Overlimit status of slave circuit 0: Normal 1: Upper limit reached 2: Lower limit reached

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
CAS_State (input)	INT	0	1	0	-	Cascade use mode 0: Slave non-automatic mode 1: Slave automatic mode
IsMaster	BOOL	0	1	0		Master control 0: No 1: Yes
IsSlave	BOOL	0	1	0	-	Slave control 0: No 1: Yes
PhysicalQuantity	INT	0	1	0	-	Physical object to control 0: General 1: Temperature
PhysicalUnit	DINT	-	-	0	-	Unit of the physical object (reserved)
ATNum	DINT	0	20	2		Number of auto-tuning waves (applicable to only unipolar) 0: Reserved 1 to 20: Number of shockwaves
ATOut	REAL	rOutMin	rOutMax	100	%	Auto-tuning output percentage (applicable to only unipolar)

The following table describes variables in the PID_InputConfig structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
PVType	DINT	0	1	0	-	0: Use the current PV as the input 1: Use the peripheral analog as the input and convert it based on the analog range
AIMin	DINT	-	AIMax	0	-	Minimum value for analog conversion
AIMax	DINT	AIMin	-	20,000	-	Maximum value for analog conversion
PVMin	REAL	-	PVMax	0.0	-	Minimum value of PV
PVMax	REAL	PVMin	-	100.0	-	Maximum value of PV

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
PVLowLimit	REAL	-	PVHighLimit	-100.0	-	Lower alarm limit for PV
PVHighLimit	REAL	PVLowLimit	-	1000.0	-	Higher alarm limit for PV
ErrorAction	BOOL	FALSE	TRUE	FALSE	-	Action for error alarm FALSE: Remain the value upon an error alarm TRUE: Output the alternative value upon an error alarm
ErrorOut	REAL	OutMin	OutMax	0	-	Output of an alternative percentage upon an error alarm
PVUnit	DINT	0	10	0	-	Reserved interface 0: °C 1: %
InlockSW	BOOL	0	1	0	-	Interlock switch
InlockVal	REAL	-100	100	0	-	Interlock security value
SPSel	BOOL	0	1	0	-	Setpoint source 0: External (SP configuration) 1: Internal (panel operation)
SPRatio	REAL	0	-	0.0	-	SP setpoint change
SPInt	REAL	-	-	0.0	-	Internal setpoint (reserved)
SPTrack	BOOL	0	1	1	-	Internal setpoint tracking switch (reserved) Default: TRUE

The following table describes variables in the PID_OutputConfig structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
OutType	DINT	0	2	0	-	Output type: 0: PWM output 1: Analog output 2: Percent output
OutMin	REAL	-100.0	OutMax	0.0	%	Minimum value of output percentage

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
OutMax	REAL	OutMin	100	100	%	Maximum value of output percentage
AOMin	DINT	-	AOMax	0	-	Minimum value of analog output
AOMax	DINT	AOMin	-	20,000	-	Maximum value of analog output
OutputOffset	REAL	-100.0	100.0	0.0	%	Output deviation compensation value
MinPT	DINT	0	10,000	0	ms	Minimum pulse time

The following table describes variables in the PID_SuperConfig structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
Fuzzy	BOOL	FALSE	TRUE	FALSE	-	FALSE: Disable fuzzy self-adaption TRUE: Enable fuzzy self-adaption
IntegralSuspend	BOOL	FALSE	TRUE	FALSE	-	FALSE: Enable integral TRUE: Pause integral
IntegralDivision	BOOL	FALSE	TRUE	FALSE	-	FALSE: Integral separated TRUE: Integral not separated
IntegralClear	BOOL	FALSE	TRUE	FALSE	-	FALSE: Normal integral TRUE: Clear integral
ECMax	REAL	-	-	0.0	-	Maximum temperature rise speed per second
KdGain	REAL	0.0	1.0	0.65	-	Incomplete differential coefficient A larger value indicates smoother differential effect. When the value is 1, there is no differential action.
IntegralMax	REAL	-1000.0	100.0	75.0	%	Maximum integral value
IntegralCut	REAL	0.0	1.0	0.75	-	Integral cutting coefficient for setpoint change
IntegralDivLimit	REAL	0.0	100.0	50	-	Integral separation limit
DeadBand	REAL	0.0	100.0	0	-	Deviation dead zone limit

The following table describes variables in the PID_Output structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
PVOut	REAL	-	-	-	-	PV value converted from analog
ECMaxOut	REAL	-	-	-	-	Maximum temperature rise speed per second
ATState	DINT	-	-	-	-	Auto-tuning status 0: No auto-tuning 1: During auto-tuning 2: Auto-tuning completed
ErrorID	DINT	-	-	-	-	Error code
Debug	REAL*9	-	-	-	-	Commissioning parameter
CAS_PV	REAL	-	-	-	-	Process value of slave circuit When the slave circuit is detected as non-cascade automatic mode, the output of the master circuit tracks the process value of the slave circuit.
CAS_OVE	INT	-	-	-	-	Overlimit status of slave circuit 0: Normal 1: Upper limit reached 2: Lower limit reached
CAS_State	DINT	-	-	-	-	Cascade use mode 0: Slave non-automatic mode 1: Slave automatic mode

NOTE

1. Enable the auto-tuning function at room temperature. When auto-tuning is completed, StartAT is automatically set to FALSE, and you can obtain the tuned values of Kp, Ti, and Td. Let the program continue to run and check whether the controlled object can reach steady state.
2. If the controlled object reaches the steady state, record the values of Kp, Ti, and Td, and write these values into the program as defaults. You can also manually adjust the values of Kp, Ti, and Td to meet your needs.
3. The larger the auto-tuning coefficient specified by ATCoefficient, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature can be reached, and the temperature curve will be smooth, without overshoot or with small overshoot. Unless otherwise specified, it is recommended to use the default 0.5. If you want to reach the set temperature faster, increase the auto-tuning coefficient and perform auto-tuning again. If you need a smooth curve and small overshoot, decrease the auto-tuning coefficient and perform auto-tuning again.

Parameter Description

- Enable

PID Calculation	Value	Operation
Start	TRUE	Obtain the measured value of PV, perform PID calculation, and output the result in turn.
End	FALSE	Clear the output percentage (), PWM output (OutPWM), and analog output (OutAO).

- StartAT
  This instruction provides the auto-tuning function to automatically calculate the optimal values for PID constants Kp, Ti, and Td.

  The auto-tuning execution conditions upon heating are as follows:

  - The current value is less than the setpoint. In general, the setpoint is the common operating temperature.
  - StartAT
    The following table describes the StartAT parameter.

Value of StartAT	Value of Enable	PID Calculation	Auto-tuning Started	Auto-tuning Completed	Operation
TRUE	Change from FALSE to TRUE	-	The value of ATState.1 changes to 1.	The value of ATState.1 changes to 0.	Auto-tuning is performed when instruction execution starts.

Value of StartAT	Value of Enable	PID Calculation	Auto-tuning Started	Auto-tuning Completed	Operation
Change from FALSE to TRUE	TRUE	Yes	-	The value of ATState.2 changes to 1.	Auto-tuning is performed during instruction execution.
Change from TRUE to FALSE	-	-	-	ATState.1 = 0 ATState.2 = 0	-

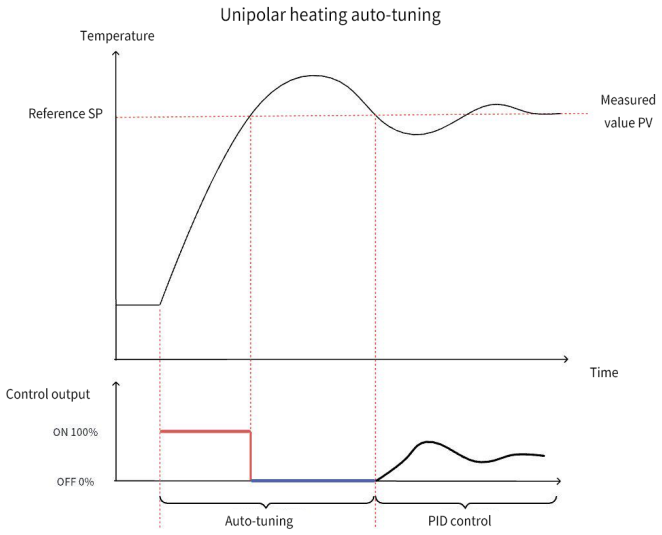
CAUTION

After auto-tuning is completed, StartAT is automatically set to FALSE.

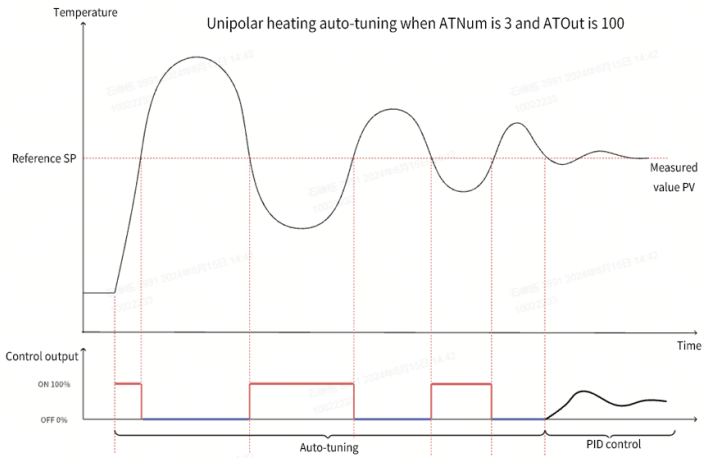
- ATOut specifies the heating or cooling output percentage in the auto-tuning stage. This parameter is available only in unipolar mode.

  ATNum specifies the number of shockwaves based on which auto-tuning is performed.

- The following figure shows the unipolar tuning process when ATOut is set to 100 and ATNum is set to 1.



- The following figure shows the unipolar tuning process on three shockwaves when ATOut is set to 100 and ATNum is set to 3.



The following table describes parameters related to the tuning process.

Parameter	Value	Not Satisfied	ErrorID	Description
Scaling coefficient Kp	> 0	The value of Error is TRUE.	Error code reported	This parameter can be set by yourself or automatically obtained through auto-tuning. A larger value indicates a stronger scaling action.
Integral time Ti	≥ 0 If the value is 0, there is no integral action.	The value of Error is TRUE.	Error code reported	This parameter can be set by yourself or automatically obtained through auto-tuning. A larger value indicates a weaker integral action.
Differential time Td	≥ 0 If the value is 0, there is no differential action.	The value of Error is TRUE.	Error code reported	This parameter can be set by yourself or automatically obtained through auto-tuning. A larger value indicates a stronger differential action.

- PID_Config
  Includes:

  GeneralConfig, InputConfig, OutputConfig, and SuperConfig

  - GeneralConfig
    The following table describes parameters in the PID_GeneralConfig structure.

PID_GeneralConfig Parameter Type	Description	Remarks
GeneralConfig.SampleTime Sampling period	The current PV value is obtained once a sampling period for one PID calculation. In general, the sampling period of the temperature collection module is used. Unless otherwise specified, the default value 100 ms can be used.	-
GeneralConfig.CycleTime Output period	Output period specified by HeatPWM. If the HeatPercent output value of the PID is 80% and the value of GeneralConfig.CycleTime is 10000 ms, the period for HeatPWM is 10000 ms, with 8s on and 2s off.	-
GeneralConfig.PIDType Type	Type of the current PID 0: PID calculation 1: PI calculation 2: PD calculation 3: ID calculation	Perform corresponding calculation control based on the selected PID type. Default: 0: PID calculation
GeneralConfig.ControlMode Control mode	0: Reverse (unipolar heating) 1: Forward (unipolar cooling)	-
GeneralConfig.ManualEnable Manual mode switch	0: Automatic mode 1: Manual mode	When the manual mode is enabled, output the value of GeneralConfig.ManualValue. The manual mode can be changed to the automatic mode through unperturbed switchover.
GeneralConfig.ATCoefficient Auto-tuning coefficient	For the auto-tuning coefficient, the larger the auto-tuned Kp value, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature after auto-tuning can be reached, and the temperature curve will be smooth, without overshoot or with small overshoot.	Use the default value 0.5 for initial parameter tuning. If you want to reach the set temperature faster, increase the auto-tuning coefficient GeneralConfig.ATCoefficient based on the control effect and perform auto-tuning again. If you need a smooth curve and small overshoot, decrease the auto-tuning coefficient GeneralConfig.ATCoefficient

PID_GeneralConfig Parameter Type	Description	Remarks
		and perform auto-tuning again.
GeneralConfig.DelayStartUp Delayed start time	Start the PID after the set delay time.	Different delayed start time can be set for off-peak start in case of multi-channel temperature control on the device. This prevents a large current fluctuation during multi-channel start.
GeneralConfig.DelayShutdown Delayed shutdown time	Delayed shutdown time. PID is shut down after delay of the set time.	-
GeneralConfig.PhysicalQuantity Physical object to control	Physical object to control. The options are as follows: 0: General. This option is mainly used for pressure and flow control. When this option is selected, you can specify the advanced parameters IntegralDivision and IntegralDivLimit for integral separation. 1: Temperature. Various optimization methods are available for temperature control.	-
GeneralConfig.ATNum Number of auto-tuning waves	After you specify this parameter, auto-tuning is performed on the specified shockwaves.	This parameter is supported only in unipolar mode.
GeneralConfig.ATOut Auto-tuning output percentage	This parameter specifies the heating or cooling output percentage in the auto-tuning stage.	This parameter is supported only in unipolar mode.

Value range of GeneralConfig.ManualValue:

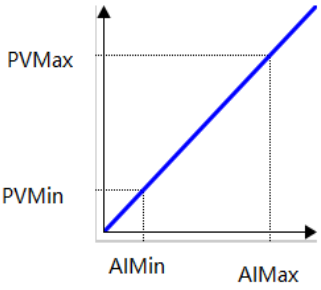
Between the minimum output and the maximum output

The following table describes the minimum output and the maximum output of GeneralConfig.ManualValue.

Control Mode	Minimum	Maximum
0	OutputConfig.OutMin	OutputConfig.OutMax
1	OutputConfig.OutMin	OutputConfig.OutMax

◦ InputConfig

The following table describes parameters in the PID_InputConfig structure.

<table>
<tr><th>PID_InputConfig Parameter Type</th><th>Description</th><th>Remarks</th></tr>
<tr><td>InputConfig.PVType
Type of sampling PV</td><td>0: Use the current PV as the input
1: Use the peripheral analog as the input</td><td>Perform conversion based on the analog range specified by InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax.</td></tr>
<tr><td>InputConfig. AIMin
Minimum value for analog conversion</td><td rowspan="2">Use the peripheral analog as the input and perform conversion based on the analog range specified by InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax.</td><td rowspan="2">They are used to convert an analog to a PV value.
 
</td></tr>
<tr><td>InputConfig.AIMax
Maximum value for analog conversion</td></tr>
<tr><td>InputConfig.PVMin</td><td>Minimum PV value corresponding to InputConfig.AIMin</td><td rowspan="2">Use the peripheral analog as the input and perform conversion based on the analog range specified by InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax.</td></tr>
<tr><td>InputConfig.PVMax</td><td>Maximum PV value corresponding to InputConfig.AIMax</td></tr>
<tr><td>InputConfig.Input Min Lower input limit alarm</td><td>When the PV value or the PV value converted from the analog is lower than the value of InputConfig.InputMin, an alarm is generated for exceeding the lower limit.</td><td rowspan="2">Alarms are generated based on InputConfig.ErrorAction and ErrorOut.</td></tr>
<tr><td>InputConfig.Input Max Upper input limit alarm</td><td>When the PV value or the PV value converted from the analog is higher than the value of InputConfig.InputMax, an alarm is generated for exceeding the upper limit.</td></tr>
<tr><td>InputConfig. ErrorAction Action upon an error alarm</td><td rowspan="2">FALSE: Remain the output value upon an error alarm
TRUE: Output the alternative value InputConfig.ErrorOut upon an error alarm</td><td rowspan="2">-</td></tr>
<tr><td>InputConfig.ErrorOut Error alarm</td></tr>
<tr><td>InputConfig.SPSel Setpoint source</td><td>Select the SP value used for PID calculation.
0: External (SP configuration)</td><td>-</td></tr>
</table>

PID_InputConfig Parameter Type	Description	Remarks
	1: Internal (panel operation)	
InputConfig.SPRatio SP setpoint change	If SPRatio is greater than 0, the final value turns closer to the setpoint based on the change of SPRatio. This parameter is calculated once every task period.	-

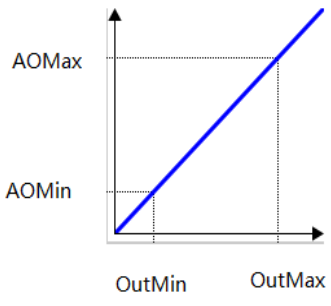
Value range of InputConfig.ErrorOut

The following table describes the minimum output and the maximum output of InputConfig.ErrorOut.

Control Mode	Minimum	Maximum
0	OutputConfig.OutMin	OutputConfig.OutMax
1	OutputConfig.OutMin	OutputConfig.OutMax

◦ OutputConfig

The following table describes parameters in the PID_OutputConfig structure.

PID_OutputConfig Parameter Type	Description	Remarks
OutputConfig.OutType Output type	0: PWM output. OutPercent indicates the percentage for PWM output. 1: Analog output. OutPercent indicates the percentage for analog output.	-
OutputConfig.OutMin Minimum output percentage	Convert the output percentage of PID calculation based on the analog range specified by OutputConfig.OutMin and OutputConfig.OutMax as well as the corresponding OutputConfig.AOMin and OutputConfig.AOMax.	They are used to limit the output of PID calculation and convert the analog output. 
OutputConfig.OutMax Maximum output percentage		
OutputConfig.AOMin Minimum value of analog output	Convert the output percentage of PID calculation based on the analog range specified by OutputConfig.OutMin and OutputConfig.OutMax as well as the corresponding OutputConfig.AOMin and OutputConfig.AOMax.	They are used to convert the analog output.
OutputConfig.AOMax Maximum value of analog output		

PID_OutputConfig Parameter Type	Description	Remarks
OutputConfig.OutputOffset Output deviation compensation value	OUT = PID calculation value + OutputOffset The final output is equal to the PID calculation value plus the value of OutputOffset. They are used feedforward connection and compensation.	-
OutputConfigHeat.MinPT Minimum pulse time	This parameter specifies the minimum pulse time during which PWM is enabled or disabled, in milliseconds. The pulse is not enabled when the enable time is less than that specified by MinPT. The pulse is not disabled when the disable time is less than that specified by MinPT.	-

◦ SuperConfig

The following table describes parameters in the PID_SuperConfig structure.

PID_SuperConfig Parameter Type	Description	Remarks
SuperConfig.Fuzzy Fuzzy self-adaption enable	0: Disable fuzzy self-adaption 1: Enable fuzzy self-adaption	PID parameters can be dynamically adjusted based on fuzzy self-adaption (available for unipolar heating).
SuperConfig.ECMax Maximum temperature rise speed per second	This parameter is used in fuzzy self-adaptive mode. The maximum temperature rise speed of the device can be specified by yourself or obtained through auto-tuning (PIDOutput.ECMaxOut).	-
SuperConfig.KdGain Incomplete differential coefficient	The value ranges from 0.0 to 1.0. A larger value indicates smoother differential effect. When the value is 1, there is no differential action.	-
SuperConfig.IntegralSuspend Integral pause	0: Enable integral (default) 1: Pause integral	-
SuperConfig.IntegralDivision Integral separation	0: Integral separated (default) 1: Integral not separated	-
	0: Normal integral (default)	-

PID_SuperConfig Parameter Type	Description	Remarks
SuperConfig.IntegralClear Integral clear	1: Integral clear	
SuperConfig.IntegralMax Maximum integral value	The maximum integral value is 75% by default.	-
SuperConfig.IntegralCut Setpoint	Integral cutting coefficient for setpoint change The integral cutting coefficient is 0.75 by default.	-
SuperConfig.Reserve Reserved	Reserved	-
SuperConfig.IntegralDivLimit Integral separation limit	When the deviation is greater than the value of IntegralDivLimit, no integration is performed.	-
SuperConfig.DeadBand Deviation dead zone limit	When the deviation of |SP – PV| is less than the value of DeadBand, the deviation is considered as 0 (no PID calculation).	-

- Output structure
  The following table describes parameters in the output structure.

Parameter	Description
PIDOutput.PVOut	Actually calculated PV value
PIDOutput.ECMaxOut	Maximum temperature rise speed per second, auto-tuned
PIDOutput.ATState	Auto-tuning status 0: No auto-tuning 1: During auto-tuning 2: Auto-tuning completed
PIDOutput.ErrorID	Error code. For details, see 5.20.5 Error Codes.

- Cascade function

Cascade control involves two or more cascaded PID controllers. The control variable of the master controller is interconnected with the setpoint of the slave controller, creating a nested control circuit. Cascade control in a slave circuit can compensate for interference in the internal circuit faster than in the slower main circuit.

Note: Cascade control is possible only when there are other measurable variables in the process, and the internal control circuit is much faster than the external circuit.

Unperturbed switchover: When the slave circuit is not in cascade mode (automatic mode by external setpoint), the main circuit switches to the tracking mode. The output value of the main circuit tracks the process value of the slave circuit, allowing for a smooth transition back to the cascade mode.

Priority (Error response > Interlock > Manual > Master tracking mode triggered by slave > Automatic)

- Working modes

Working Mode	Action
Automatic	The output value of the PID module is calculated based on the following rule: scaling, integral, and differential in turn. The calculation result of the PID module is restricted by the output amplitude.
Manual	The output is manually controlled (specified by the operator or in online logic). The output of this mode is restricted by the output amplitude.
Interlock	In interlock mode, PID stops automatic calculation, and the output value of the PID module is a pre-set value of InlockVal, which is generally referred to as the interlock point. The interlock function is also used for tracking. When a certain condition is met, for example, InlockSW is set to TRUE, the PID module automatically switches to the interlock mode. When the interlock condition is not met, the PID module switches back to the previous working mode.

(1) Interlock has the highest priority, followed by manual, and finally automatic.

(2) Automatic mode features much control with fixed setpoint, which can be specified on the configuration interface.

(3) You can choose to use the external setpoint or internal setpoint based on the value of PIDConfig.InputConfig.SPSel.

  - If SPSel is set to FALSE, the external setpoint in the logical configuration is used.
  - If SPSel is set to TRUE, the internal setpoint is used.

- Unperturbed switchover

Unperturbed Switchover Method	Description
Interlock to manual	In interlock mode, the interlock value is directly output. In this case, the manual value tracks the output value to ensure unperturbed switchover.
Interlock to automatic	In interlock mode, the interlock value is directly output. After the interlock mode is switched to the automatic mode, if the external setpoint is used, the logic operation value is directly used for the PID calculation output, which may cause perturbation.
Manual to interlock	In manual mode, the manual value is output. After the manual mode is switched to the interlock mode, the output is the interlock value, and perturbation occurs.
Manual to automatic	In manual mode, the output is the value specified by the operator. If the external setpoint is used, the logic operation value is directly used for the PID calculation output. In this case, the switchover is unperturbed (with the incremental part perturbed).

Unperturbed Switchover Method	Description
Automatic to interlock	In automatic mode, the output is the normal operation value. After the automatic mode is switched to the interlock mode, the output is the interlock value, and perturbation occurs.
Automatic to manual	In automatic mode, the output is the normal operation value. In this case, the manual value tracks the output value to ensure unperturbed switchover.

- Setpoint change limit function

  In automatic mode, the setpoint (external or internal) change is checked. If the change exceeds a preset limit (specified by PIDConfig.InputConfig.SPRatio), the actual setpoint used in the calculation is restricted to the change value and the output gradually approaches the setpoint based on the magnitude of the change.

$$\Delta SP > SPRatio\left(Setpoint\ change\right):\ SP = LastSP + SPRatio$$

$$\Delta SP < SPRatio\left(Setpoint\ change\right):\ SP = LastSP - SPRatio$$

Integral Separation Enable PIDConfig.SuperConfig.IntegralDivision	Integral Separation Limit PIDConfig. SuperConfig.IntegralDivLimit
TRUE	When |Deviation| is greater than or equal to the integral separation limit, no integral component operation is performed. In this case, PD control is used.
	When |Deviation| is less than the integral separation limit, the full integral operation is performed. In this case, PID control is used.
FALSE	The full integral operation is always performed.

  - Integral pause

    If PIDConfig.GeneralConfig.IntegralSuspend is set to TRUE, integral pause is triggered. In this case, the integral action remains unchanged and no more integral accumulation occurs.

  - Integral clear

    If PIDConfig.GeneralConfig.IntegralClear is set to TRUE, integral clear is triggered. In this case, the integral action is cleared to 0.

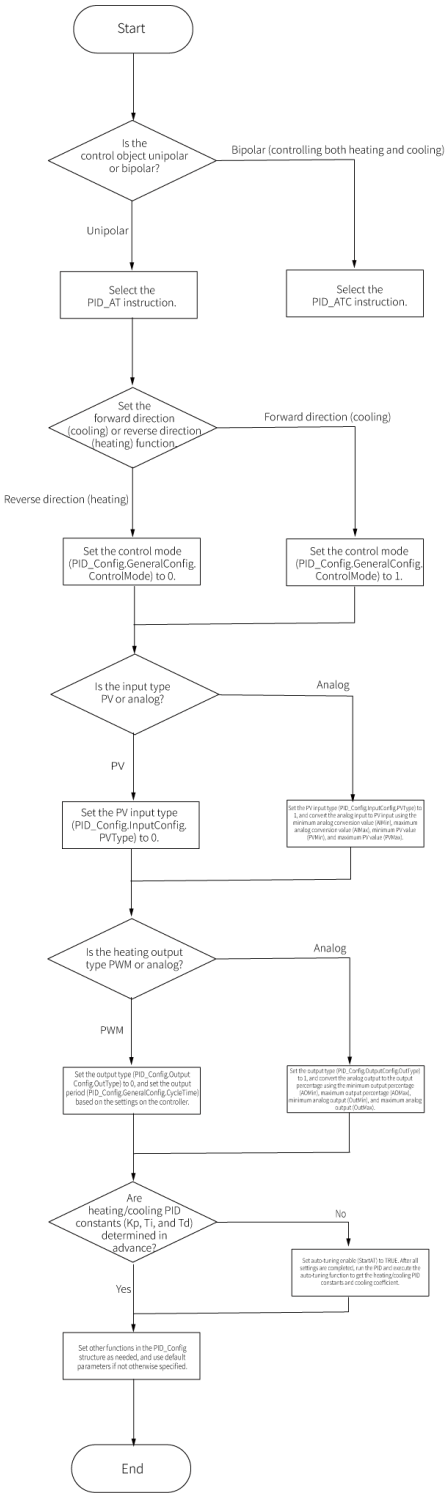
  - Integral cutting coefficient

    When the setpoint changes, you can reduce the initial integral action by specifying PIDConfig.SuperConfig.IntegralCut. The value range is from 0 to 1. The default value is 0.75.

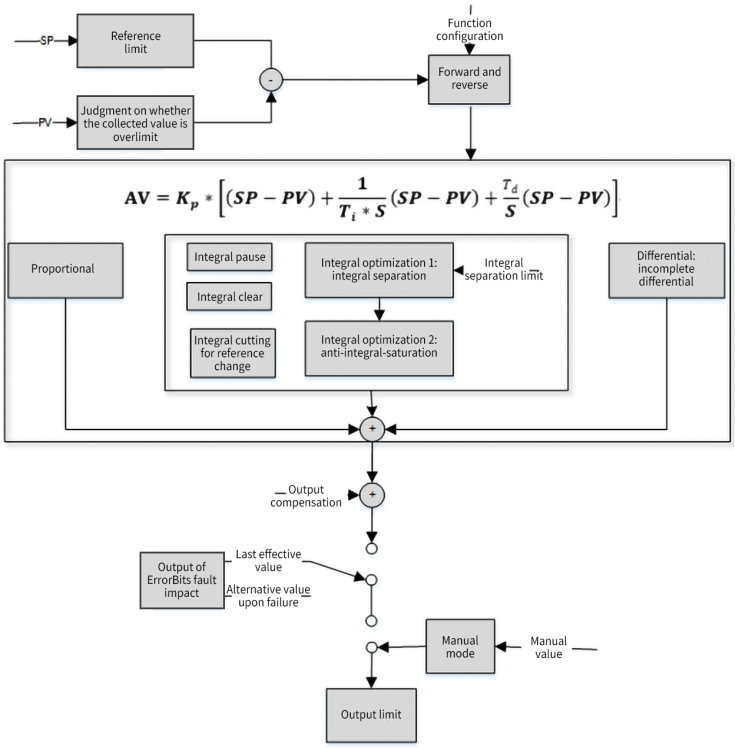
  - Dead zone control function

You can specify PIDConfig.SuperConfig.DeadBand to determine the dead zone width. Within the dead zone range, the adjustment is stable and the PID output does not change. If the value is beyond the dead zone range, PID calculation is normal.

Initial Setting Process



Operating Logic Diagram



- Calculation formula

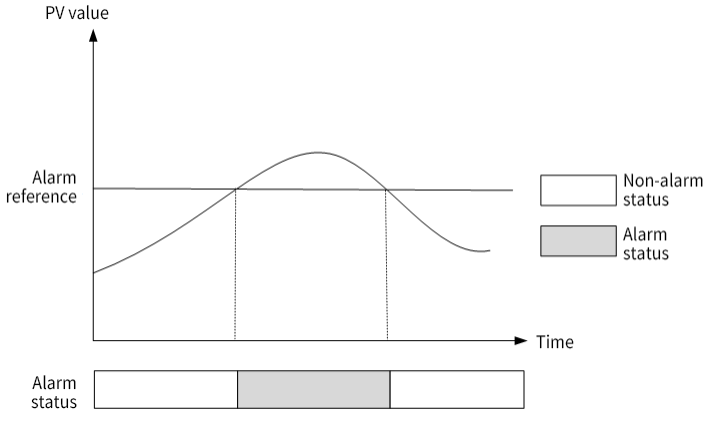
$$\mathbf{AV} = \boldsymbol{K_p} * \left[(\boldsymbol{SP} - \boldsymbol{PV}) + \frac{\mathbf{1}}{\boldsymbol{T_i} * \boldsymbol{S}}(\boldsymbol{SP} - \boldsymbol{PV}) + \frac{T_d}{\boldsymbol{S}}(\boldsymbol{SP} - \boldsymbol{PV})\right]$$

- Discretization formula

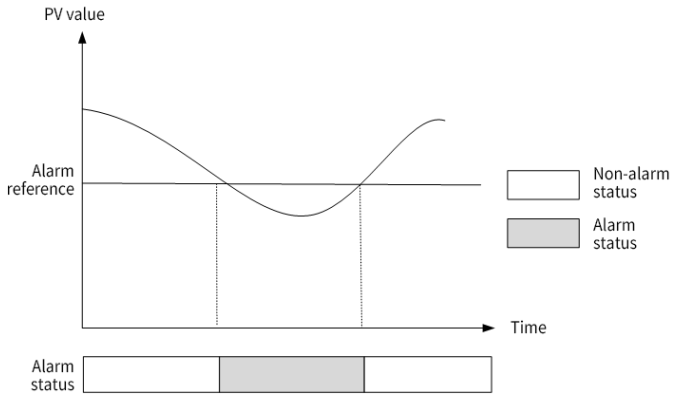
$$\mathbf{AV(k)} = \boldsymbol{K_p} * \mathbf{E}(\boldsymbol{k}) + \boldsymbol{K_p} * \frac{\boldsymbol{cycle}}{\boldsymbol{T_i}} \sum \mathbf{E}(\boldsymbol{k}) + \frac{\boldsymbol{K_p} * \boldsymbol{T_D}}{\boldsymbol{cycle}} (\mathbf{E}(\boldsymbol{k}) - \mathbf{E}(\boldsymbol{k} - \mathbf{1}))$$

Upper and lower limit alarms

- Upper limit alarm of the input PV value
  When the measured value of PV (or the PV value converted from the analog) is greater than the upper limit, an alarm is reported.



- Lower limit alarm of the input PV value
  When the measured value of PV (or the PV value converted from the analog) is less than the lower limit, an alarm is reported.



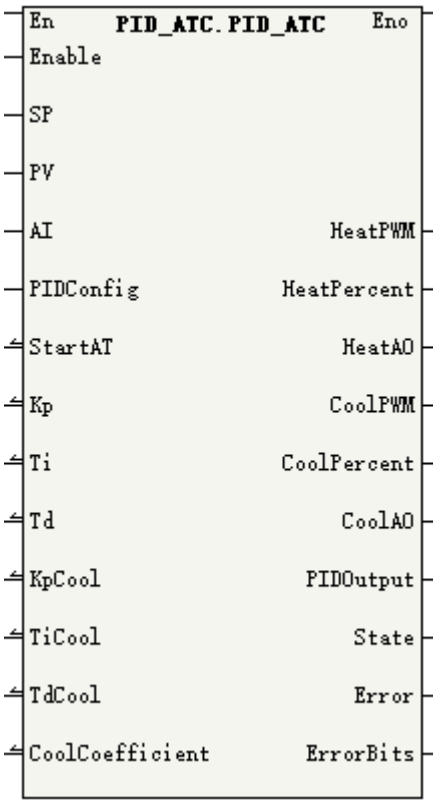
5.20.4 PID_ATC

Introduction

The PID_ATC instruction performs bipolar auto-tuning PID calculation to control the parameters of a close-loop control system.

PID_ATC -- Bipolar temperature PID controller

Graphic Block



The following table describes the instruction format.

Operand	Name	Description	Range	Data Type
16-bit Instruction	-			
32-bit Instruction	Bipolar temperature PID controller			

Operand	Name	Description	Range	Data Type
S1	SP	Setpoint, in °C	-	REAL
S2	PV	Feedback value, in °C	-	REAL
S3	AI	Analog input value	-	DINT
S4	PIDConfig	PID parameter configuration structure	-	_sPID_ATC_Config
S5	StartAT	PID parameter auto-tuning enable. After auto-tuning is completed, internal auto reset occurs. TRUE: Enable auto-tuning FALSE: Disable auto-tuning	-	BOOL
S6	Kp	Heating scaling	0.01 to 1000	REAL
S7	Ti	Heating integral time, in seconds A larger value indicates less integral effect. When the value is 0, there is no integral action.	0.00 to 10,000	REAL
S8	Td	Heating differential time, in seconds A larger value indicates larger differential effect. When the value is 0, there is no differential action.	0.00 to 10,000	REAL
S9	CoolKp	Cooling scaling	0.01 to 1000	REAL
S10	CoolTi	Cooling integral time, in seconds A larger value indicates less integral effect. When the value is 0, there is no integral action.	0.00 to 10,000	REAL
S11	CoolTd	Cooling differential time, in seconds A larger value indicates larger differential effect. When the value is 0, there is no differential action.	0.00 to 10,000	REAL
S12	CoolCoefficient	Cooling coefficient	0.00 to 100	REAL
D1	HeatPWM	Heating PWM output TRUE: Enable heating FALSE: Disable heating	-	BOOL
D2	HeatPercent	Heating output percentage (selected based on output types)	0.00 to 100	REAL

Operand	Name	Description	Range	Data Type
D3	HeatAO	Heating analog output value	-	DINT
D4	CoolPWM	Cooling PWM output TRUE: Enable cooling FALSE: Disable cooling	-	BOOL
D5	CoolPercent	Cooling output percentage (selected based on output types)	0.00 to 100	REAL
D6	CoolAO	Cooling analog output value	-	DINT
D7	PIDOutput	Output variable structure	-	_sPID_Output
D8	State	Operation state PID operation state Options: 0: Disable 1: Manual 2: Auto 3: Interlock 4: Auto-tuning 5: Error value output 6: Alternative output for the incorrect security value 7: Master tracking	-	DINT
D9	Error	Error flag	-	BOOL
D10	ErrorBits	Error code. For details, see 5.20.5 Error Codes. Bit 0: PV overlimit Bit 1: Parameter error	-	DINT

The following table lists soft elements in the instruction.

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	-	-	√	-	-	√
S5	√	√	√	-	-	-	-	-	-
S6	-	-	-	√	√	√	√	-	-
S7	-	-	-	√	√	√	√	-	-
S8	-	-	-	√	√	√	√	-	-
S9	-	-	-	√	√	√	√	-	-
S10	-	-	-	√	√	√	√	-	-
S11	-	-	-	√	√	√	√	-	-
S12	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	-	-	-	√	√	√	-	√	-
D3	-	-	-	√	√	√	√	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	√	-	√	-
D6	-	-	-	√	√	√	√	-	-
D7	-	-	-	-	-	√	-	-	√
D8	-	-	-	√	√	√	√	-	-
D9	√[1]	√	√	-	-	-	-	-	-
D10	-	-	-	√	√	√	√	-	-

NOTE

[1]: The X element is not supported.

Instruction Description

CAUTION

- Before running the PID, check whether the parameters are correct.
- It is recommended that the auto-tuning process starts at the room temperature.
- If the common temperature range that you use is 150°C to 200°C, tuning can be carried out at 175°C. It is important to select a common temperature for more accurate parameter tuning.
- In general, auto-tuning can be carried out first to obtain the tuned PID parameters. Then, you can adjust the PID parameters as needed based on the control effect.
- After the project calls and downloads the instruction to the PLC, the project periodically checks whether the input parameters are correct even if the instruction is not triggered.

Structure Description

The following table lists structures of the PID_ATC_Config type.

Variable Name	Type	Range	Default	Unit	Description
GeneralConfig	PID_GeneralConfig	-	-	-	PID general configuration
InputConfig	PID_InputConfig	-	-	-	PID input configuration
OutputConfigHeat	PID_OutputConfig	-	-	-	PID heating output configuration
OutputConfigCool	PID_OutputConfig	-	-	-	PID cooling output configuration
CoolConfig	PID_CoolConfig	-	-	-	PID cooling configuration
SuperConfig	PID_SuperConfig	-	-	-	PID advanced configuration

The following table describes variables in the PID_GeneralConfig structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
SampleTime	DINT	1	1000000	100	ms	Heating sampling period
CycleTime	DINT	1	1000000	1000	ms	Heating output period
PIDType	DINT	0	3	0	-	0: PID 1: PI

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
						2: PD 3: ID
ControlMode	DINT	0	1	0	-	0: Reverse (unipolar heating) 1: Forward (unipolar cooling) 2: Bipolar control based on cooling coefficient 3: Bipolar control based on independent PID parameter
ManualEnable	BOOL	FALSE	TRUE	FALSE	-	Manual mode switch
ManualValue	REAL	OutMin	OutMax	0.0	%	Manual mode output. The value ranges from -100 to +100 when cooling is enabled.
ATCoefficient	REAL	0.0	1.0	0.5	-	Auto-tuning coefficient
DelayStartUp	DINT	0	1000000	0	ms	Delayed start time. PID is started after delay of the set time for off-peak start in case of multi-channel temperature control on the device.
Delayshutdown	DINT	0	1000000	0	ms	Delayed shutdown time. PID is shut down after delay of the set time.
CAS_PV (input)	REAL	-	-	0	-	Process value of slave circuit When the slave circuit is detected as non-cascade automatic mode, the output of the master circuit tracks the process value of the slave circuit.
CAS_OVE (input)	INT	0	2	0	-	Overlimit status of slave circuit 0: Normal 1: Upper limit reached 2: Lower limit reached
CAS_State (input)	INT	0	1	0	-	Cascade use mode 0: Slave non-automatic mode 1: Slave automatic mode
IsMaster	BOOL	0	1	0	-	Master control 0: No 1: Yes
IsSlave	BOOL	0	1	0	-	Slave control

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
						0: No 1: Yes
PhysicalQuantity	INT	0	1	0	-	Physical object to control 0: General 1: Temperature
PhysicalUnit	DINT	-	-	0	-	Unit of the physical object (reserved)
ATNum	DINT	0	20	2		Number of auto-tuning waves (applicable to only unipolar) 0: Reserved; 1 to 20: Number of shockwaves
ATOut	REAL	rOutMin	rOutMax	100	%	Auto-tuning output percentage (applicable to only unipolar)

The following table describes variables in the PID_InputConfig structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
PVType	DINT	0	1	0	-	0: Use the current PV as the input 1: Use the peripheral analog as the input and convert it based on the analog range
AIMin	DINT	-	AIMax	0	-	Minimum value for analog conversion
AIMax	DINT	AIMin	-	20000	-	Maximum value for analog conversion
PVMin	REAL	-	PVMax	0.0	-	Minimum value of PV
PVMax	REAL	PVMin	-	100.0	-	Maximum value of PV
PVLowLimit	REAL	-	PVHighLimit	-100.0	-	Lower alarm limit for PV
PVHighLimit	REAL	PVLowLimit	-	1000.0	-	Higher alarm limit for PV
ErrorAction	BOOL	FALSE	TRUE	FALSE	-	Action for error alarm FALSE: Remain the value upon an error alarm TRUE: Output the alternative value upon an error alarm

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
ErrorOut	REAL	OutMin	OutMax	0.0	-	Output of an alternative percentage upon an error alarm
PVUnit	DINT	0	10	0	-	Reserved interface 0: ℃ 1: %
InlockSW	BOOL	0	1	0	-	Interlock switch
InlockVal	REAL	-100	100	0	-	Interlock security value
SPSel	BOOL	0	1	0	-	Setpoint source 0: External (SP configuration) 1: Internal (panel operation)
SPRatio	REAL	0	-	0.0	-	SP setpoint change
SPInt	REAL	-	-	0.0	-	Internal setpoint (reserved)
SPTrack	BOOL	0	1	1	-	Internal setpoint tracking switch (reserved) Default: TRUE

The following table describes variables in the PID_OutputConfig structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
OutType	DINT	0	2	0	-	Output type: 0: PWM output 1: Analog output 2: Percent output
OutMin	REAL	-100.0	OutMax	0.0	%	Minimum value of output percentage
OutMax	REAL	OutMin	100.0	100.0	%	Maximum value of output percentage
AOMin	DINT	-	AOMax	0	-	Minimum value of analog output
AOMax	DINT	AOMin	-	20,000	-	Maximum value of analog output
OutputOffset	REAL	-100.0	100.0	0.0	%	Output deviation compensation value

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
MinPT	DINT	0	10,000	0	ms	Minimum pulse time

The following table describes variables in the PID_CoolConfig structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
CoolPIDType	DINT	0	3	0	-	0: PID 1: PI 2: PD 3: ID
CoolSampleTime	DINT	1	1,000,000	100	ms	Cooling sampling period
CoolCycleTime	DINT	1	1,000,000	5000	ms	Cooling output period
CoolOffset	REAL	-1000.0	1000.0	0.0	-	Cooling deviation Cooling setpoint = SP + OffsetCool

The following table describes variables in the PID_SuperConfig structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
Fuzzy	BOOL	FALSE	TRUE	FALSE	-	FALSE: Disable fuzzy self-adaption TRUE: Enable fuzzy self-adaption
IntegralSuspend	BOOL	FALSE	TRUE	FALSE	-	FALSE: Enable integral TRUE: Pause integral
IntegralDivision	BOOL	FALSE	TRUE	FALSE	-	FALSE: Integral separated TRUE: Integral not separated
IntegralClear	BOOL	FALSE	TRUE	FALSE	-	FALSE: Normal integral TRUE: Clear integral
ECMax	REAL	-	-	0	-	Maximum temperature rise speed per second
KdGain	REAL	0.0	1.0	0.65	-	Incomplete differential coefficient A larger value indicates smoother differential effect. When the value is 1, there is no differential action.
IntegralMax	REAL	-100.0	100.0	100.0	%	Maximum integral value

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
IntegralCut	REAL	0.0	1.0	0.75	-	Integral cutting coefficient for setpoint change
IntegralDivLimit	REAL	0.0	100.0	50	-	Integral separation limit
DeadBand	REAL	0.0	100.0	0	-	Deviation dead zone limit

The following table describes variables in the PID_Output structure.

Variable Name	Type	Minimum	Maximum	Default	Unit	Description
PVOut	REAL	-	-	-	-	PV value converted from analog
ECMaxOut	REAL	-	-	-	-	Maximum temperature rise speed per second
ATState	DINT	-	-	-	-	Auto-tuning status 0: No auto-tuning 1: During auto-tuning 2: Auto-tuning completed
ErrorID	DINT	-	-	-	-	Error code
Debug	REAL*9	-	-	-	-	Commissioning parameter
CAS_PV	REAL	-	-	-	-	Process value of slave circuit When the slave circuit is detected as non-cascade automatic mode, the output of the master circuit tracks the process value of the slave circuit.
CAS_OVE	INT	-	-	-	-	Overlimit status of slave circuit 0: Normal 1: Upper limit reached 2: Lower limit reached
CAS_State	DINT	-	-	-	-	Cascade use mode 0: Slave non-automatic mode 1: Slave automatic mode

NOTE

- Enable the auto-tuning function at room temperature. When auto-tuning is completed, StartAT is automatically set to FALSE, and you can obtain the tuned values of Kp, Ti, and Td. Let the program continue to run and check whether the controlled object can reach steady state.
- If the controlled object reaches the steady state, record the values of Kp, Ti, and Td, and write these values into the program as defaults. You can also manually adjust the values of Kp, Ti, and Td to meet your needs.
- The larger the auto-tuning coefficient specified by ATCoefficient, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature can be reached, and the temperature curve will be smooth, without overshoot or with small overshoot. Unless otherwise specified, it is recommended to use the default 0.5. If you want to reach the set temperature faster, increase the auto-tuning coefficient and perform auto-tuning again. If you need a smooth curve and small overshoot, decrease the auto-tuning coefficient and perform auto-tuning again.

Parameter Description

- Enable
  The following table describes the Enable parameter.

PID Calculation	Value	Operation
Start	TRUE	Obtain the measured value of PV, perform PID calculation, and output the result in turn.
End	FALSE	Output percentage values of HeatPercent and CoolPercent and clear PWM outputs HeatPWM and CoolPWM and analog outputs HeatAO and CoolAO.

- StartAT

This instruction provides the auto-tuning function to automatically calculate the optimal values for PID constants Kp, Ti, and Td.

The auto-tuning execution conditions upon heating are as follows:

  - The current value is less than the setpoint. In general, the setpoint is the common operating temperature.
  - StartAT
    The following table describes the StartAT parameter.

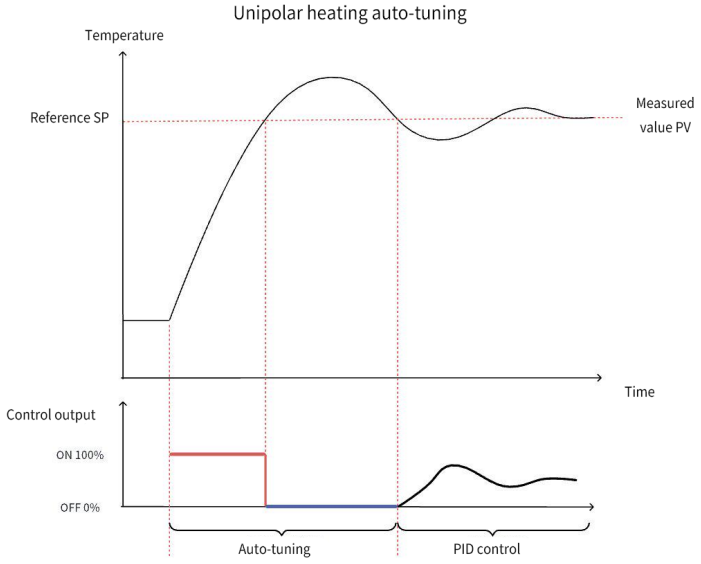
Value of StartAT	Value of Enable	PID Calculation	Auto-tuning Started	Auto-tuning Completed	Operation
TRUE	Change from FALSE to TRUE	-	The value of ATState.1 changes to 1.	The value of ATState.1 changes to 0.	Auto-tuning is performed when instruction

Value of StartAT	Value of Enable	PID Calculation	Auto-tuning Started	Auto-tuning Completed	Operation
					execution starts.
Change from FALSE to TRUE	TRUE	Yes	-	The value of ATState.2 changes to 1.	Auto-tuning is performed during instruction execution.
Change from TRUE to FALSE	-	-	-	ATState.1 = 0 ATState.2 = 0	-

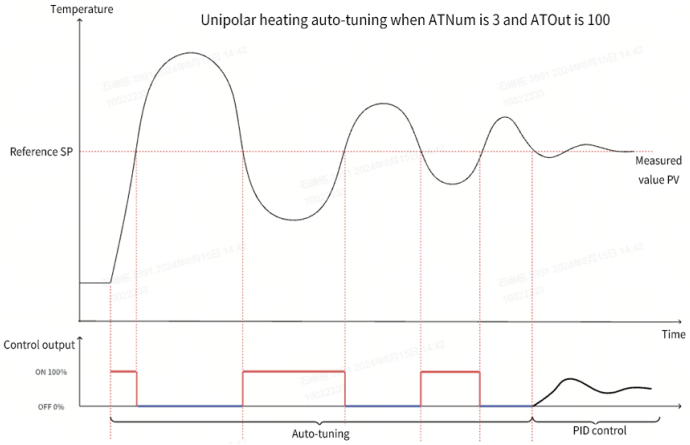
NOTE

After auto-tuning is completed, StartAT is automatically set to FALSE.

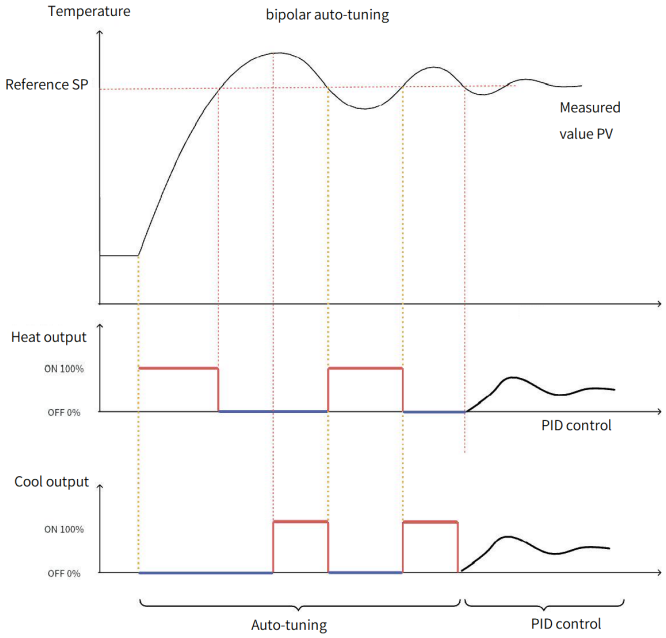
- ATOut specifies the heating or cooling output percentage in the auto-tuning stage. This parameter is available only in unipolar mode.
- ATNum specifies the number of shockwaves based on which auto-tuning is performed.
- The following figure shows the unipolar tuning process when ATOut is set to 100 and ATNum is set to 1.



- The following figure shows the unipolar tuning process on three shockwaves when ATOut is set to 100 and ATNum is set to 3.



- During bipolar auto-tuning, two waveforms are generated near the setpoint. After auto-tuning is completed, StartAT is set to FALSE.



NOTE

During the auto-tuning process, parameter configuration cannot be modified. Otherwise, the parameters may be improper.

The following table describes parameters related to the tuning process.

Parameter	Value	Not Satisfied	ErrorID	Description
Scaling coefficient Kp	> 0.01	The value of Error is TRUE.	Error code reported	This parameter can be set by yourself or automatically obtained through auto-tuning. A larger value indicates a stronger scaling action.

Parameter	Value	Not Satisfied	ErrorID	Description
Integral time Ti	≥ 0 If the value is 0, there is no integral action.	The value of Error is TRUE.	Error code reported	This parameter can be set by yourself or automatically obtained through auto-tuning. A larger value indicates a weaker integral action.
Differential time Td	≥ 0 If the value is 0, there is no differential action.	The value of Error is TRUE.	Error code reported	This parameter can be set by yourself or automatically obtained through auto-tuning. A larger value indicates a stronger differential action.
Cooling scaling coefficient KpCool	> 0	The value of Error is TRUE.	Error code reported	This parameter can be set by yourself or automatically obtained through auto-tuning. A larger value indicates a stronger scaling action.
Cooling integral time TiCool	≥ 0 If the value is 0, there is no integral action.	The value of Error is TRUE.	Error code reported	This parameter can be set by yourself or automatically obtained through auto-tuning.
Cooling differential time TdCool	≥ 0 If the value is 0, there is no differential action.	The value of Error is TRUE.	Error code reported	This parameter can be set by yourself or automatically obtained through auto-tuning.
Cooling coefficient CoolCoefficient	≥ 0 If the value is 0, there is no cooling action.	The value of Error is TRUE.	Error code reported	This parameter can be set by yourself or automatically obtained through auto-tuning. When GeneralConfig.Mode is set to 2, the cooling output is calculated based on the cooling efficient CoolCoefficient. A smaller cooling efficient indicates weaker cooling output. A larger cooling efficient indicates stronger cooling output. Cooling output = PID cooling output * Cooling coefficien

- PID_Config includes the following structures:
  GeneralConfig, InputConfig, OutputConfigHeat, OutputConfigCool, CoolConfig, and SuperConfig
  - GeneralConfig
    The following table describes parameters in the PID_GeneralConfig structure.

PID_GeneralConfig Parameter Type	Description	Remarks
GeneralConfig.SampleTime Sampling period	The current PV value is obtained once a sampling period for one PID calculation. In general, the sampling period of the temperature collection module is used. Unless otherwise specified, the default value 100 ms can be used.	-
GeneralConfig.CycleTime Output period	Output period specified by HeatPWM. If the HeatPercent output value of the PID is 80% and the value of GeneralConfig.CycleTime is 10000 ms, the period for HeatPWM is 10000 ms, with 8s on and 2s off.	-
GeneralConfig.PIDType Type	Type of the current PID 0: PID calculation 1: PI calculation 2: PD calculation 3: ID calculation	Perform corresponding calculation control based on the selected PID type. Default: 0: PID calculation
GeneralConfig.ControlMode Control mode	0: Reverse (unipolar heating) 1: Forward (unipolar cooling) 2: Bipolar control based on cooling coefficient 3: Bipolar control based on independent PID parameter	-
GeneralConfig.ManualEnable Manual mode switch	0: Automatic mode 1: Manual mode Unipolar: between the minimum output and the maximum output Bipolar: between the maximum cooling output and the maximum heating output	When the manual mode is enabled, output the value of GeneralConfig.ManualValue. In bipolar mode, the positive value of GeneralConfig.ManualValue corresponds to the heating output value and the negative value of GeneralConfig.ManualValue corresponds to the cooling output value. The manual mode can be changed to the automatic mode through unperturbed switchover.

PID_GeneralConfig Parameter Type	Description	Remarks
GeneralConfig.ATCoefficient Auto-tuning coefficient	For the auto-tuning coefficient, the larger the auto-tuned Kp value, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature after auto-tuning can be reached, and the temperature curve will be smooth, without overshoot or with small overshoot.	Use the default value 0.5 for initial parameter tuning. If you want to reach the set temperature faster, increase the auto-tuning coefficient GeneralConfig.ATCoefficient based on the control effect and perform auto-tuning again. If you need a smooth curve and small overshoot, decrease the auto-tuning coefficient GeneralConfig.ATCoefficient and perform auto-tuning again.
GeneralConfig.DelayStartUp Delayed start time	Start the PID after the set delay time.	Different delayed start time can be set for off-peak start in case of multi-channel temperature control on the device. This prevents a large current fluctuation during multi-channel start.
GeneralConfig.Delayshutdown Delayed shutdown time	Shut down the PID after the set delay time.	-
GeneralConfig.PhysicalQuantity Physical object to control	Physical object to control. The options are as follows: 0: General. This option is mainly used for pressure and flow control. When this option is selected, you can specify the advanced parameters IntegralDivision and IntegralDivLimit for integral separation. 1: Temperature. Various optimization methods are available for temperature control.	-
GeneralConfig.ATNum Number of auto-tuning waves	After you specify this parameter, auto-tuning is performed on the specified shockwaves.	This parameter is supported only in unipolar mode.
GeneralConfig.ATOut		This parameter is supported only in unipolar mode.

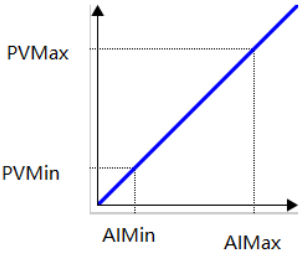
PID_GeneralConfig Parameter Type	Description	Remarks
Auto-tuning output percentage	This parameter specifies the heating or cooling output percentage in the auto-tuning stage.	

The following table describes minimum and maximum values in different control modes.

Control Mode	Minimum	Maximum
0	OutputConfigHeat.OutMin	OutputConfigHeat.OutMax
1	OutputConfigHeat.OutMin	OutputConfigHeat.OutMax
2	OutputConfigCool.OutMax	OutputConfigHeat.OutMax
3	OutputConfigCool.OutMax	OutputConfigHeat.OutMax

◦ InputConfig

The following table describes parameters in the PID_InputConfig structure.

<table>
<tr><th>PID_InputConfig Parameter Type</th><th>Description</th><th>Remarks</th></tr>
<tr><td>InputConfig.PVType
Sampling PV type</td><td>0: Use the current PV as the input
1: Use the peripheral analog as the input</td><td>Perform conversion based on the analog range specified by InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax.</td></tr>
<tr><td>InputConfig. AIMin
Minimum value for analog conversion</td><td rowspan="2">Use the peripheral analog as the input and perform conversion based on the analog range specified by InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax.</td><td rowspan="2">They are used to convert an analog to a PV value.
 
</td></tr>
<tr><td>InputConfig.AIMax
Maximum value for analog conversion</td></tr>
<tr><td>InputConfig.PVMin</td><td>Minimum PV value corresponding to InputConfig.AIMin</td><td rowspan="2">Use the peripheral analog as the input and perform conversion based on the analog range specified by InputConfig.AIMin and InputConfig.AIMax as well as the corresponding PVMin and PVMax.</td></tr>
<tr><td>InputConfig.PVMax</td><td>Maximum PV value corresponding to InputConfig.AIMax</td></tr>
</table>

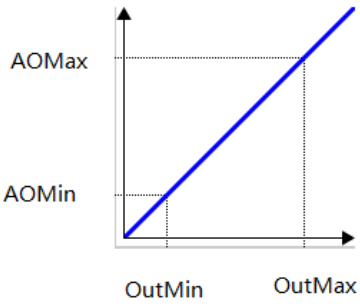
PID_InputConfig Parameter Type	Description	Remarks
InputConfig.InputMin Upper and lower input limit alarms	When the PV value or the PV value converted from the analog is lower than the value of InputConfig.InputMin, an alarm is generated for exceeding the lower limit.	Alarms are generated based on InputConfig.ErrorAction and ErrorOut.
InputConfig.InputMax Upper and lower input limit alarms	When the PV value or the PV value converted from the analog is higher than the value of InputConfig.InputMax, an alarm is generated for exceeding the upper limit.	
InputConfig.ErrorAction Error alarm	FALSE: Remain the output value upon an error alarm TRUE: Output the alternative value InputConfig.ErrorOut upon an error alarm	Unipolar: between the minimum output and the maximum output Bipolar: between the maximum cooling output and the maximum heating output
InputConfig.ErrorOut Error alarm		
InputConfig.SPSel Setpoint source	Select the SP value used for PID calculation. 0: External (SP configuration) 1: Internal (SPInt set through panel operation)	-
InputConfig.SPRatio SP setpoint change	If SPRatio is greater than 0, the final value turns closer to the setpoint based on the change of SPRatio. This parameter is calculated once every task cycle.	-

The following table describes minimum and maximum values in different control modes.

Control Mode	Minimum	Maximum
0	OutputConfigHeat.OutMin	OutputConfigHeat.OutMax
1	OutputConfigHeat.OutMin	OutputConfigHeat.OutMax
2	OutputConfigCool.OutMax	OutputConfigHeat.OutMax
3	OutputConfigCool.OutMax	OutputConfigHeat.OutMax

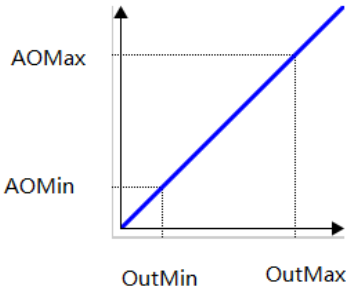
◦ OutputConfigHeat

The following table describes parameters in the PID_OutputConfig structure.

PID_OutputConfig Parameter Type	Description	Remarks
OutputConfigHeat.OutType Output type	0: PWM output. HeatPercent indicates the percentage for PWM output. 1: Analog output. HeatPercent indicates the percentage for analog output.	-
OutputConfigHeat.OutMin Minimum value of output percentage	Convert the output percentage of PID calculation based on the analog range specified by OutputConfigHeat.OutMin and OutputConfigHeat.OutMax as well as the corresponding OutputConfigHeat.AOMin and OutputConfigHeat.AOMax.	They are used to limit the output of PID calculation and convert the analog output. 
OutputConfigHeat.OutMax Maximum value of output percentage		
OutputConfigHeat.AOMin Minimum value of analog output	Convert the output percentage of PID calculation based on the analog range specified by OutputConfigHeat.OutMin and OutputConfigHeat.OutMax as well as the corresponding OutputConfigHeat.AOMin and OutputConfigHeat.AOMax.	They are used to convert the analog output.
OutputConfigHeat.AOMax Maximum value of analog output		
OutputConfigHeat.OutputOffset Output deviation compensation value	OUT = PID calculation value + OutputOffset The final output is equal to the PID calculation value plus the value of OutputOffset. They are used feedforward connection and compensation.	-
OutputConfigHeat.MinPT Minimum pulse time	This parameter specifies the minimum pulse time during which PWM is enabled or disabled, in milliseconds. The pulse is not enabled when the enable time is less than that specified by MinPT. The pulse is not disabled when the disable time is less than that specified by MinPT.	-

◦ OutputConfigCool

The following table describes parameters in the PID_CoolConfig structure.

PID_CoolConfig Parameter Type	Description	Remarks
OutputConfigCool.OutType Output type	0: PWM output. CoolPercent indicates the percentage for PWM output. 1: Analog output. CoolPercent indicates the percentage for analog output.	-
OutputConfigCool.OutMin Minimum value of output percentage	Convert the output percentage of PID calculation based on the analog range specified by OutputConfigCool.OutMin and OutputConfigCool.OutMax as well as the corresponding OutputConfigCool.AOMin and OutputConfigCool.AOMax.	They are used to limit the output of PID calculation and convert the analog output. 
OutputConfigCool.OutMax Maximum value of output percentage	-	-
OutputConfigCool.AOMin Minimum value of analog output	Convert the output percentage of PID calculation based on the analog range specified by OutputConfigCool.OutMin and OutputConfigCool.OutMax as well as the corresponding OutputConfigCool.AOMin and OutputConfigCool.AOMax.	They are used to convert the analog output.
OutputConfigCool.AOMax Maximum value of analog output		-
OutputConfigCool.OutputOffset Output deviation compensation value	OUT = PID calculation value + OutputOffset: The final output is equal to the PID calculation value plus the value of OutputOffset.	-

PID_CoolConfig Parameter Type	Description	Remarks
OutputConfigCool.MinPT Minimum pulse time	This parameter specifies the minimum pulse time during which PWM is enabled or disabled, in milliseconds. The pulse is not enabled when the enable time is less than that specified by MinPT. The pulse is not disabled when the disable time is less than that specified by MinPT.	-

◦ CoolConfig

The following table describes parameters in the PID_CoolConfig structure.

PID_CoolConfig Parameter Type	Description	Remarks
CoolConfig.CoolPIDType	Type of the current PID 0: PID calculation 1: PI calculation 2: PD calculation 3: ID calculation	Perform corresponding calculation control based on the selected PID type. Default: 0: PID calculation
CoolConfig.CoolSampleTime Sampling period	The current PV value is obtained once a sampling period for one PID calculation. In general, the sampling period of the temperature collection module is used. Unless otherwise specified, the default value 100 ms can be used.	-
CoolConfig.CoolCycleTime Output period	This parameter specifies the output period. If the CoolPercent output value of the PID is 80% and the value of CoolConfig.CoolCycleTime is 10000 ms, the period for CoolPWM is 10000 ms, with 8s on and 2s off.	-
CoolConfig.CoolOffset Cooling deviation	Cooling setpoint = Heating setpoint SP + CoolOffset	-

◦ SuperConfig

The following table describes parameters in the PID_SuperConfig structure.

PID_SuperConfig Parameter Type	Description	Remarks
SuperConfig.Fuzzy Fuzzy self-adaption	0: Disable fuzzy self-adaption 1: Enable fuzzy self-adaption	PID parameters can be dynamically adjusted based on

PID_SuperConfig Parameter Type	Description	Remarks
		fuzzy self-adaption (available for unipolar heating).
SuperConfig.ECMax Maximum temperature rise speed per second	This parameter is used in fuzzy self-adaptive mode. The maximum temperature rise speed of the device can be specified by yourself or obtained through auto-tuning (PIDOutput.ECMaxOut).	-
SuperConfig.KdGain Incomplete differential coefficient	The value ranges from 0.0 to 1.0. A larger value indicates smoother differential effect. When the value is 1, there is no differential action.	-
SuperConfig.IntegralSuspend Integral pause	0: Enable integral (default) 1: Pause integral	-
SuperConfig.IntegralDivision Integral separation	0: Integral separated (default) 1: Integral not separated	-
SuperConfig.IntegralClear Integral clear	0: Normal integral (default) 1: Integral clear	-
SuperConfig.IntegralMax Maximum integral value	The maximum integral value is 75% by default.	-
SuperConfig.IntegralCut Setpoint	Integral cutting coefficient for setpoint change The integral cutting coefficient is 0.75 by default.	-
SuperConfig.Reserve Reserved	Reserved	-
SuperConfig.IntegralDivLimit Integral separation limit	When the deviation is greater than the value of IntegralDivLimit, no integration is performed.	-
SuperConfig.DeadBand Deviation dead zone limit	When the deviation of |SP - PV| is less than the value of DeadBand, the deviation is considered as 0 (no PID calculation).	-

- Output structure

The following table describes parameters in the output structure.

PIDOutput.PVOut	Actually calculated PV value
PIDOutput.ECMaxOut	Maximum temperature rise speed per second, auto-tuned
PIDOutput.ATState	Auto-tuning status 0: No auto-tuning 1: During auto-tuning 2: Auto-tuning completed
PIDOutput.ErrorID	Error code For details, see 5.20.5 Error Codes.

- Cascade function

Cascade control involves two or more cascaded PID controllers. The control variable of the master controller is interconnected with the setpoint of the slave controller, creating a nested control circuit. Cascade control in a slave circuit can compensate for interference in the internal circuit faster than in the slower main circuit.

Note: Cascade control is possible only when there are other measurable variables in the process, and the internal control circuit is much faster than the external circuit.

Unperturbed switchover: When the slave circuit is not in cascade mode (automatic mode by external setpoint), the main circuit switches to the tracking mode. The output value of the main circuit tracks the process value of the slave circuit, allowing for a smooth transition back to the cascade mode.

Priority (Error response > Interlock > Manual > Master tracking mode triggered by slave > Automatic)

- Working modes

Working Mode	Action
Automatic	The output value of the PID module is calculated based on the following rule: scaling, integral, and differential in turn. The calculation result of the PID module is restricted by the output amplitude.
Manual	The output is manually controlled (specified by the operator or in online logic). The output of this mode is restricted by the output amplitude.
Interlock	In interlock mode, PID stops automatic calculation, and the output value of the PID module is a pre-set value of InlockVal, which is generally referred to as the interlock point. The interlock function is also used for tracking. When a certain condition is met, for example, InlockSW is set to TRUE, the PID module automatically switches to the interlock mode. When the interlock condition is not met, the PID module switches back to the previous working mode.

(1) Interlock has the highest priority, followed by manual, and finally automatic.

(2) Automatic mode features much control with fixed setpoint, which can be specified on the configuration interface.

(3) You can choose to use the external setpoint or internal setpoint based on the value of PIDConfig.InputConfig.SPSel.

If SPSel is set to FALSE, the external setpoint in the logical configuration is used.

If SPSel is set to TRUE, the internal setpoint is used.

- Unperturbed switchover

Unperturbed Switchover Method	Description
Interlock to manual	In interlock mode, the interlock value is directly output. In this case, the manual value tracks the output value to ensure unperturbed switchover.
Interlock to automatic	In interlock mode, the interlock value is directly output. After the interlock mode is switched to the automatic mode, if the external setpoint is used, the logic operation value is directly used for the PID calculation output, which may cause perturbation.
Manual to interlock	In manual mode, the manual value is output. After the manual mode is switched to the interlock mode, the output is the interlock value, and perturbation occurs.
Manual to automatic	In manual mode, the output is the value specified by the operator. If the external setpoint is used, the logic operation value is directly used for the PID calculation output. In this case, the switchover is unperturbed (with the incremental part perturbed).
Automatic to interlock	In automatic mode, the output is the normal operation value. After the automatic mode is switched to the interlock mode, the output is the interlock value, and perturbation occurs.
Automatic to manual	In automatic mode, the output is the normal operation value. In this case, the manual value tracks the output value to ensure unperturbed switchover.

- Setpoint change limit function

  In automatic mode, the setpoint (external or internal) change is checked. If the change exceeds a preset limit (specified by PIDConfig.InputConfig.SPRatio), the actual setpoint used in the calculation is restricted to the change value and the output gradually approaches the setpoint based on the magnitude of the change.

$$\Delta SP > SPRatio\left(Setpoint\ change\right):\ SP = LastSP + SPRatio$$

$$\Delta SP < SPRatio\left(Setpoint\ change\right):\ SP = LastSP - SPRatio$$

Integral Function

- Integral Separation

Integral saturation is mainly caused by the integral item. To overcome integral saturation, it is crucial to limit the integral action to prevent excessive integral accumulation. To eliminate integral saturation, the integral separation function can be employed to provide the anti-integral-saturation capability to the function block. The following table describes the integral separation configuration for non-temperature objects.

Integral Separation Enable	Integral Separation Limit
PIDConfig.SuperConfig.IntegralDivision	PIDConfig. SuperConfig.IntegralDivLimit

TRUE	When |Deviation| is greater than or equal to the integral separation limit, no integral component operation is performed. In this case, PD control is used.
	When |Deviation| is less than the integral separation limit, the full integral operation is performed. In this case, PID control is used.
FALSE	The full integral operation is always performed.

- Integral pause

If PIDConfig.GeneralConfig.IntegralSuspend is set to TRUE, integral pause is triggered. In this case, the integral action remains unchanged and no more integral accumulation occurs.

- Integral clear

If PIDConfig.GeneralConfig.IntegralClear is set to TRUE, integral clear is triggered. In this case, the integral action is cleared to 0.

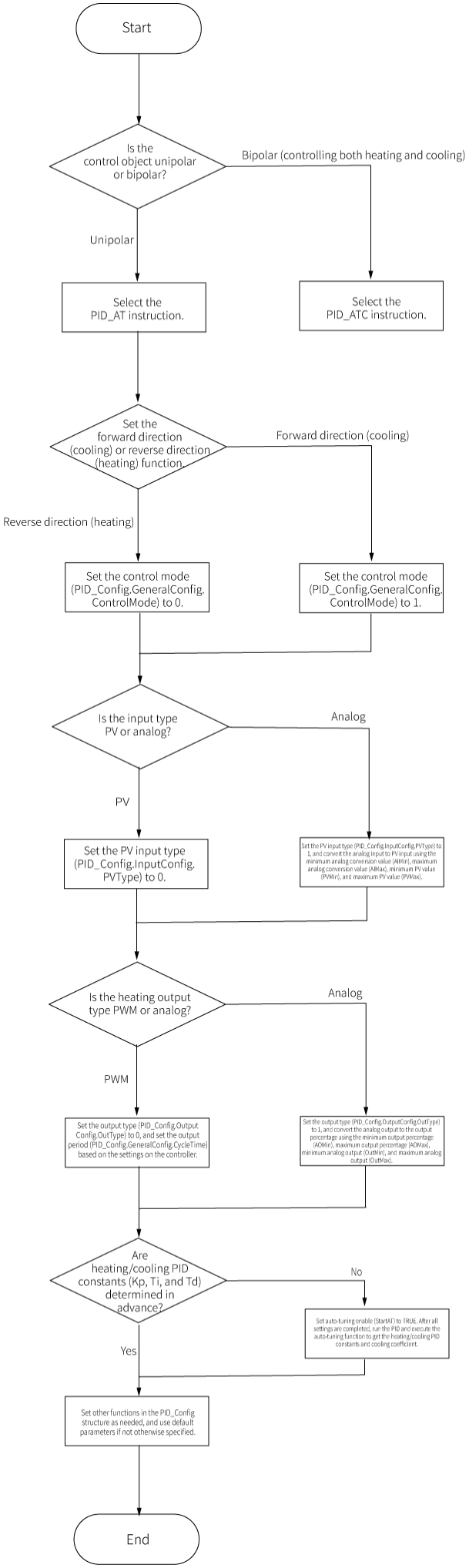
Integral cutting coefficient

When the setpoint changes, you can reduce the initial integral action by specifying PIDConfig.SuperConfig.IntegralCut. The value range is from 0 to 1. The default value is 0.75.

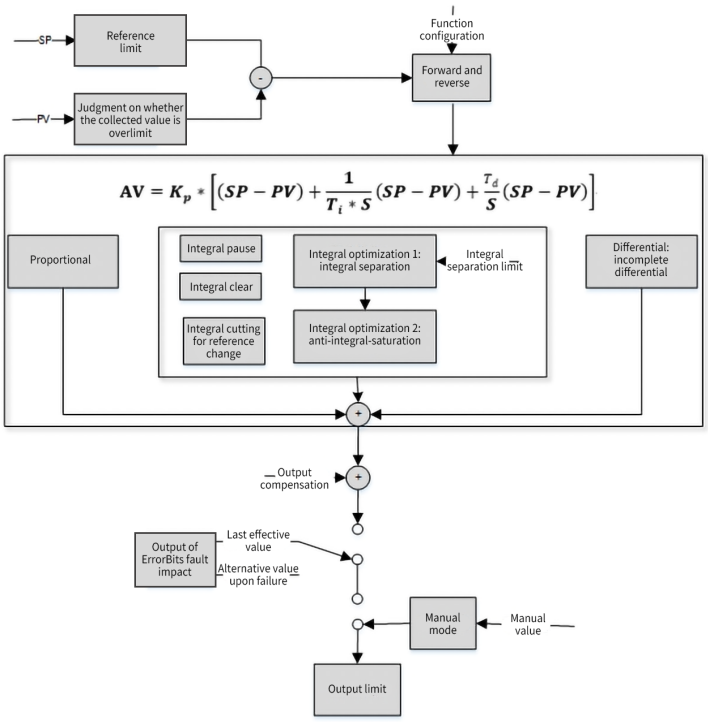
- Dead zone control function

  You can specify PIDConfig.SuperConfig.DeadBand to determine the dead zone width. Within the dead zone range, the adjustment is stable and the PID output does not change. If the value is beyond the dead zone range, PID calculation is normal.

Initial setting process


Start
Is the control object unipolar or bipolar?
Bipolar (controlling both heating and cooling)
Unipolar
Select the PID_AT instruction.
Select the PID_ATC instruction.
Set the forward direction (cooling) or reverse direction (heating) function.
Forward direction (cooling)
Reverse direction (heating)
Set the control mode (PID_Config.GeneralConfig.ControlMode) to 0.
Set the control mode (PID_Config.GeneralConfig.ControlMode) to 1.
Is the input type PV or analog?
Analog
PV
Set the PV input type (PID_Config.InputConfig.PVType) to 0.
Set the PV input type (PID_Config.InputConfig.PVType) to 1, and convert the analog input to PV input using the minimum analog conversion value (AMin), maximum analog conversion value (AMax), minimum PV value (PVMin), and maximum PV value (PVMax).
Is the heating output type PWM or analog?
Analog
PWM
Set the output type (PID_Config.OutputConfig.OutType) to 0, and set the output period (PID_Config.GeneralConfig.CycleTime) based on the settings on the controller.
Set the output type (PID_Config.OutputConfig.OutType) to 1, and convert the analog output to the output percentage using the minimum output percentage (AOMin), maximum output percentage (AOMax), minimum analog output (OutMin), and maximum analog output (OutMax).
Are heating/cooling PID constants (Kp, Ti, and Td) determined in advance?
No
Set auto-tuning enable (StartAT) to TRUE. After all settings are completed, run the PID and execute the auto-tuning function to get the heating/cooling PID constants and cooling coefficient.
Yes
Set other functions in the PID_Config structure as needed, and use default parameters if not otherwise specified.
End

Operating logic diagram



PID_ATC is a dedicated temperature controller designed for heating, cooling, and heating/cooling applications. It provides two independent outputs, one for heating and one for cooling. It supports cascade configuration applications and parameter auto-tuning.

- Calculation formula

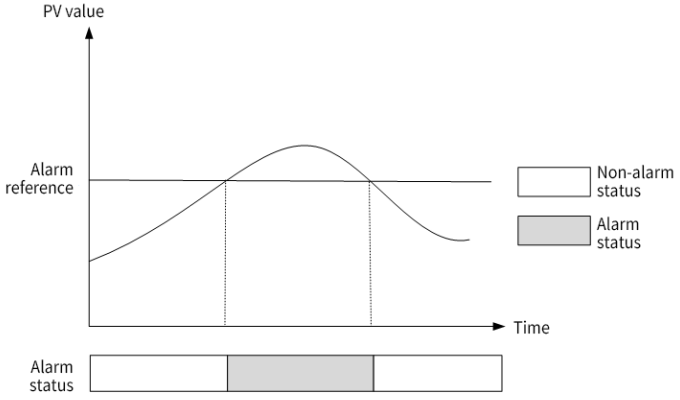
$$\mathbf{AV} = \boldsymbol{K_p} * \left[(\boldsymbol{SP} - \boldsymbol{PV}) + \frac{\mathbf{1}}{\boldsymbol{T_i} * \boldsymbol{S}}(\boldsymbol{SP} - \boldsymbol{PV}) + \frac{\boldsymbol{T_d}}{\boldsymbol{S}}(\boldsymbol{SP} - \boldsymbol{PV})\right]$$

- Discretization formula

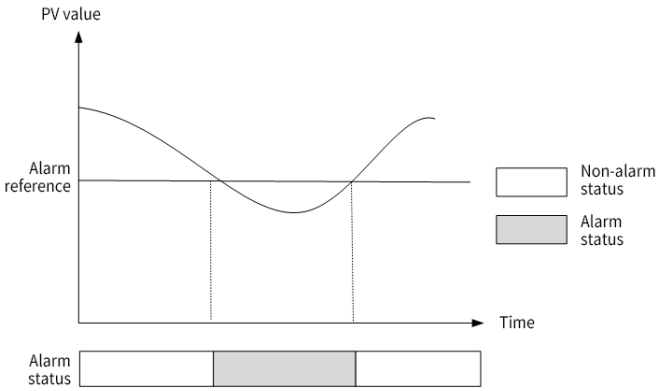
$$\mathbf{AV(k)} = \boldsymbol{K_p} * \mathbf{E}(\boldsymbol{k}) + \boldsymbol{K_p} * \frac{\boldsymbol{cycle}}{\boldsymbol{T_i}} \sum \mathbf{E}(\boldsymbol{k}) + \frac{\boldsymbol{K_p} * \boldsymbol{T_D}}{\boldsymbol{cycle}}(\mathbf{E}(\boldsymbol{k}) - \mathbf{E}(\boldsymbol{k} - \mathbf{1}))$$

Upper and lower limit alarms

- Upper limit alarm of the input PV value
  When the measured value of PV (or the PV value converted from the analog) is greater than the upper limit, an alarm is reported.



- Lower limit alarm of the input PV value
  When the measured value of PV (or the PV value converted from the analog) is less than the lower limit, an alarm is reported.



5.20.5 Error Codes

Category	Error No.	Error Codes	Description
No error	0	NO_ERROR :=0	No error
Input interface parameter error	1	KP_ERROR	Kp configuration error
	2	TI_ERROR	Ti configuration error
	3	TD_ERROR	Td configuration error
	4	KPCOOL_ERROR	Cooling Kp configuration error (available for PID_ATC)
	5	TICOOL_ERROR	Cooling Ti configuration error (available for PID_ATC)
	6	TDCOOL_ERROR	Cooling Td configuration error (available for PID_ATC)
	7	COOL_COEFFICIENT_ERROR	Cooling coefficient configuration error (available for PID_ATC)
General configuration	8	SAMPLE_TIME_ERROR	Sampling period - General configuration error
	9	CYCLE_TIME_ERROR	Output period - General configuration error
	10	PID_TYPE_ERROR	PID type - General configuration error
	11	MODE_ERROR	PID control mode - General configuration error
	12	MANUAL_VALUE_ERROR	Manual output value - General configuration error
	13	ATCOEFFICIENT_ERROR	Auto-tuning coefficient - General configuration error

Category	Error No.	Error Codes	Description
	14	DELAY_STARTUP_ERROR	Delayed startup - General configuration error
	15	DELAY_SHUTDOWN_ERROR	Delayed shutdown - General configuration error
Input configuration	16	PVTYPE_ERROR	PV type - Input configuration error
	17	AI_MIN_ERROR	Minimum value for analog conversion - Input configuration error
	18	AI_MAX_ERROR	Maximum value for analog conversion - Input configuration error
	19	PV_MIN_ERROR	Minimum value of PV - Input configuration error
	20	PV_MAX_ERROR	Maximum value of PV - Input configuration error
	21	PV_LOW_LIMIT_ERROR	Input lower limit - Input configuration error
	22	PV_HIGH_LIMIT_ERROR	Input upper limit - Input configuration error
	23	ERROR_OUT_ERROR	Output of an alternative value upon an error alarm - Input configuration error
	24	PV_UNIT_ERROR	PV unit configuration error
Output configuration	25	OUT_TYPE_ERROR	Output type - Output configuration error
	26	OUT_MIN_ERROR	Minimum value of output percentage - Output configuration error
	27	OUT_MAX_ERROR	Maximum value of output percentage - Output configuration error
	28	AO_MIN_ERROR	Minimum value of analog output - Output configuration error
	29	AO_MAX_ERROR	Maximum value of analog output - Output configuration error
	30	OUTPUT_OFFSET_ERROR	Output compensation - Output configuration error
Cooling output configuration (available for PID_ATC)	31	COOL_OUT_TYPE_ERROR	Cooling output type - Cooling output configuration error
	32	COOL_OUT_MIN_ERROR	Minimum value of cooling output percentage - Cooling output configuration error

Category	Error No.	Error Codes	Description
	33	COOL_OUT_MAX_ERROR	Maximum value of cooling output percentage - Cooling output configuration error
	34	COOL_AO_MIN_ERROR	Minimum value of cooling analog output - Cooling output configuration error
	35	COOL_AO_MAX_ERROR	Maximum value of cooling analog output - Cooling output configuration error
	36	COOL_OUTPUT_OFFSET_ERROR	Cooling output compensation - Cooling output configuration error
Advanced configuration	37	ECMAX_ERROR	Maximum temperature rise speed - Advanced configuration error
	38	KD_GAIN_ERROR	Incomplete differential coefficient - Advanced configuration error
	39	INTEGRAL_MAX_ERROR	Maximum integral value - Advanced configuration error
	40	INTEGRAL_CUT_ERROR	Integral cutting coefficient for setpoint change - Advanced configuration error
Cooling configuration (available for PID_ATC)	41	COOL_PID_TYPE_ERROR	Cooling PID type - Cooling configuration error
	42	SAMPLE_COOL_TIME_ERROR	Cooling sampling period - Cooling configuration error
	43	CYCLE_COOL_TIME_ERROR	Cooling output period - Cooling configuration error
	44	COOL_OFFSET_ERROR	Cooling setting deviation - Cooling configuration error
PV limit alarm	45	PV_LOW_LIMIT_ALARM	Lower limit alarm of the PV value. Check the PIDConfig.InputConfig.PVLowLimit and PIDConfig.InputConfig.PVHighLimit parameters to ensure that the PV value falls within the range.
	46	PV_HIGH_ LIMIT_ALARM	Upper limit alarm of the PV value. Check the PIDConfig.InputConfig.PVLowLimit and PIDConfig.InputConfig.PVHighLimit parameters to ensure that the PV value falls within the range.
Input interface configuration error	47	PV_AND_AI_NOINPUT	No input configuration for PV and AI

Category	Error No.	Error Codes	Description
Input configuration error	48	SP_RAT_ERROR	Setpoint change configuration error - Input configuration error
Advanced configuration error	49	INTEGRAL_DIV_ERROR	Integral separation limit configuration error - Advanced configuration error
	50	DEADBAND_ERROR	Control dead zone configuration error - Advanced configuration error
Cooling output configuration error (available for PID_ATC)	51	COOL_MINPT_ERROR	Minimum cooling on/off time configuration error - Cooling output configuration error
Output configuration error	52	OUT_MINPT_ERROR	Minimum on/off time configuration error - Output configuration error
Input configuration error	53	INLOCKVAL_ERR	Interlock security value configuration error
General configuration error	54	CASSLAVESTATE_PV_ERR	PV overlimit in slave data
	55	CASSLAVESTATE_OVE_ERR	OVE configuration error in slave data
	56	CASSLAVESTATE_CASSTATE_ ERR	Slave status configuration error in slave data
Auto-tuning wave quantity configuration error	57	AT_NUM_ERROR	Auto-tuning wave quantity configuration error
Auto-tuning output percentage configuration error	58	AT_OUT_ERROR	Auto-tuning output percentage configuration error

For all configuration errors, check the upper and lower limits of parameters.

Error Handling

PIDConfig.InputConfig.ErrorAction specifies the action upon an error alarm.

1. When PIDConfig.InputConfig.ErrorAction is set to FALSE, the output remains unchanged upon an error alarm. When the error is recovered, the current output is recalculated.
2. When PIDConfig.InputConfig.ErrorAction is set to TRUE, the alternative value specified by ErrorOut is output upon an error alarm. When the error is recovered, the current output is recalculated.

6 Appendix

6.1 ASCII Code Conversion

The ASCII code conversion table is as follows:

Bin (Binary)	Oct (Octal)	Dec (Decimal)	Hex (Hexadecimal)	Abbreviation/Character	Remarks
0000 0000	00	0	0x00	NUL (null)	Null character
0000 0001	01	1	0x01	SOH (start of headline)	Start of headline
0000 0010	02	2	0x02	STX (start of text)	Start of text
0000 0011	03	3	0x03	ETX (end of text)	End of text
0000 0100	04	4	0x04	EOT (end of transmission)	End of transmission
0000 0101	05	5	0x05	ENQ (enquiry)	Enquiry
0000 0110	06	6	0x06	ACK (acknowledge)	Acknowledgement
0000 0111	07	7	0x07	BEL (bell)	Bell
0000 1000	010	8	0x08	BS (backspace)	Backspace
0000 1001	011	9	0x09	HT (horizontal tab)	Horizontal tab
0000 1010	012	10	0x0A	LF (NL line feed, new line)	Line feed
0000 1011	013	11	0x0B	VT (vertical tab)	Vertical tab
0000 1100	014	12	0x0C	FF (NP form feed, new page)	Form feed
0000 1101	015	13	0x0D	CR (carriage return)	Carriage return
0000 1110	016	14	0x0E	SO (shift out)	Shift out
0000 1111	017	15	0x0F	SI (shift in)	Shift in
0001 0000	020	16	0x10	DLE (data link escape)	Data link escape
0001 0001	021	17	0x11	DC1 (device control 1)	Device control 1
0001 0010	022	18	0x12	DC2 (device control 2)	Device control 2
0001 0011	023	19	0x13	DC3 (device control 3)	Device control 3
0001 0100	024	20	0x14	DC4 (device control 4)	Device control 4

Bin (Binary)	Oct (Octal)	Dec (Decimal)	Hex (Hexadecimal)	Abbreviation/Character	Remarks
0001 0101	025	21	0x15	NAK (negative acknowledge)	Negative acknowledgement
0001 0110	026	22	0x16	SYN (synchronous idle)	Synchronous idle
0001 0111	027	23	0x17	ETB (end-of-transmission block)	End-of-transmission block
0001 1000	030	24	0x18	CAN (cancel)	Cancel
0001 1001	031	25	0x19	EM (end of medium)	End of medium
0001 1010	032	26	0x1A	SUB (substitute)	Substitute
0001 1011	033	27	0x1B	ESC (escape)	Escape (overflow)
0001 1100	034	28	0x1C	FS (file separator)	File separator
0001 1101	035	29	0x1D	GS (group separator)	Group separator
0001 1110	036	30	0x1E	RS (record separator)	Record separator
0001 1111	037	31	0x1F	US (unit separator)	Unit separator
0010 0000	040	32	0x20	(space)	Space
0010 0001	041	33	0x21	!	Exclamation mark
0010 0010	042	34	0x22	"	Double quotes
0010 0011	043	35	0x23	#	Hashtag
0010 0100	044	36	0x24	$	Dollar sign
0010 0101	045	37	0x25	%	Percent sign
0010 0110	046	38	0x26	&	Ampersand
0010 0111	047	39	0x27	'	Closed single quote
0010 1000	050	40	0x28	(	Open bracket
0010 1001	051	41	0x29	)	Closing bracket
0010 1010	052	42	0x2A	*	Asterisk
0010 1011	053	43	0x2B	+	Plus
0010 1100	054	44	0x2C	,	Comma

Bin (Binary)	Oct (Octal)	Dec (Decimal)	Hex (Hexadecimal)	Abbreviation/Character	Remarks
0010 1101	055	45	0x2D	-	Minus/Dash
0010 1110	056	46	0x2E	.	Period
0010 1111	057	47	0x2F	/	Slash
0011 0000	060	48	0x30	0	Character 0
0011 0001	061	49	0x31	1	Character 1
0011 0010	062	50	0x32	2	Character 2
0011 0011	063	51	0x33	3	Character 3
0011 0100	064	52	0x34	4	Character 4
0011 0101	065	53	0x35	5	Character 5
0011 0110	066	54	0x36	6	Character 6
0011 0111	067	55	0x37	7	Character 7
0011 1000	070	56	0x38	8	Character 8
0011 1001	071	57	0x39	9	Character 9
0011 1010	072	58	0x3A	:	Colon
0011 1011	073	59	0x3B	;	Semicolon
0011 1100	074	60	0x3C	<	Less than
0011 1101	075	61	0x3D	=	Equal sign
0011 1110	076	62	0x3E	>	Greater than
0011 1111	077	63	0x3F	?	Question mark
0100 0000	0100	64	0x40	@	Email symbol
0100 0001	0101	65	0x41	A	Uppercase A
0100 0010	0102	66	0x42	B	Uppercase B
0100 0011	0103	67	0x43	C	Uppercase C
0100 0100	0104	68	0x44	D	Uppercase D
0100 0101	0105	69	0x45	E	Uppercase E

Bin (Binary)	Oct (Octal)	Dec (Decimal)	Hex (Hexadecimal)	Abbreviation/Character	Remarks
0100 0110	0106	70	0x46	F	Uppercase F
0100 0111	0107	71	0x47	G	Uppercase G
0100 1000	0110	72	0x48	H	Uppercase H
0100 1001	0111	73	0x49	I	Uppercase I
01001010	0112	74	0x4A	J	Uppercase J
0100 1011	0113	75	0x4B	K	Uppercase K
0100 1100	0114	76	0x4C	L	Uppercase L
0100 1101	0115	77	0x4D	M	Uppercase M
0100 1110	0116	78	0x4E	N	Uppercase N
0100 1111	0117	79	0x4F	O	Uppercase O
0101 0000	0120	80	0x50	P	Uppercase P
0101 0001	0121	81	0x51	Q	Uppercase Q
0101 0010	0122	82	0x52	R	Uppercase R
0101 0011	0123	83	0x53	S	Uppercase S
0101 0100	0124	84	0x54	T	Uppercase T
0101 0101	0125	85	0x55	U	Uppercase U
0101 0110	0126	86	0x56	V	Uppercase V
0101 0111	0127	87	0x57	W	Uppercase W
0101 1000	0130	88	0x58	X	Uppercase X
0101 1001	0131	89	0x59	Y	Uppercase Y
0101 1010	0132	90	0x5A	Z	Uppercase Z
0101 1011	0133	91	0x5B	[	Open square bracket
0101 1100	0134	92	0x5C	\	Backslash
0101 1101	0135	93	0x5D	]	Closing square bracket
0101 1110	0136	94	0x5E	^	Caret

Bin (Binary)	Oct (Octal)	Dec (Decimal)	Hex (Hexadecimal)	Abbreviation/Character	Remarks
0101 1111	0137	95	0x5F	_	Underscore
0110 0000	0140	96	0x60	`	Opening single quote
0110 0001	0141	97	0x61	a	Lowercase a
0110 0010	0142	98	0x62	b	Lowercase b
0110 0011	0143	99	0x63	c	Lowercase c
0110 0100	0144	100	0x64	d	Lowercase d
0110 0101	0145	101	0x65	e	Lowercase e
0110 0110	0146	102	0x66	f	Lowercase f
0110 0111	0147	103	0x67	g	Lowercase g
0110 1000	0150	104	0x68	h	Lowercase h
0110 1001	0151	105	0x69	i	Lowercase i
0110 1010	0152	106	0x6A	j	Lowercase j
0110 1011	0153	107	0x6B	k	Lowercase k
0110 1100	0154	108	0x6C	l	Lowercase l
0110 1101	0155	109	0x6D	m	Lowercase m
0110 1110	0156	110	0x6E	n	Lowercase n
0110 1111	0157	111	0x6F	o	Lowercase o
0111 0000	0160	112	0x70	p	Lowercase p
0111 0001	0161	113	0x71	q	Lowercase q
0111 0010	0162	114	0x72	r	Lowercase r
0111 0011	0163	115	0x73	s	Lowercase s
0111 0100	0164	116	0x74	t	Lowercase t
0111 0101	0165	117	0x75	u	Lowercase u
0111 0110	0166	118	0x76	v	Lowercase v
0111 0111	0167	119	0x77	w	Lowercase w

Bin (Binary)	Oct (Octal)	Dec (Decimal)	Hex (Hexadecimal)	Abbreviation/Character	Remarks
0111 1000	0170	120	0x78	x	Lowercase x
0111 1001	0171	121	0x79	y	Lowercase y
0111 1010	0172	122	0x7A	z	Lowercase z
0111 1011	0173	123	0x7B	{	Opening curly bracket
0111 1100	0174	124	0x7C	|	Perpendicular
0111 1101	0175	125 m	0x7D	}	Closing curly bracket
0111 1110	0176	126	0x7E	to	Tilde
0111 1111	0177	127	0x7F	DEL (delete)	Delete

6.2 Error Codes

The software tool prompts various categories of error codes when faults occur in user programming. The following table lists the error codes and corresponding solutions.

Error code	Error message	Detailed description	Solution
Program			
1500	User program watchdog timed out	The user program execution time is too long and has exceeded the set program watchdog time.	Increase the watchdog time as appropriate, or check whether there is a program block with unexpectedly long execution time in the user program.
1501	Undefined instruction	The instruction is not supported.	Upgrade the PLC firmware to the version that supports the instruction.
1502	The user program is incomplete, and the length is incorrect.	The user program is incomplete, and the length is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1503	Program authorization protection identifier error. Check whether the identifier matches.	The program authorization protection identifier is incorrect. Check whether the authorization protection identifier of the device is set correctly.	Contact the equipment provider.

Error code	Error message	Detailed description	Solution
1504	User program empty	The user program is empty. There is no valid program.	Re-download the user program.
1505	The block POU identifier is incorrect.	The block POU identifier is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1510	The subprogram identifier is incorrect.	The subprogram identifier is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1511	The subprogram type is incorrect.	The subprogram type is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1512	The subprogram serial number is incorrect or out of range.	The subprogram serial number is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1513	The subprogram address is incorrect, duplicated, or conflicting.	The subprogram address is incorrect, duplicated, or conflicting.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1514	The interrupt subprogram serial number is incorrect or out of range.	The interrupt subprogram serial number is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1515	The interrupt subprogram address is incorrect, duplicated, or conflicting.	The interrupt subprogram address is incorrect, duplicated, or conflicting.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1516	The interrupt subprogram edge is incorrect (not rising edge or falling edge).	The interrupt subprogram edge is incorrect (not rising edge or falling edge).	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1517			

Error code	Error message	Detailed description	Solution
	The interrupt timing duration range of the interrupt subprogram timer is incorrect.	The interrupt timing duration range of the interrupt subprogram timer is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1520	The OBprog program identifier is incorrect.	The OBprog program identifier is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1521	The OBprog program type is incorrect.	The OBprog program type is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1522	The OBprog program serial number is incorrect or out of range.	The OBprog program serial number is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1523	The OBprog program address is incorrect, duplicated, or conflicting.	The OBprog program address is incorrect, duplicated, or conflicting.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1524	The variable quantity of the OBprog program is incorrect.	The variable quantity of the OBprog program is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1525	The variable length of the OBprog program is incorrect.	The variable length of the OBprog program is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1526	The header data of the OBprog program is incorrect.	The header data of the OBprog program is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1530	The LBL serial number of the CJ-LBL instruction is incorrect or out of range.	The LBL serial number of the CJ-LBL instruction is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.

Error code	Error message	Detailed description	Solution
1531	The LBL address of the CJ-LBL instruction is incorrect, duplicated, or conflicting.	The LBL address of the CJ-LBL instruction is incorrect, duplicated, or conflicting.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5001	Exception in user program execution or instruction return value error, some instructions not executed	Execution of the user program is abnormal or the return value of the instruction is incorrect, and some instructions are not executed, causing program execution to end abnormally.	Check the logic of the user program for any exception in execution process or execution logic.
5010	The subprogram serial number of the CALL instruction is incorrect or out of range.	The subprogram serial number of the CALL instruction is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5011	The subprogram of the CALL instruction does not exist or is not initialized.	The subprogram of the CALL instruction does not exist or is not initialized.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5012	The number of subprogram nesting levels of CALL instruction is out of range or no greater than 0.	The number of subprogram nesting levels of the CALL instruction is out of range.	Modify the program logic to reduce the subprogram nesting levels.
5013	The subprogram of the CALL instruction returns a relationship error.	The subprogram of the CALL instruction returns a relationship error.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5014	Mismatch between subprogram call and subprogram return	Subprogram execution is abnormal. The subprogram call and subprogram return do not match.	Check whether the subprogram call and return are disordered due to the abnormal end of the user program.
5015	Interrupt subprogram undefined	The interrupt subprogram is undefined or does not exist.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.

Error code	Error message	Detailed description	Solution
5016	The interrupt queue of the interrupt subprogram timer is full and the interrupt is lost.	The interrupt queue of the interrupt subprogram timer is full and the interrupt is lost.	Modify the interrupt subprogram attributes or logic, and reduce the number of interrupts as appropriate.
5020	The FBFC program serial number is incorrect or out of range.	The FBFC program serial number is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5021	The FBFC program does not exist or is not initialized.	The FBFC program does not exist or is not initialized.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5022	The variable of the FBFC program does not exist or is not initialized.	The variable of the FBFC program does not exist or is not initialized.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5023	The number of program nesting levels of FBFC instruction is out of range or no greater than 0.	The number of FBFC program nesting levels is out of range.	Modify the program logic to reduce the FBFC program nesting levels.
5024	The FBFC program returns a relationship error.	The FBFC program returns a relationship error.	Check whether the FBFC special instruction is used in the wrong position, or recompile and download the user program.
5025	Mismatch between OBprog program call and program return	OBprog program execution is abnormal. The program call and program return do not match.	Check whether the program call and return are disordered due to the abnormal end of the user program.
5030	The LBL serial number of the CJ-LBL instruction is incorrect or out of range.	The LBL serial number of the CJ-LBL instruction is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.

Error code	Error message	Detailed description	Solution
5031	The LBL of the CJ-LBL instruction does not exist or is not initialized.	The LBL of the CJ-LBL instruction does not exist or is not initialized.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5032	The number of nesting levels of FOR-NEXT instruction is out of range or no greater than 0.	The number of nesting levels of the FOR-NEXT instruction is out of range.	Modify the program logic to reduce the FOR-NEXT instruction nesting levels.
5033	The number of FOR-NEXT instruction loops is out of range or no greater than 0.	The number of FOR-NEXT instruction loops is out of range or no greater than 0.	Modify the program logic to change the number of FOR-NEXT instruction loops.
5034	The number of FOR-NEXT instruction loops equals 0.	The number of FOR-NEXT instruction loops equals 0.	Modify the program logic to change the number of FOR-NEXT instruction loops.
5035	The FOR and NEXT instructions are not paired.	The FOR and NEXT instructions are not paired.	Check whether the disorder is caused by abnormal stop of the user program.
5080	Array subscript access out of bounds	The array access subscript is greater than the maximum array subscript value, and the subscript value in use has been changed to the maximum array subscript value.	Double-click the fault code to go to the corresponding program position to modify the subscript value.
5081	The division-by-zero protection is triggered and the divisor 0 is replaced by 1 automatically.	The division-by-zero protection is triggered and the divisor 0 is replaced by 1 automatically.	Double-click the fault code to go to the corresponding program position to modify the divisor.
5082	The program loop has no response for a long time.	The program loop has no response for a long time.	Double-click the fault code to go to the corresponding program position to modify the loop statement.
5083	Array subscript access out of bounds	The array access subscript is less than 0, and the subscript value in use has been changed to 0.	Double-click the fault code to go to the corresponding program position to modify the subscript value.
5084	Invalid data	The floating-point data is invalid.	

Error code	Error message	Detailed description	Solution
			Check whether the input values of functions such as LN, LOG, SQRT are legal.
5101	The address of the parameter variable of the instruction is incorrect, or the variable does not exist.	The address of the parameter variable of the instruction is incorrect, or the variable does not exist.	Check whether the address of the parameter variable of the instruction is normal and whether the variable exists.
5102	The size of the parameter variable of the instruction is incorrect. The variable does not exist or is out of range.	The size of the parameter variable of the instruction is incorrect. The variable does not exist or is out of range.	Check whether the data length of the parameter variable of the instruction is out of range.
5103	xxxx0001: error	xxxx0001: error	xxxx0001: error
5104	The instruction parameter sequence or relationship is incorrect.	The instruction parameter sequence or relationship is incorrect.	Check whether the parameter sequence or relationship of the instruction is correct.
5105	The character string data or length of the string instruction is incorrect.	The character string data or length of the string instruction is incorrect.	Check whether the character string data of the string instruction is illegal.
5110	The serial number of the Pointer is incorrect or out of range.	The serial number of the Pointer is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5111	The Pointer is not initialized or does not point to a valid data variable.	The Pointer is not initialized or does not point to a valid data variable.	Check whether the Pointer is initialized and whether it points to a valid variable address.

Error code	Error message	Detailed description	Solution
5112	The variable pointed to by the Pointer does not exist or is out of range.	The variable pointed to by the Pointer does not exist or is out of range.	Check the variable address pointed to by the Pointer or initialize the Pointer again.
5113	The offset of the Pointer is out of range.	The offset of the Pointer is out of range.	Check whether the offset of the Pointer is too large. If yes, reduce the offset.
5114	The variable pointed to by the execution result of the Pointer does not exist or is out of range.	The variable pointed to by the execution result of the Pointer does not exist or is out of range.	Check whether the variable address pointed to by the execution result of the Pointer exists or is out of range.
5120	Failed to instantiate the counter instruction.	Failed to instantiate the counter instruction.	Recompile and download the user program.
5121	The comparand of the counter instruction is incorrect or out of range.	The comparand of the counter instruction is incorrect or out of range.	Check whether the comparand of the counter instruction is incorrect or out of range.
5130	Failed to instantiate the timer instruction.	Failed to instantiate the timer instruction.	Recompile and download the user program.
5131	The comparand of the timer instruction is incorrect or out of range.	The comparand of the timer instruction is incorrect or out of range.	Check whether the comparand of the timer instruction is incorrect or out of range.
5140	The number of SFC STL parallel branch/parallel recombination/ selective branch/ selective recombination lines is out of the range.	The number of SFC STL parallel branch/parallel recombination/ selective branch/selective recombination lines is out of the range.	Ensure that the number of SFC STL parallel branch/parallel recombination/selective branch/ selective recombination lines is within the specified range.

Error code	Error message	Detailed description	Solution
5150	Failed to instantiate the function block instruction.	Failed to instantiate the function block instruction.	Recompile and download the user program.
5160	The subscript variable code of the array is incorrect or does not exist.	The subscript variable code of the array is incorrect or does not exist.	Recompile and download the user program.
5161	Array subscript variable data error or out of range	The subscript variable of the array is incorrect or out of range.	Modify the value of the subscript variable so that the array falls within the allowable range.
5170	Failed to instantiate the PID_AT instruction.	Failed to instantiate the PID_AT instruction.	Recompile and download the user program.
5171	Failed to instantiate the PID_ATC instruction.	Failed to instantiate the PID_ATC instruction.	Recompile and download the user program.
5172	PID AT/PID ATC instruction input parameter PV/AI error	PID AT/PID ATC instruction input parameter PV/AI error	Check whether parameter PV/AI is input or the PVType parameter is set properly.
5600	Failed to instantiate the SerialSR instruction.	Failed to instantiate the SerialSR instruction.	Recompile and download the user program.
5601	The port ID of the SerialSR instruction is out of range.	The port ID of the SerialSR instruction is out of range.	Modify the port ID of the SerialSR instruction.
5602	The protocol of the SerialSR instruction is incorrect.	The protocol of the SerialSR instruction is incorrect.	Set the free protocol for the serial port by using the software tool.
5603	SerialSR instruction port conflict	Multiple instructions call the SerialSR instruction at the same time, and the instruction that fails to preempt the port reports an error.	Modify the instruction scheduling timing to implement time division multiplexing.
5604	The sent data length of SerialSR instruction is out of range or smaller than 0.	The sent data length of SerialSR instruction is out of range or smaller than 0.	Check whether the sent data length of SerialSR instruction is out of range or smaller than 0.

Error code	Error message	Detailed description	Solution
5605	SerialSR instruction TX data buffer error	Failed to obtain the TX data buffer of the SerialSR instruction.	Enable this instruction again.
5606	The received data length of SerialSR instruction is out of range or smaller than 0.	The received data length of SerialSR instruction is out of range or smaller than 0.	Check whether the received data length of SerialSR instruction is out of range or smaller than 0.
5607	SerialSR instruction RX data buffer error	Failed to obtain the RX data buffer of the SerialSR instruction.	Enable this instruction again.
6580	The axis ID specified in the CANopen axis instruction is invalid.	The axis ID specified in the CANopen axis instruction is invalid.	Modify the axis ID.
6701	Invalid memory address: element or variable non-existent	The memory address is invalid. The element or variable to access does not exist.	Modify the instruction parameter to use a valid element or variable.
6705	Invalid memory size: memory non-existent or out of range	The memory size is invalid. The number of elements or variables to access is too large or out of range.	Modify the instruction parameter to adjust the number of elements or variables.
6706	Improper data or data out of range	The instruction parameter is improper or out of the allowable range.	Refer to the instructions guide to modify the instruction parameter value.
6711	Invalid variable address: variable non-existent	The variable address is invalid. The element or variable to access does not exist.	Modify the instruction parameter to use a valid element or variable.
6712	Invalid variable size: variable out of range	The variable size is invalid. The number of elements or variables to access is too large or out of range.	Modify the instruction parameter to adjust the number of elements or variables.
6713	The variable encoding is invalid.	The variable encoding is invalid.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
CPU			
1011		FPGA initialization failure	

Error code	Error message	Detailed description	Solution
	FPGA initialization failure		The device hardware is faulty. Replace the device and return the faulty device to the factory for repair.
1012	Interrupt initialization failed	Interrupt initialization failed.	The device hardware is faulty. Replace the device and return the faulty device to the factory for repair.
1013	Timer interrupt initialization failed	Failed to initialize the timer interrupt of the user program.	Restart the device and try again, or replace the device and return the faulty device to the factory for repair.
5200	Error in data retention upon power failure	Check whether the function of data retention upon power failure works properly.	Check whether the function of data retention upon power failure works properly.
5238	2038 problem imminence warning	The device will not work normally after 11:14:07 on January 19, 2038 (UTC+8).	Change the device time.
5250	Low RTC battery voltage	The battery voltage of the RTC clock is low. If the device is powered off at this time, the system time will be restored to the initial value.	Replace the battery of the RTC clock while keeping the device powered on.
5900	Network down: Ethernet IP address conflict	When the device connects to the network or starts running after stop, or when its IP address is modified, it detects whether its IP address is used by other devices in the current network. If yes, the device automatically shuts down the network to avoid conflict.	Change the device IP address.
Local I/O			
5300	Initialization failed	Initialization failed.	The device hardware is faulty. Replace the device and return the faulty device to the factory for repair.
5301	Invalid DI filter parameter configuration	The DI filter parameter configuration data is invalid.	Modify the DI filter parameter configuration data.
Extension Module			

Error code	Error message	Detailed description	Solution
5400	Failed to initialize extension module interface hardware	The hardware of the extension module interface is faulty, which causes the initialization to fail.	The device hardware is faulty. Replace the device and return the faulty device to the factory for repair.
5401	Failed to parse extension module configuration data	The configuration data of the extension module cannot be parsed correctly because its format does not meet requirements.	Clear the compilation information using AutoShop and recompile and download the program. If the problem persists, delete the module configurations and add modules and configurations again one by one.
5402	Failed to initialize extension module interface slot	The slot of the extension module interface is faulty, which causes the initialization to fail.	1. Check whether the extension module interface slot is short-circuited. If yes, eliminate the short circuit. 2. Check whether the installed module hardware works properly. If not, replace the module.
5403	Extension module not installed	The extension module is configured but not installed.	Install the extension module as required, or modify the configuration of the extension module.
5404	Module installed inconsistent with module configured	The module installed in the slot must be inconsistent with the configured module; otherwise, it cannot work properly.	Install the extension module as required and modify module configuration accordingly to ensure consistency.
5405	Extension module interface hardware exception	The extension module interface is abnormal.	1. Check whether the extension module interface slot is short-circuited. If yes, eliminate the short circuit. 2. Check whether the installed module hardware works properly. If not, replace the module.
5406	The extension module interface software is abnormal.	The extension module interface software is abnormal.	1. Upgrade the PLC firmware. 2. If the problem persists after the firmware upgrade, replace the device and return the faulty device to the factory for repair.
5411	Module in the slot not powered	The module requires external power supply to function properly, but the external power supply is not on.	Connect the external power supply correctly according to the module specifications.
5412	Slot module hardware fault	The module has an internal fault and cannot work properly.	Replace the module and return the faulty module to the factory for repair.

Error code	Error message	Detailed description	Solution
5413	Slot module over-temperature	The module detected a high internal temperature that may lead to malfunction.	1. Do not install the module in an environment that does not meet the relevant temperature requirements. 2. Replace the module and return the faulty module to the factory for repair.
5419	Slot module channel input or output overflow	For the input channel, the input signal has exceeded the upper sampling threshold. Sampling cannot be performed properly, and there is a possibility that the input port may be burned. For the output channel, the output value of the corresponding channel has exceeded the set upper threshold, and signals cannot be output properly.	Input channel: Check the actual input signal value. If the signal input to this channel has exceeded the set sampling range under normal working conditions, modify the sampling range as appropriate. If the signal is abnormal, check the output device or instrument of the signal. Output channel: Check the set output value and ensure that the set output is within the set range. If the set range cannot meet requirements, modify it as appropriate.
5420	Slot module channel input or output underflow	For the input channel, the input signal has fallen below the lower sampling threshold, and sampling cannot be performed properly. For the output channel, the output value of the corresponding channel has fallen below the set lower threshold, and signals cannot be output properly.	Input channel: Check the actual input signal value. If the signal input to this channel has exceeded the set sampling range under normal working conditions, modify the sampling range as appropriate. If the signal is abnormal, check the output device or instrument of the signal. Output channel: Check the set output value and ensure that the set output is within the set range. If the set range cannot meet requirements, modify it as appropriate.
5421	Slot module channel input upper limit exceeded or current output disconnected	For the input channel, the input signal has exceeded the upper sampling threshold. At this time, the signal can be sampled normally but the accuracy cannot be guaranteed. For the current output channel, the	Input channel: Check the actual input signal value. If the signal input to this channel has exceeded the set sampling range under normal working conditions,

Error code	Error message	Detailed description	Solution
		output port is not connected to the load or the impedance of the connected load is too large, so that the current cannot be output normally.	modify the sampling range as appropriate. If the signal is abnormal, check the output device or instrument of the signal. Current output channel: Ensure that the load of the output port is connected properly and reliably, and that the load impedance is within the range specified in the module specifications.
5422	Slot module channel input lower limit exceeded or voltage output short-circuited	For the input channel, the input signal has fallen below the lower sampling threshold. At this time, the signal can be sampled normally but the accuracy cannot be guaranteed. For the voltage output channel, the output port is possibly short-circuited or the impedance of the connected load is too small, so that the voltage cannot be output normally.	Input channel: Check the actual input signal value. If the signal input to this channel has exceeded the set sampling range under normal working conditions, modify the sampling range as appropriate. If the signal is abnormal, check the output device or instrument of the signal. Voltage output channel: Ensure that the load of the output port is connected properly and reliably, and that the load impedance is within the range specified in the module specifications.
5423	Slot module channel input disconnected or output hardware faulty	For the input channel, no input signal is connected to the input port or the input signal is too weak and cannot be detected or sampled. For the output channel, the channel hardware is faulty and may have burned out.	Input channel: Ensure that the signal of the input port is normal and valid and is connected properly and reliably. Output channel: Replace the module and return the faulty module to the factory for repair.
Local encoder axis			
6300	Input device not assigned or assigned input device invalid	The local encoder axis must be assigned with a high-speed counter, and each high-speed counter can only be assigned to one axis, otherwise the axis cannot work properly.	Assign a high-speed counter that has not been assigned yet in "Input Device" on the "Basic Settings" page of the axis.
6301	Axis unit conversion configuration invalid	After a high-speed counter is assigned to an axis, its count value (pulse unit) is converted into the equivalent in user unit (Unit) according to the unit conversion	Check the settings on the "Unit Conversion Settings" page of the axis and correct the parameter values.

Error code	Error message	Detailed description	Solution
		setting parameter. If the number of pulses per revolution of the encoder, the displacement of the encoder per revolution, or the gear ratio of the transmission device is set incorrectly, the axis cannot work properly.	
6302	Axis software limit or revolution cycle configuration invalid	In linear mode, the negative limit must be less than 0, and the positive limit must be greater than 0. In rotary mode, the revolution cycle must be greater than 0. Since the high-speed counter is a 32-bit counter, the negative limit, positive limit, and revolution cycle must be 32-bit integers in the range of [-2147483648, +2147483647] after being converted into pulse units.	Linear mode: Modify the positive and negative limits to ensure that the negative limit is less than 0, the positive limit is greater than 0, and they are 32-bit integers in the range of [-2147483648, +2147483647] after being converted into pulse units. Rotary mode: Modify the revolution cycle to ensure that it is greater than 0 and is a 32-bit integer in the range of [-2147483648, +2147483647] after being converted into pulse units.
6303	Axis counting mode or signal source configuration invalid	The high-speed counter supports the following counting modes and signal sources: A/B phase frequency multiplication by 1: X0-A phase, X1-B phase, X2-A phase, X3-B phase A/B phase frequency multiplication by 2: X0-A phase, X1-B phase, X2-A phase, X3-B phase A/B phase frequency multiplication by 4: X0-A phase, X1-B phase, X2-A phase, X3-B phase CW/CCW: X0-CW, X1-CCW, X2-CW, X3-CCW Pulse+direction: X0-pulse, X1-direction, X2-pulse, X3-B-direction	Select a supported counting mode and signal source.
6304	Axis preset function: input terminal invalid	The preset function supports the input terminals X0, X1, X2, X3, X4, X5, X6, and X7.	Select an input terminal supported by the preset function.
6305	Axis probe 1: input terminal invalid	Probe 1 supports the input terminals X0, X1, X2, X3, X4, X5, X6, and X7.	Select an input terminal supported by probe 1.
6306	Axis probe 2: input terminal invalid	Probe 2 supports the input terminals X0, X1, X2, X3, X4, X5, X6, and X7.	Select an input terminal supported by probe 2.

Error code	Error message	Detailed description	Solution
6307	Axis comparison output: output terminal invalid	The comparative output supports the output terminals Y0, Y1, Y2, and Y3.	Select an output terminal supported by the comparison output.
6308	Axis comparison output: pulse width invalid	When the unit is ms, the time range is 0.1 ms to 6553.5 ms. When the unit is Unit, the set value must fall between 1 and 65535 after being converted into pulse units.	Modify the pulse width to ensure that it is within the allowable range.
CANlink			
6400	Station address conflict: Station address already exists in the network.	In CANlink communication, the addresses of all stations connected to the network must be unique. Address conflict detection is performed after a device node is powered on and initialized or the station address is modified. If the address is duplicated, a fault is reported and all CANlink bus activities of the node are stopped.	Change the station address to ensure that there are no duplicate addresses in the network..
6401	Slave offline	Failed to communicate with the slave because it is offline.	Check whether the CAN network connection works properly. Ensure that the connection is reliable without short circuit or open circuit, CANH and CANL are not reversely connected, and the terminal resistance is normal.
6411	Slave configuration exception response (1) “Undefined encoding used”	During configuration of a slave, the slave returns exception response (1) “Undefined encoding used”.	Check whether the type/model of the connected device is consistent with the configuration.
6412	Slave configuration exception response (2) “Configured index exceeds the maximum value supported by the node”	During configuration of a slave, the slave returns exception response (2) “Configured index exceeds the maximum value supported by the node”.	Check whether the type/model of the connected device is consistent with the configuration.
6413	Slave configuration exception response (3) “Register address non-	During configuration of a slave, the slave returns exception response (3) “Register address non-existent or inaccessible”.	Check whether the type/model of the connected device is consistent with the configuration.

Error code	Error message	Detailed description	Solution
	existent or inaccessible”		
6415	Slave configuration exception response (5) “Register data length invalid”	During configuration of a slave, the slave returns exception response (5) “Register data length invalid”.	Check whether the type/model of the connected device is consistent with the configuration.
6416	Waiting for slave configuration command response timed out	During configuration of a slave, waiting for slave response timed out.	Check whether the type/model of the connected device is consistent with the configuration.
6421	Slave synchronization exception response (1) “Illegal command code”	When a synchronization command is sent to a slave, the slave returns exception response (1) “Illegal command code”.	Check whether the type/model of the connected device is consistent with the configuration.
6422	Slave synchronization exception response (2) “Register address non-existent or inaccessible”	When synchronization data is sent to a slave, the slave returns exception response (2) “Register address non-existent or inaccessible”.	Check whether the type/model of the connected device is consistent with the configuration.
6423	Slave synchronization exception response (3) “Value beyond allowable range”	When synchronization data is sent to a slave, the slave returns exception response (3) “Value beyond allowable range”.	1. Check whether the set value in the corresponding register address has exceeded the allowed range. 2. Check whether the type/model of the connected device is consistent with the configuration.
6424	Slave synchronization exception response (4) “Operation unreachable or not allowed in the current state”	When synchronization data is sent to a slave, the slave returns exception response (4) “Operation unreachable or not allowed in the current state”.	Check whether the type/model of the connected device is consistent with the configuration.
6425	Slave synchronization exception response (5) “Data length invalid”	When synchronization data is sent to a slave, the slave returns exception response (5) “Data length invalid”.	Check whether the type/model of the connected device is consistent with the configuration.

Error code	Error message	Detailed description	Solution
6426	Waiting for slave synchronization command response timed out	Waiting for slave response to a synchronization command timed out.	Check whether the type/model of the connected device is consistent with the configuration.
CANopen			
6401	Node offline	Failed to communicate with the node because it is offline.	Check whether the CAN network connection works properly. Ensure that the connection is reliable without short circuit or open circuit, CANH and CANL are not reversely connected, and the terminal resistance is normal.
Modbus Master			
5500	8-bit data required for Modbus RTU serial port	Use 8-bit data for Modbus RTU serial port.	Use 8-bit data for Modbus RTU serial port.
6001	Slave returned exception response (01) “Illegal parameter”	The parameter received in the query is not an allowable action for the server (or slave). This may be because the parameter is only applicable to new devices, and is not implementable in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example, because it is unconfigured and is being asked to return register values.	Check whether the server (or slave) supports the parameter.
6002	Slave returned exception response (02) “Illegal data address”	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with the offset 96 and the length 4 will succeed, but a request with the offset 96 and the length 5 will result in exception code 02.	Check whether the corresponding parameter of the server (or slave) supports all the addresses accessed by this configuration.

Error code	Error message	Detailed description	Solution
6003	Slave returned exception response (03) “Illegal data”	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does not mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the Modbus protocol is unaware of the significance of any particular value of any particular register.	Check whether the value is within the allowed range.
6004	Slave returned exception response (04) “Slave device fault”	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.	Check whether the slave is abnormal or faulty.
6128	Response station number and requested station number mismatch	After the master sends a request frame, the station number in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6129	Response parameter and requested parameter mismatch	After the master sends a request frame, the parameter in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6130	Response data address and requested data address mismatch	After the master sends a request frame, the data address in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6131	Response data value and requested data value mismatch	After the master sends a request frame, the data value in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6240	Cache address mapping in	The cache address mapping in the configuration is invalid and the	Modify the cache address mapping in the configuration to a valid variable or element address.

Error code	Error message	Detailed description	Solution
	configuration invalid	configuration cannot be executed correctly.	
6255	Request timed out.	After sending a request frame, if the master does not receive a response from the slave within the specified timeout period, it retries according to the set number of retries. When the retry attempts exceed the set number, the master considers the slave abnormal and reports a request timeout error.	1. Ensure that the communication network cable is connected reliably. 2. Ensure that the slave station number is consistent with the configured slave station number. 3. Modify the timeout period to ensure that the master can receive the response frame within the timeout period. 4. Check whether the connected slave is a normal Modbus slave.
Modbus TCP Master			
6000	Configuration disconnected	The Modbus TCP client fails to establish a TCP connection with the server.	1. Ensure that the communication network cable is connected reliably. 2. Check whether the slave IP address and port ID are consistent with the configuration. 3. If the client and server are connected through a network bridge, router, or gateway, make sure that the client and server gateways are set correctly.
6001	Slave returned exception response (01) “Illegal parameter”	The parameter received in the query is not an allowable action for the server (or slave). This may be because the parameter is only applicable to new devices, and is not implementable in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example, because it is unconfigured and is being asked to return register values.	Check whether the server (or slave) supports the parameter.
6002	Slave returned exception response (02) “Illegal data address”	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer	Check whether the corresponding parameter of the server (or slave) supports all the addresses accessed by this configuration.

Error code	Error message	Detailed description	Solution
		length is invalid. For a controller with 100 registers, a request with the offset 96 and the length 4 will succeed, but a request with the offset 96 and the length 5 will result in exception code 02.	
6003	Slave returned exception response (03) “Illegal data”	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does not mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the Modbus protocol is unaware of the significance of any particular value of any particular register.	Check whether the value is within the allowed range.
6004	Slave returned exception response (04) “Slave device fault”	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.	Check whether the slave is abnormal or faulty.
6128	Response station number and requested station number mismatch	After the master sends a request frame, the station number in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6129	Response parameter and requested parameter mismatch	After the master sends a request frame, the parameter in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6130	Response data address and requested data address mismatch	After the master sends a request frame, the data address in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6131	Response data value and	After the master sends a request frame, the data value in the	Check whether the connected slave is a normal Modbus slave.

Error code	Error message	Detailed description	Solution
	requested data value mismatch	received response frame is inconsistent with that in the transmitted request frame.	
6240	Cache address mapping in configuration invalid	The cache address mapping in the configuration is invalid and the configuration cannot be executed correctly.	Modify the cache address mapping in the configuration to a valid variable or element address.
6255	Request timed out.	After sending a request frame, if the master does not receive a response from the slave within the specified timeout period, it retries according to the set number of retries. When the retry attempts exceed the set number, the master considers the slave abnormal and reports a request timeout error.	1. Ensure that the communication network cable is connected reliably. 2. Ensure that the slave station number is consistent with the configured slave station number. 3. Modify the timeout period to ensure that the master can receive the response frame within the timeout period. 4. Check whether the connected slave is a normal Modbus slave.
EtherCAT			
8001	Failed to read the master configuration information.	Failed to read the master configuration information.	Check whether the board software and software tool versions match.
8002	Failed to obtain slave configuration parameters.	Failed to obtain slave configuration parameters.	Check whether the board software and software tool versions match.
8003	EtherCAT startup timed out.	EtherCAT startup timed out.	1. Check whether the network is properly connected. 2. Check whether the connected slave is consistent with the configuration. 3. Check whether the slave type matches.
8004	Failed to request the master.	Failed to request the master.	Restart the PLC.
8200	Failed to write the slave startup parameters to the SDO.	Failed to write the slave startup parameters to the SDO.	1. Check whether there is an object dictionary that is not supported by the slave in the startup parameter list.

Error code	Error message	Detailed description	Solution
			2. Check whether the value of the object dictionary is out of range.
8201	The slave is lost during operation.	The slave is lost during operation.	1. Check whether the network with the slave is disconnected. 2. Check whether the slave is powered off. 3. Check whether the actual slave alias is consistent with the configured project name in the alias mode.
8202	The slave state machine is switched to non-OP mode.	The slave state machine is switched to non-OP mode.	1. Check whether the network with the slave is disconnected. 2. Check whether the slave is powered off.
8203	Slave state machine switching failed.	Slave state machine switching failed.	-
8204	The slave type is incorrect.	The slave type is incorrect.	1. Check whether the network cable is connected reversely. 2. Check whether the connected device matches the configuration.
8205	PDO address error	PDO address error	1. Check whether the memory runs out. 2. Check whether the software tool and board software versions match. 3. Power off and restart the PLC.
8206	Incorrect PDO length	Incorrect PDO length	Check whether the software tool and board software versions match.
8301	Failed to switch to INIT state.	Failed to switch to INIT state.	Check whether the slave station machine supports state transition.
8302	Failed to switch to PerOP state.	Failed to switch to PerOP state.	Check whether the slave supports the CoE protocol.
8304	Failed to switch to SafeOP state.	Failed to switch to SafeOP state.	Check whether the PDO communication configuration is correct.

Error code	Error message	Detailed description	Solution
8308	Failed to switch communication state.	Failed to switch communication state.	1. Check the network communication quality. 2. Check whether the EtherCAT task cycle is appropriate.
8310	FMMU unit configuration error	FMMU unit configuration error	Check whether the slave supports the FMMU unit.
8311	Email configuration error	Email configuration error	Check whether the slave supports the SM unit.
8400	ECTA configuration error	ECTA configuration error	Check whether the configured expansion module is consistent with the actually connected expansion module.
8401	ECTA hardware error	ECTA hardware error	1. Check whether the connection between the ECTA and the expansion module is loose. 2. Replace the ECTA.
8402	ECTA expansion module error	ECTA expansion module error	1. Locate the expansion module with the ERR indicator on. 2. Read the diagnosis object dictionary of the faulty module by using ETC_ReadParameter_CoE. 3. Determine the fault type based on description of the diagnosis object dictionary of the expansion module in the ECTA application guide and eliminate the fault.
Motion control axis			
9001	Local axis emergency stop active	The emergency stop terminal input is active, and the pulse output is stopped.	Disable the emergency stop terminal input, and then call the MC_Reset instruction to reset the fault.
9003	Overspeed	The pulse output frequency exceeds 200 kHz.	Check whether the pulse frequency obtained by multiplying the target velocity by the gear ratio exceeds 200 kHz.
9020	Homing error	The negative limit is not mapped.	Map the negative limit on the configuration interface.

Error code	Error message	Detailed description	Solution
9021	Homing error	The positive limit is not mapped.	Map the positive limit on the configuration interface.
9022	Homing error	The home signal is not mapped.	Map the home switch on the configuration interface.
9023	Homing error	1. The output frequency exceeds 200 kHz when the axis runs at the homing velocity. 2. The output frequency exceeds 200 kHz when the axis runs at the homing approach velocity.	1. Modify the unit conversion setting to ensure that the homing velocity and homing approach velocity do not exceed 200 kHz. 2. Change the homing velocity to ensure that the output frequency does not exceed 200 kHz. 3. Change the homing approach velocity to ensure that the output frequency does not exceed 200 kHz.
9024	Homing error	Homing timed out.	1. Check whether the limit signal and home signal can be connected normally. 2. Check whether the homing timeout time is too short.
9025	Homing error	The limit signal is incorrect during homing.	Check whether the limit signal that is not applicable to the current homing mode is triggered.
9030	Limit function enabled	The limit signal input is active during positioning.	Check whether the limit is reached during normal running.
9031	Synchronization error	The target number of transmitted pulses and the actual number of transmitted pulses do not match.	Check whether the limit is reached during normal positioning.
9101	Axis type error or non-existent	1. The type of the axis specified by AxisID is incorrect. 2. The axis specified by AxisID does not exist.	1. Check whether the instruction supports the axis specified by AxisID. 2. Check whether the axis specified by AxisID exists.
9102	Failed to configure the axis.	1. The axis configuration data is lost. 2. The axis configuration parameters are improper.	Check whether the parameters are correct.
9103	The “MC_Reset” instruction is called	The “MC_Reset” instruction is called when the axis is not faulty.	Check whether the “MC_Reset” instruction is called when the axis is not switched to ErrorStop state.

Error code	Error message	Detailed description	Solution
	when the axis is not faulty.		
9104	The axis is in unknown state when the MC_ReadStatus instruction is called.	The axis is in unknown state when the MC_ReadStatus instruction is called.	Check whether the current state of the axis is uncontrollable by using the online monitoring function.
9105	Current position setting not allowed	The MC_SetPosition instruction is called during running or stop.	Set the current position when the axis is in StandStill, Poweroff, or ErrorStop state.
9106	The axis is stopping upon a fault.	The axis is stopping upon a fault.	Execute the instruction after stop upon fault is completed, the fault is resolved, and the reset instruction is executed.
9107	The parameters are improper.	The parameters are improper.	Check whether the parameters on the left of the instruction are set properly.
9108	The PLCOpen state machine is improper.	The PLCOpen state machine is improper.	Check whether the current PLCOpen state machine satisfies the execution conditions for this instruction. If not, call the relevant instruction to switch the axis to the required state.
9110	The MC_Stop instruction is called repeatedly during stop.	The MC_Stop instruction is called repeatedly during stop.	Trigger only one MC_Stop instruction at a time.
9111	The instruction linked list is lost.	The instruction linked list is lost.	1. Check whether the software tool version and board version match. 2. Contact the manufacturer.
9112	AxisID changed	The value of AxisID is changed while the instruction flow is active.	Do not change the axis number while the flow is active for Enable instructions such as MC_Power and MC_Jog.
9113	Reset by executing the MC_Reset instruction timed out.	Reset by executing the MC_Reset instruction timed out.	1. Check whether the drive fault can be reset. 2. Check whether the axis fault type supports reset.

Error code	Error message	Detailed description	Solution
9114	The axis fails to write to 0x6060.	The axis fails to write to 0x6060.	1. Check for interference in network communication. 2. Check whether the slave supports the object dictionary 0x6060.
9115	The MC_Halt instruction is called when the axis is in Stopping state.	The MC_Halt instruction is called when the axis is in Stopping state.	Do not call the MC_Halt instruction when the axis is in Stopping state.
9116	The current axis is in online commissioning mode.	The current axis is in online commissioning mode.	Check whether the current axis is in online commissioning mode. PLC motion control instructions are invalid in online commissioning mode.
9117	Reserved	Reserved	Reserved
9118	The acceleration (deceleration) of the instruction exceeds the maximum acceleration.	The acceleration (deceleration) of the instruction exceeds the maximum acceleration.	Check whether the acceleration (deceleration) of the instruction exceeds the maximum acceleration.
9119	The target velocity of the MC_Jog instruction exceeds the maximum jogging velocity.	The target velocity of the MC_Jog instruction exceeds the maximum jogging velocity.	Check whether the target velocity of the MC_Jog instruction exceeds the maximum jogging velocity.
9120	The target velocity exceeds the maximum velocity.	The target velocity exceeds the maximum velocity.	Check whether the target velocity of the instruction exceeds the maximum velocity.
9121	The forward and reverse motion signals of the jog instruction are both active.	The forward and reverse motion signals of the jog instruction are both active.	Ensure that the forward and reverse motion signals of the jog instruction are not active at the same time.
9122	The control word is not mapped to the EtherCAT bus axis.	The control word is not mapped to the EtherCAT bus axis.	Add the control word in the PDO and map it to the axis.

Error code	Error message	Detailed description	Solution
9123	The target position is not mapped to the EtherCAT bus axis.	The target position is not mapped to the EtherCAT bus axis.	Add the target position in the PDO and map it to the axis.
9124	The target torque is not mapped to the EtherCAT bus axis.	The target torque is not mapped to the EtherCAT bus axis.	Add the target torque in the PDO and map it to the axis.
9125	The status word is not mapped to the EtherCAT bus axis.	The status word is not mapped to the EtherCAT bus axis.	Add the status word in the PDO and map it to the axis.
9126	The current position is not mapped to the EtherCAT bus axis.	The current position is not mapped to the EtherCAT bus axis.	Add the feedback position in the PDO and map it to the axis.
9127	The 0x60fd is not mapped to the EtherCAT bus axis.	The 0x60fd is not mapped to the EtherCAT bus axis.	Add 0x60fd in the PDO and map it to the axis.
9128	The current torque is not mapped to the EtherCAT bus axis.	The current torque is not mapped to the EtherCAT bus axis.	Add the current torque in the PDO and map it to the axis.
9129	The probe control word is not mapped to the EtherCAT bus axis.	The probe control word is not mapped to the EtherCAT bus axis.	Add the probe control word in the PDO and map it to the axis.
9130	The probe status word is not mapped to the EtherCAT bus axis.	The probe status word is not mapped to the EtherCAT bus axis.	Add the probe status word in the PDO and map it to the axis.
9131	The probe position is not mapped to the EtherCAT bus axis.	The probe position is not mapped to the EtherCAT bus axis.	Add the probe position in the PDO and map it to the axis.
9132	An interrupt positioning instruction is being executed and the probe channel is occupied.	An interrupt positioning instruction is being executed and the probe channel is occupied.	The probe instruction and interrupt positioning instruction must not occupy the same probe channel at the same time. When the two instructions are called simultaneously in the program,

Error code	Error message	Detailed description	Solution
			the interrupt positioning instruction takes priority..
9133	The imaginary axis mode is enabled.	The imaginary axis mode is enabled.	The current instruction does not support the imaginary axis mode.
9134	The imaginary axis probe is being used.	The imaginary axis probe is being used.	Two imaginary axis probes are supported. Check whether the current probe is out of range.
9135	The interrupt signal is not triggered in the interrupt positioning instruction.	The interrupt signal is not triggered in the interrupt positioning instruction.	During execution of the interrupt positioning instruction, no interrupt signal is detected after positioning is completed.
9136	The probe channel is occupied by another instruction during the interrupt positioning process.	The probe channel is occupied by another instruction during the interrupt positioning process.	Ensure that the probe channel is not occupied during the interrupt positioning process.
9137	The control mode 0x6060 is not mapped to the bus driver.	The control mode 0x6060 is not mapped to the bus driver.	Add 0x6060 in the PDO and map it to the axis.
9138	The control mode 0x6061 is not mapped to the bus driver.	The control mode 0x6061 is not mapped to the bus driver.	Add 0x6061 in the PDO and map it to the axis.
9139	The MC_Home instruction is called repeatedly during homing.	The MC_Home instruction is called repeatedly during homing.	Do not call the MC_Home instruction repeatedly during homing.
9140	The target torque of the instruction exceeds the maximum value.	The target torque of the instruction exceeds the maximum value.	Check whether the target torque of the instruction exceeds the positive and negative torque limits.
9141	The maximum velocity is not mapped to the bus driver.	The maximum velocity is not mapped to the bus driver.	Add “0x607f” in the PDO and map it to the axis.

Error code	Error message	Detailed description	Solution
9142	The immediate stop instruction is active.	The immediate stop instruction is active.	Check whether the immediate stop instruction has been called.
9143	The immediate stop instruction is called repeatedly.	The immediate stop instruction is called repeatedly.	Check whether the immediate stop instruction is called repeatedly.
9144	The limit is reached during jogging.	The limit is reached during jogging.	Check whether the limit is active.
9145	Target position exceeded 9999999	The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target position must not exceed this value.	1. Check whether the target position is correct. Set the target position again. 2. Change the gear ratio to ensure that the target position is not greater than 9999999.
9146	Target velocity exceeded 9999999	The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target velocity must not exceed this value.	1. Check whether the target velocity is correct. Set the target velocity again. 2. Change the gear ratio to ensure that the target velocity is not greater than 9999999.
9147	Target acceleration exceeded 9999999	The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target acceleration must not exceed this value.	1. Check whether the target acceleration is correct. Reset the target acceleration. 2. Change the gear ratio to ensure that the target acceleration is not greater than 9999999.
9148	Target deceleration exceeded 9999999	The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target deceleration must not exceed this value.	1. Check whether the target deceleration is correct. Reset the target deceleration again. 2. Change the gear ratio to ensure that the target deceleration is not greater than 9999999.
9149	Axis in sync control mode, abortion not allowed	1. A single-axis motion instruction is called when the axis is performing interpolation in sync control mode. The single-axis motion instruction reports an error.	Do not call single-axis motion instructions during interpolation.
9150	“MC_Halt” in execution, abortion not allowed	The MC_MoveSuperImposer instruction is called while the MC_Halt instruction is still active.	Do not call the MC_MoveSuperImposer instruction while the MC_Halt instruction is still active.

Error code	Error message	Detailed description	Solution
9151	The variable PulseWidth of the MC_MoveVelocityCSV instruction is out of range.	The variable PulseWidth of the MC_MoveVelocityCSV instruction is out of range.	Ensure that the parameter value is within the allowable range.
9152	The object dictionary 60FFh is not associated in I/O mapping of the bus servo axis when the MC_MoveVelocityCSV instruction is called.	The object dictionary 60FFh is not associated in I/O mapping of the bus servo axis when the MC_MoveVelocityCSV instruction is called.	Ensure that the object dictionary 60FFh is associated in I/O mapping of the bus servo axis when the MC_MoveVelocityCSV instruction is called.
9153	The probe terminal is not configured.	The probe terminal is not configured.	Check whether the software tool version supports configuration of the probe terminal ID.
9154	The value of ParameterIndex of the MC_SetAxisConfigPara instruction is out of range.	The value of ParameterIndex of the MC_SetAxisConfigPara instruction is out of range.	Ensure that the parameter value is within the allowable range.
9155	The configuration parameters of the axis are being modified, and execution of this instruction is not allowed before the modification is completed.	The configuration parameters of the axis are being modified, and execution of this instruction is not allowed before the modification is completed.	Perform the enable operation after axis initialization is completed.
9156	MC_SetAxisConfigPara does not support multi-execution.	MC_SetAxisConfigPara does not support multi-execution.	This instruction does not support re-execution or multi-execution.
9157	The gear/cam motion instruction is not supported by the axis due to axis properties.	The gear/cam motion instruction is not supported by the axis due to axis properties.	Ensure that the axis is not in single-axis mode or that the PLC supports the motion instruction.

Error code	Error message	Detailed description	Solution
9200	Failed to obtain the cam table configuration file.	Failed to obtain the cam table configuration file.	1. Check whether the board software and software tool match. 2. Re-download the cam configuration table.
9201	Failed to obtain the master axis.	Failed to obtain the master axis.	1. Check whether the master axis called in the program exists. 2. Check whether the master axis has reported an error. 3. Check whether the master axis is a profile axis.
9202	Failed to obtain the slave axis.	Failed to obtain the slave axis.	1. Check whether the slave axis called in the program exists. 2. Check whether the slave axis has reported an error.
9203	Failed to obtain the cam table.	Failed to obtain the cam table.	Check whether the cam table called in the program exists.
9204	The number of cams executed simultaneously in the PLC program exceeds the maximum allowable value.	The number of cams executed simultaneously in the PLC program exceeds the maximum allowable value.	Check whether the number of cams executed simultaneously in the program exceeds the threshold.
9205	The corresponding cam node is not found.	The corresponding cam node is not found.	1. This instruction can be called only when the slave axis is in cam engagement state. 2. The specified slave axis is not a cam node. Set the parameters again.
9206	The master axis is changed during cam engagement.	The master axis is changed during cam engagement.	Do not change the master axis during cam engagement.
9207	StartMode of the MC_CamIn instruction is out of range.	StartMode of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.
9208	StartPosition of the MC_CamIn instruction exceeds	StartPosition of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion

Error code	Error message	Detailed description	Solution
	the maximum allowable value.		control instruction does not exceed 9999999.
9209	MasterStartDistance of the MC_CamIn instruction exceeds the maximum allowable value.	MasterStartDistance of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9210	MasterScaling of the MC_CamIn instruction exceeds the maximum allowable value.	MasterScaling of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9211	SlaveScaling of the MC_CamIn instruction exceeds the maximum allowable value.	SlaveScaling of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9212	MasterOffset of the MC_CamIn instruction exceeds the maximum allowable value.	MasterOffset of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9213	SlaveOffset of the MC_CamIn instruction exceeds the maximum allowable value.	SlaveOffset of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9214	MasterScaling of the MC_CamIn instruction is not a positive number.	MasterScaling of the MC_CamIn instruction is not a positive number.	Set this parameter to a positive number.
9215	SlaveScaling of the MC_CamIn instruction is not a positive number.	SlaveScaling of the MC_CamIn instruction is not a positive number.	Set this parameter to a positive number.

Error code	Error message	Detailed description	Solution
9216	ReferenceType of the MC_CamIn/ MC_GearIn instruction is out of range.	ReferenceType of the MC_CamIn/ MC_GearIn instruction is out of range.	Ensure that the parameter value is within the specified range.
9217	Direction of the MC_CamIn instruction is out of range.	Direction of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.
9218	BufferMode of the MC_CamIn instruction is out of range.	BufferMode of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.
9219	The master axis phases in the node array of the cam table are not sorted in ascending order.	The master axis phases in the node array of the cam table are not sorted in ascending order.	Sort the master axis phases in ascending order when customizing cam table nodes.
9220	The curve type setting of the node array of the cam table is out of range.	The curve type setting of the node array of the cam table is out of range.	Check whether the curve type of the cam node array is set incorrectly.
9221	The target deceleration of the MC_CamOut instruction exceeds the maximum allowable value.	The target deceleration of the MC_CamOut instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9222	MC_CamOut target deceleration out of range	The target deceleration of the MC_GearOut instruction is out of range and causes stop upon a fault.	Ensure that the target deceleration is within the specified range.
9223	The target acceleration of the MC_Phasing instruction exceeds the maximum allowable value.	The target acceleration of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.

Error code	Error message	Detailed description	Solution
9224	The target acceleration of the MC_Phasing instruction is out of range.	The target acceleration of the MC_Phasing instruction is out of range.	Ensure that the target acceleration is within the specified range.
9225	The target velocity of the MC_Phasing instruction exceeds the maximum allowable value.	The target velocity of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9226	The target velocity of the MC_Phasing instruction is out of range.	The target velocity of the MC_Phasing instruction is out of range.	Ensure that the target deceleration is within the specified range.
9227	The curve type setting of the MC_GearOut instruction is out of range.	The curve type setting of the MC_GearOut instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.
9228	The value of Mode of the MC_CamOut instruction is out of range.	The value of Mode of the MC_CamOut instruction is out of range.	Ensure that the value of Mode is within the specified range.
9229	The MC_GenerateCamTable instruction detects that the cam node array is empty.	The MC_GenerateCamTable instruction detects that the cam node array is empty.	Contact Inovance for technical support.
9230	The node quantity specified by the MC_GenerateCamTable instruction exceeds the maximum allowable value.	The node quantity specified by the MC_GenerateCamTable instruction exceeds the maximum allowable value.	Check whether the target node quantity specified in the instruction is beyond the specified range.

Error code	Error message	Detailed description	Solution
9231	The value of Mode of the MC_GenerateCamTable instruction is out of range.	The value of Mode of the MC_GenerateCamTable instruction is out of range.	Ensure that the parameter value is within the specified range.
9232	The node quantity specified by the MC_GenerateCamTable instruction is too small.	The node quantity specified by the MC_GenerateCamTable instruction is too small.	Ensure that the node quantity is 2 or more.
9233	The RatioNumerator parameter in the gear instruction is set to 0.	The RatioNumerator parameter in the gear instruction is set to 0.	Set this parameter to a non-zero integer.
9234	The RatioDenominator parameter in the gear instruction is not greater than 0.	The RatioDenominator parameter in the gear instruction is not greater than 0.	Set this parameter to an integer greater than 0.
9235	The MC_GenerateCamTable instruction is being executed when the MC_SaveCamTable instruction is called.	The MC_GenerateCamTable instruction is being executed when the MC_SaveCamTable instruction is called.	Do not call the MC_SaveCamTable instruction before the cam table data update operation is completed.
9236	The MC_SaveCamTable instruction is being executed on the cam table when the MC_GenerateCamTable instruction is called.	The MC_SaveCamTable instruction is being executed on the cam table when the MC_GenerateCamTable instruction is called.	Do not call the MC_GenerateCamTable instruction before the cam table is saved.
9237	Failed to open the cam table file during execution of the	Failed to open the cam table file during execution of the MC_SaveCamTable instruction.	1. Check whether the PLC memory runs out. 2. Replace the PLC.

Error code	Error message	Detailed description	Solution
	MC_SaveCamTable instruction.		
9238	Failed to write the cam point quantity when the cam table is being saved.	Failed to write the cam point quantity when the cam table is being saved.	1. Check whether the PLC memory runs out. 2. Replace the PLC.
9239	Failed to write data when the cam table is being saved.	Failed to write data when the cam table is being saved.	1. Check whether the PLC memory runs out. 2. Replace the PLC.
9240	The phase of the first point is not 0.	The phase of the first point is not 0.	Ensure that the phase of the first point is 0.
9241	The displacement of the first point is not 0.	The displacement of the first point is not 0.	Ensure that the displacement of the first point is 0.
9242	The value of Mode of the MC_GearOut instruction is out of range.	The value of Mode of the MC_GearOut instruction is out of range.	Ensure that the value of Mode is within the specified range.
9243	The target deceleration of the MC_Phasing instruction exceeds the maximum allowable value.	The target deceleration of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9244	The target deceleration of the MC_GearIn instruction exceeds the maximum allowable value.	The target deceleration of the MC_GearIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9245	The value of Periodic of the MC_CamIn instruction is out of range.	The value of Periodic of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.

Error code	Error message	Detailed description	Solution
9246	The phase in the cam table exceeds the maximum allowable value.	The phase in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.
9247	The absolute value of the displacement in the cam table exceeds the maximum allowable value.	The absolute value of the displacement in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.
9248	The absolute value of the link velocity in the cam table exceeds the maximum allowable value.	The absolute value of the link velocity in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.
9249	The gear node is empty.	The gear node is empty.	Contact Inovance for technical support.
9250	The master axis and slave axis are the same.	The master axis and slave axis are the same.	Do not use the same axis as both the master axis and slave axis of the cam gear.
9251	The configuration address of the master axis is greater than or equal to that of the slave axis.	The configuration address of the master axis is greater than or equal to that of the slave axis.	When ReferenceType is set to set position of the current cycle, ensure that the configuration address of the master axis is less than that of the slave axis..
9252	The master axis filter coefficient fFilter[0] corresponding to the slave axis is out of range.	The master axis filter coefficient fFilter[0] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).
9253	The master axis filter coefficient fFilter[1] corresponding to the slave axis is out of range.	The master axis filter coefficient fFilter[1] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).

Error code	Error message	Detailed description	Solution
9254	The master axis filter coefficient fFilter[2] corresponding to the slave axis is out of range.	The master axis filter coefficient fFilter[2] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).
9255	The sum of the master axis filter coefficients corresponding to the slave axis is not 1.	The sum of the master axis filter coefficients corresponding to the slave axis is not 1.	Ensure that the sum of the master axis filter coefficients corresponding to the slave axis is 1.
9256	The start position and start distance of the master axis in the MC_CamIn instruction are improper.	The start position and start distance of the master axis in the MC_CamIn instruction are improper.	If the master axis works in linear mode and Direction in the instruction is set to positive, ensure that the cam synchronization point is not less than the cam engagement point.
9257	The start position and start distance of the master axis in the MC_CamIn instruction are improper.	The start position and start distance of the master axis in the MC_CamIn instruction are improper.	If the master axis works in linear mode and Direction in the instruction is set to negative, ensure that the cam synchronization point is not greater than the cam engagement point.
9258	The target deceleration of the MC_GearOut instruction exceeds the maximum allowable value.	The target deceleration of the MC_GearOut instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9259	MC_Phasing target deceleration out of range	The target deceleration of the MC_Phasing instruction is out of range and causes stop upon a fault.	Ensure that the target deceleration is within the specified range.
9260	MC_GearIn target deceleration out of range	The target deceleration of the MC_GearIn instruction is out of range and causes stop upon a fault.	Ensure that the target deceleration is within the specified range.
9261	MC_GearOut target deceleration out of range	The target deceleration of the MC_GearOut instruction is out of range and causes stop upon a fault.	Ensure that the target deceleration is within the specified range.

Error code	Error message	Detailed description	Solution
9262	The target acceleration of the MC_GearIn instruction exceeds the maximum allowable value.	The target acceleration of the MC_GearIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9263	The target acceleration of the MC_GearIn instruction is out of range.	The target acceleration of the MC_GearIn instruction is out of range.	Ensure that the target acceleration is within the specified range.
9264	MC_Phasing curve type setting out of range	The curve type setting of the MC_GearOut instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.
9265	MC_GearIn curve type setting out of range	The curve type setting of the MC_GearOut instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.
9266	MC_GearOut curve type setting out of range	The curve type setting of the MC_GearOut instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.
9267	The slave axis is modified during the cam operation.	The slave axis is modified during the cam operation.	Do not modify the slave axis during the cam operation.
9268	The value of PhasingMode of the MC_Phasing instruction is out of range.	The value of PhasingMode of the MC_Phasing instruction is out of range.	Ensure that parameters are within proper ranges.
9269	The current axis is not in cam control mode when the MC_CamOut instruction is called.	The current axis is not in cam control mode when the MC_CamOut instruction is called.	Ensure that the axis works in cam control mode when the MC_CamOut instruction is called.

Error code	Error message	Detailed description	Solution
9270	The current axis is not in gear control mode when the MC_GearOut instruction is called.	The current axis is not in gear control mode when the MC_GearOut instruction is called.	Ensure that the axis works in gear control mode when the MC_GearOut instruction is called.
9271	The position change of the master axis is too large within a single EtherCAT cycle during cam/gear operation.	The position change of the master axis is too large within a single EtherCAT cycle during cam/gear operation.	Ensure that the position change of the master axis is not greater than half a cam cycle within a single EtherCAT cycle.
9272	The point specified by Phase in the MC_GetCamTableDistance instruction does not fall between the start and end points.	The point specified by Phase in the MC_GetCamTableDistance instruction does not fall between the start and end points.	Ensure that the point specified by Phase is within the specified curve.
9273	The slave axis is changed during execution of the MC_GearIn instruction.	The slave axis is changed during execution of the MC_GearIn instruction.	Do not change the slave axis during execution of the MC_GearIn instruction.
9274	The value of Channel of the MC_DigitalCamSwitch instruction is out of range.	The value of Channel of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.
9275	The axis is not found.	The axis is not found.	Ensure that the axis specified by Axis exists.
9276	The number of tappets allowed to be executed at the same time is out of range.	The number of tappets allowed to be executed at the same time is out of range.	Ensure that the number of tappets allowed to be executed at the same time is no more than 8.
9277	The value of ReferenceType of the MC_DigitalCamSwi	The value of ReferenceType of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.

Error code	Error message	Detailed description	Solution
	tch instruction is out of range.		
9278	The value of Number of the MC_DigitalCamSwitch instruction is out of range.	The value of Number of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.
9279	The Switches array of the MC_DigitalCamSwitch instruction is empty.	The Switches array of the MC_DigitalCamSwitch instruction is empty.	Check that the connected device meets the requirements.
9280	The value of fPosition of the tappet array is out of range.	The value of fPosition of the tappet array is out of range.	Ensure that the parameter value is within the specified range.
9281	The value of iMode of the tappet array is out of range.	The value of iMode of the tappet array is out of range.	Ensure that the parameter value is within the specified range.
9282	The value of iDirection of the tappet array is out of range.	The value of iDirection of the tappet array is out of range.	Ensure that the parameter value is within the specified range.
9283	The value of fParameter of the tappet array is out of range.	The value of fParameter of the tappet array is out of range.	Ensure that the parameter value is within the specified range.
9284	When the tappet comparison point is set to time mode, the time setting is out of range.	When the tappet comparison point is set to time mode, the time setting is out of range.	Ensure that the parameter value is within the specified range.

Error code	Error message	Detailed description	Solution
9285	The selected axis is not under cam control when ReferenceType of the MC_DigitalCamSwitch instruction is set to 3.	The selected axis is not under cam control when ReferenceType of the MC_DigitalCamSwitch instruction is set to 3.	Call the instruction after the cam control takes effect.
9286	Axis communication is interrupted during tappet execution.	Axis communication is interrupted during tappet execution.	Ensure that axis communication is not interrupted during tappet execution.
9287	The comparison position start points are the same during tappet execution.	The comparison position start points are the same during tappet execution.	Ensure that the start points are not duplicate.
9288	The comparison position start and end point are the same during tappet execution.	The comparison position start and end point are the same during tappet execution.	Ensure that the start and end points are not duplicate.
9289	The selected tappet terminal is being used by another function.	The selected tappet terminal is being used by another function.	Check whether the terminal is set as the pulse output axis.
9290	The MC_DigitalCamSwitch instruction cannot be executed because the state of the motion control axis is improper.	The MC_DigitalCamSwitch instruction cannot be executed because the state of the motion control axis is improper.	Do not execute the MC_DigitalCamSwitch instruction in homing mode.
9291	The MasterSyncPosition setting in the MC_GearInPos instruction is out of range.	The MasterSyncPosition setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.

Error code	Error message	Detailed description	Solution
9292	The SlaveSyncPosition setting in the MC_GearInPos instruction is out of range.	The SlaveSyncPosition setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.
9293	The MasterStarDistance setting in the MC_GearInPos instruction is out of range.	The MasterStarDistance setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.
9294	The Velocity setting in the MC_GearInPos instruction exceeds the system limit.	The Velocity setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.
9295	The Velocity setting in the MC_GearInPos instruction exceeds the setting limit.	The Velocity setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.
9296	The Acceleration setting in the MC_GearInPos instruction exceeds the system limit.	The Acceleration setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.
9297	The Acceleration setting in the MC_GearInPos instruction exceeds the setting limit.	The Acceleration setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.
9298	The Deceleration setting in the MC_GearInPos instruction exceeds the system limit.	The Deceleration setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.
9299	The Deceleration setting in the MC_GearInPos	The Deceleration setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.

Error code	Error message	Detailed description	Solution
	instruction exceeds the setting limit.		
9300	The AvoidReversal setting in the MC_GearInPos instruction is out of range.	The AvoidReversal setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.
9301	The master axis speed is zero when the MC_GearInPos instruction is started.	The master axis speed is zero when the MC_GearInPos instruction is started.	Ensure that the master axis speed is not zero when starting this instruction.
9302	The master axis did not move during the catching phase of the MC_GearInPos instruction.	The master axis did not move during the catching phase of the MC_GearInPos instruction.	When MasterStarDistance is set to 0, ensure that the input MasterSyncPosition does not overlap with the current position of the master axis.
9303	When the MC_GearInPos instruction is started, the speed of the slave axis is not zero before entering the catching phase.	When the MC_GearInPos instruction is started, the speed of the slave axis is not zero before entering the catching phase.	Ensure that the slave axis remains stationary before entering the catching phase.
9304	Failed to enter the catching phase when the MC_GearInPos instruction is executed.	Failed to enter the catching phase when the MC_GearInPos instruction is executed.	Ensure that the master axis can enter the catching phase under the current position and motion direction conditions.
9305	The velocity of the slave axis exceeds the limit during execution of the MC_GearInPos instruction.	The velocity of the slave axis exceeds the limit during execution of the MC_GearInPos instruction.	Ensure that the parameter value is within the specified range.

Error code	Error message	Detailed description	Solution
9306	The cam node array specified by the MC_GenerateCamTable instruction is abnormal.	The cam node array specified by the MC_GenerateCamTable instruction is abnormal.	Ensure that the cam node array specified by the instruction is not the node array of the cam system variable.
9400	Maximum axis group quantity exceeded	The number of axis groups exceeds the maximum allowable value.	Reduce the number of axis groups in the project so that it does not exceed the maximum value.
9401	An axis in the axis group is faulty.	An axis in the axis group is faulty.	Locate the faulty axis, view the fault codes of the axis, and rectify the fault.
9402	The number of buffered interpolation instructions is greater than 8.	The number of buffered interpolation instructions is greater than 8.	Check whether the number of buffered interpolation instructions is greater than 8.
9403	Axis reused	An axis in the axis group is reused.	Each axis can be used in only one axis group. Check whether there is a reused axis in the axis group and replace it with an unused axis..
9404	Failed to create axis group	The x-axis or y-axis does not exist.	Check whether the x-axis and y-axis exist. An axis group consists of at least the x-axis and y-axis.
9405	The specified z-axis does not exist.	The z-axis is specified in the instruction but does not exist in the configuration.	Check whether the Z-axis specified in the instruction exists.
9406	Specified auxiliary axis non-existent	The auxiliary axis is specified in the instruction but does not exist in the configuration.	Check whether the auxiliary axis specified in the instruction exists.
9407	Axis group ID duplicated	The specified axis group ID has been used.	Change the axis group ID because the axis group ID must be unique.
9408	Failed to configure the axis.	Failed to configure the axis.	Check whether any axis in the axis group fails to be configured. If yes, check whether the board software and the software tool match.
9409	The axis ID is less than 0.	The axis ID is less than 0.	Check whether the ID of an axis in the axis group is less than 0.
9410			

Error code	Error message	Detailed description	Solution
	Axis group not released	The axis group is not released because the same MC_SetAxesGroup instruction is triggered repeatedly in a short time period.	Do not re-trigger the MC_SetAxesGroup instruction while its Busy signal output is still active.
9411	The MC_GroupStop instruction is aborted.	The MC_GroupStop instruction is aborted.	Check whether an instruction with higher priority is called while the MC_GroupStop instruction is still active.
9412	The value of CircAxes of the circular interpolation instruction is out of range.	The value of CircAxes of the circular interpolation instruction is out of range.	Check whether the value of CircAxes of the circular interpolation instruction is out of range.
9413	The value of CircMode of the circular interpolation instruction is out of range.	The value of CircMode of the circular interpolation instruction is out of range.	Check whether the value of CircMode of the circular interpolation instruction is out of range.
9414	The value of PathChoice of the circular interpolation instruction is out of range.	The value of PathChoice of the circular interpolation instruction is out of range.	Check whether the value of PathChoice of the circular interpolation instruction is out of range.
9415	The value of StopMode of the stop instruction is out of range.	The value of StopMode of the stop instruction is out of range.	Check whether the value of StopMode of the stop instruction is out of range.
9416	The x-axis is set to ring mode.	The x-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9417	The y-axis is set to ring mode.	The y-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9418	The z-axis is set to ring mode.	The z-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.

Error code	Error message	Detailed description	Solution
9419	The auxiliary axis is set to ring mode.	The auxiliary axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9420	The circular interpolation instruction is triggered repeatedly.	The circular interpolation instruction is triggered repeatedly.	Do not re-trigger the same circular interpolation instruction while its Busy signal output is still active.
9421	The linear interpolation instruction is triggered repeatedly.	The linear interpolation instruction is triggered repeatedly.	Do not re-trigger the same linear interpolation instruction while its Busy signal output is still active.
9422	Failed to obtain the axis group.	Failed to obtain the axis group.	Check whether the axis group specified by GroupID has been created by calling MC_SetAxesGroup.
9423	Failed to configure the axis.	Failed to configure the axis.	Check whether an instruction is triggered when axis configuration is not completed. Check whether the communication state of all axes in the axis group is Axis ready.
9424	An axis is disabled.	An axis is disabled.	Do not call the interpolation instruction when any axis is in Disabled state.
9425	Axis in execution of single-axis motion instruction	The interpolation instruction is triggered when an axis is executing a single-axis motion instruction.	Do not call the interpolation instruction when any axis is executing single-axis motion instructions and not in StandStill state.
9426	An axis is in Stopping state.	An axis is in Stopping state.	Do not call the interpolation instruction when any axis is in Stopping state after executing the MC_Stop instruction.
9427	The axis group is in Stopping state.	The axis group is in Stopping state.	Do not call the interpolation instruction while the MC_GroupStop instruction is still active.
9428	An axis is in Homing state.	An axis is in Homing state.	Do not call the interpolation instruction when any axis is in Homing state after executing the MC_Home instruction.
9429		An axis is executing the position setting instruction.	Do not call the interpolation instruction when any axis is setting the current

Error code	Error message	Detailed description	Solution
	An axis is executing the position setting instruction.		position by executing the MC_SetPosition instruction.
9430	An axis is in commissioning state.	An axis is in commissioning state.	Do not call the interpolation instruction when any axis is in commissioning state.
9431	An axis enters the commissioning state during interpolation, which aborts instruction execution of other axes.	An axis enters the commissioning state during interpolation, which aborts instruction execution of other axes.	Check whether any axis enters the commissioning state during interpolation and aborts instruction execution of other axes.
9432	Failed to request the memory.	Failed to request the memory.	Check whether the memory runs out. Contact Inovance.
9433	The target velocity is 0 or less than 0.	The target velocity is 0 or less than 0.	Ensure that the target velocity of the instruction is greater than 0.
9434	The target acceleration is 0 or less than 0.	The target acceleration is 0 or less than 0.	Ensure that the target acceleration of the instruction is greater than 0.
9435	The target deceleration is 0 or less than 0.	The target deceleration is 0 or less than 0.	Ensure that the target deceleration of the instruction is greater than 0.
9436	The curve type setting is out of range.	The curve type setting is out of range.	Check whether the curve type is set to a value other than the T-shaped curve for the interpolation instruction.
9437	AbsRelMode is set incorrectly.	AbsRelMode is set incorrectly.	Check whether the parameter is set to a value other than the absolute positioning and relative positioning modes.
9438	BufferMode is set incorrectly.	BufferMode is set incorrectly.	Check whether the value of BufferMode is proper.
9439	InsertMode is set incorrectly.	InsertMode is set incorrectly.	Check whether the value of InsertMode is proper.
9440	An axis stops due to a fault.	An axis stops due to a fault.	Locate the faulty axis and rectify the fault based on the fault code.

Error code	Error message	Detailed description	Solution
9441	The MC_GroupStop instruction is called repeatedly.	The MC_GroupStop instruction is called repeatedly.	Do not re-trigger an MC_GroupStop instruction or call other MC_GroupStop instructions while an MC_GroupStop instruction is still active.
9442	Data buffer area not empty	The data buffer area is not empty. It is an internal fault.	Contact Inovance.
9443	Not a circle	No circle can be drawn due to improper parameter settings.	Update the parameter settings.
9444	Not a circle	The start, end, and border points in the circular interpolation instruction are the same point, and no circle can be drawn.	Check the input parameters of the circular interpolation instruction and ensure that the start, end, and border points can form a circle.
9445	The instruction buffer area is full.	The instruction buffer area is full.	Contact Inovance for technical support.
9446	The velocity of the x-axis exceeds the maximum allowable velocity.	The velocity of the x-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the x-axis is not greater than the maximum allowable velocity.
9447	The velocity of the y-axis exceeds the maximum allowable velocity.	The velocity of the y-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the y-axis is not greater than the maximum allowable velocity.
9448	Z-axis exceeding maximum velocity	The target velocity of the z-axis is greater than the maximum allowable velocity.	Ensure that the target velocity of the z-axis is not greater than the maximum allowable velocity.
9449	The velocity of the auxiliary axis exceeds the maximum allowable velocity.	The velocity of the auxiliary axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the auxiliary axis is not greater than the maximum allowable velocity.
9450	Failed to obtain the number of axis groups.	Failed to obtain the number of axis groups.	Update the software tool to the latest version.
9451	Internal fault	Internal fault	Contact Inovance.
9452	The instruction is called when the	The instruction is called when the axis is in StandStill state.	Do not call this instruction when the axis is StandStill state.

Error code	Error message	Detailed description	Solution
	axis is in StandStill state.		
9453	Maximum velocity exceeded	The maximum velocity specified on the axis group configuration interface is exceeded.	Check whether the target velocity of the instruction is greater than the maximum velocity specified on the axis group configuration interface.
9454	The maximum allowable acceleration (deceleration) is exceeded.	The maximum allowable acceleration (deceleration) is exceeded.	Check whether the target acceleration (deceleration) of the instruction is greater than the maximum acceleration (deceleration) specified on the axis group configuration interface.
9455	The axis group becomes faulty due to an error reported by the linear interpolation instruction.	The axis group becomes faulty due to an error reported by the linear interpolation instruction.	Identify the first linear interpolation instruction that reports the error and troubleshoot the fault based on the fault code.
9456	The axis group becomes faulty due to an error reported by the circular interpolation instruction.	The axis group becomes faulty due to an error reported by the circular interpolation instruction.	Identify the first circular interpolation instruction that reports the error and troubleshoot the fault based on the fault code.
9457	The axis group becomes faulty due to an error reported by the axis group stop instruction.	The axis group becomes faulty due to an error reported by the axis group stop instruction.	Identify the first axis group stop instruction that reports the error and troubleshoot the fault based on the fault code.
9458	The axis group becomes faulty due to an error reported by the axis group pause instruction.	The axis group becomes faulty due to an error reported by the axis group pause instruction.	Identify the first axis group pause instruction that reports the error and troubleshoot the fault based on the fault code.
9459	The x-axis in the axis group is performing the interpolation algorithm of another axis group.	The x-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9460			

Error code	Error message	Detailed description	Solution
	The y-axis in the axis group is performing the interpolation algorithm of another axis group.	The y-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9461	The z-axis in the axis group is performing the interpolation algorithm of another axis group.	The z-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9462	The auxiliary axis in the axis group is performing the interpolation algorithm of another axis group.	The auxiliary axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9463	Axes in synchronous mode but not under axis group control when the MC_GroupStop instruction is called	When the MC_GroupStop instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the MC_GroupStop instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the MC_GroupStop instruction when the axes enter the synchronous mode due to other instructions.
9464	Axes in synchronous mode but not under axis group control when the linear or circular interpolation instruction is called	When the linear or circular interpolation instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the linear or circular interpolation instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the linear or circular interpolation instruction when the axes enter the synchronous mode due to other non-axis-group instructions.
9465	Axes in synchronous mode but not under axis group control when the MC_GroupPause instruction is called	When the MC_GroupPause instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the MC_GroupStop instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the MC_GroupStop instruction when the axes enter the synchronous mode due to other instructions.

Error code	Error message	Detailed description	Solution
9466	NumOfTurns in the MC_MoveEllipse instruction is set incorrectly.	NumOfTurns in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9467	AddLength in the MC_MoveEllipse instruction is set incorrectly.	AddLength in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9468	The axis stops due to an error of the MC_MoveEllipse instruction.	The axis stops due to an error of the MC_MoveEllipse instruction.	Check the fault code of the MC_MoveEllipse instruction with an exception to locate the fault.
9469	CircAxes in the MC_MoveEllipse instruction is set incorrectly.	CircAxes in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9470	CircMode in the MC_MoveEllipse instruction is set incorrectly.	CircMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9471	PathChoice in the MC_MoveEllipse instruction is set incorrectly.	PathChoice in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9472	Velocity in the MC_MoveEllipse instruction is set incorrectly.	Velocity in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9473	Acceleration in the MC_MoveEllipse instruction is set incorrectly.	Acceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9474	Deceleration in the MC_MoveEllipse instruction is set incorrectly.	Deceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9475	BufferMode in the MC_MoveEllipse	BufferMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.

Error code	Error message	Detailed description	Solution
	instruction is set incorrectly.		
9476	Failed to draw an ellipse because the center and the lengths of the major and minor axes are set incorrectly.	Failed to draw an ellipse because the center and the lengths of the major and minor axes are set incorrectly.	Ensure that the parameter value is within the allowable range.
9477	The property of the x-axis in the axis group instruction does not support the interpolation motion.	The property of the x-axis in the axis group instruction does not support the interpolation motion.	Ensure that the x-axis is not in single-axis mode.
9478	The property of the y-axis in the axis group instruction does not support the interpolation motion.	The property of the y-axis in the axis group instruction does not support the interpolation motion.	Ensure that the y-axis is not in single-axis mode.
9479	The property of the z-axis in the axis group instruction does not support the interpolation motion.	The property of the z-axis in the axis group instruction does not support the interpolation motion.	Ensure that the z-axis is not in single-axis mode.
9480	The property of the auxiliary axis in the axis group instruction does not support the interpolation motion.	The property of the auxiliary axis in the axis group instruction does not support the interpolation motion.	Ensure that the auxiliary axis is not in single-axis mode.
9501	EtherCAT bus drive error	A drive error occurs. The fault code in the object dictionary 0x603F of the drive is 0xxxxx (hexadecimal).	Determine the drive fault type according to the bus drive guide and rectify the fault.
9502	The drive is disabled.	The drive is disabled.	1. Check whether the drive status word 0x6041 switches to the disabled state during motion.

Error code	Error message	Detailed description	Solution
			2. Check whether communication is disconnected during motion.
9503	The limit is reached.	The limit is reached.	1. Check whether the software limit is configured and reached. 2. Check whether the hardware limit is reached.
9505	Failed to modify the control mode.	Failed to modify the control mode.	1. Check for interference in network communication. 2. Check whether the drive supports the object dictionary 0x6060.
9508	Homing failed	Homing failed	1. Identify the cause of the drive homing failure based on the fault code. 2. Check whether homing timed out.
9509	An axis internal calculation precision error occurs.	An axis internal calculation precision error occurs.	Check whether the floating-point data of the instruction falls beyond the single-precision floating-point number range.
9510	The following error is out of range.	The following error is out of range.	1. Check whether the acceleration is too large. 2. Check whether the set following error is too small.
9512	The servo drive is disconnected during operation.	The servo drive is disconnected during operation.	1. Check whether the drive works properly. 2. Check whether the network cable is properly connected. 3. Check for strong interference in communication.
9513	Homing failed due to a drive fault	Homing failed due to a drive fault.	Check the fault code of the drive to eliminate the fault.
9514	Homing failed because the homing offset exceeded 32 bits.	Homing failed because the homing offset exceeded 32 bits.	Check whether the homing offset multiplied by the gear ratio exceeds 32 bits; if yes, change the gear ratio.
9515	Homing failed due to loss of the slave	Homing failed because the EtherCAT drive is lost.	Contact Inovance for technical support.
9516			

Error code	Error message	Detailed description	Solution
	Homing failed because the SDO failed to write to object dictionary 0x607C.	Homing failed because the SDO failed to write to object dictionary 0x607C.	1. Check whether the drive supports 0x607C. 2. Check the network communication quality.
9517	Homing failed because SDO failed to write 6 into object dictionary 0x6060.	Homing failed because SDO failed to write 6 into object dictionary 0x6060.	1. Set 0x6060 in the PDO. 2. Check the network communication quality.
9518	Homing failed because the SDO failed to read object dictionary 0x6061.	Homing failed because the SDO failed to read object dictionary 0x6061.	1. Set 0x6061 in the PDO. 2. Check the network communication quality.
9519	Homing failed because SDO failed to write 8 into object dictionary 0x6060.	Homing failed because SDO failed to write 8 into object dictionary 0x6060.	1. Set 0x6060 in the PDO. 2. Check the network communication quality.
9551	Failed to switch the control mode.	Failed to switch the control mode.	Check for interference in network communication.
9552	The target velocity is 0.	The target velocity is 0.	Check whether the target velocity of position instructions is appropriate.
9601	The axis stops due to parameter exception of the MC_MoveAbsolute instruction.	The axis stops due to parameter exception of the MC_MoveAbsolute instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9602	The axis stops due to parameter exception of the MC_MoveRelative instruction.	The axis stops due to parameter exception of the MC_MoveRelative instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9603	The axis stops due to exception of the MC_MoveVelocity instruction.	The axis stops due to exception of the MC_MoveVelocity instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9604			

Error code	Error message	Detailed description	Solution
	The axis stops due to exception of the MC_Jog instruction.	The axis stops due to exception of the MC_Jog instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9605	The axis stops due to exception of the MC_MoveVelocityCSV instruction.	The axis stops due to exception of the MC_MoveVelocityCSV instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9606	The axis stops due to exception of the MC_MoveBuffer instruction.	The axis stops due to exception of the MC_MoveBuffer instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9607	The axis stops due to parameter exception of the MC_MoveFeed instruction.	The axis stops due to parameter exception of the MC_MoveFeed instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9608	The axis stops due to parameter exception of the MC_Stop instruction.	The axis stops due to parameter exception of the MC_Stop instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9609	The axis stops due to parameter exception of the MC_Torque instruction.	The axis stops due to parameter exception of the MC_Torque instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9610	The axis stops due to parameter exception of the MC_Halt instruction.	The axis stops due to parameter exception of the MC_Halt instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9611	The axis stops due to parameter exception of the MC_MoveSuperImposed instruction.	The axis stops due to parameter exception of the MC_MoveSuperImposed instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9612	The axis stops due to an error reported by the	The axis stops due to an error reported by the MC_SyncMoveVelocity instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.

Error code	Error message	Detailed description	Solution
	MC_SyncMoveVelocity instruction.		
9613	The axis stops due to an error reported by the MC_SyncTorqueControl instruction.	The axis stops due to an error reported by the MC_SyncTorqueControl instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9614	The axis stops due to an error reported by the MC_FollowVelocity instruction.	The axis stops due to an error reported by the MC_FollowVelocity instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9615	The axis stops due to parameter exception of the MC_SetOverRide instruction.	The axis stops due to parameter exception of the MC_SetOverRide instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9701	The encoder axis instruction failed to request the memory.	The encoder axis instruction failed to request the memory.	1. Check whether the PLC memory runs out. 2. Contact the manufacturer.
9702	1. The encoder axis type is incorrect. 2. The requested encoder axis does not exist. 3. The instruction is not supported in offline commissioning.	1. The encoder axis type is incorrect. 2. The requested encoder axis does not exist. 3. The instruction is not supported in offline commissioning.	This instruction does not support the set axis type. Check whether the axis type setting is incorrect.
9703	Failed to configure the axis.	Failed to configure the axis.	Check whether the board software and the software tool match.
9704	Counter operation command is not configured in I/O mapping of the encoder axis.	Counter operation command is not configured in I/O mapping of the encoder axis.	Configure Counter operation command in I/O mapping of the encoder axis.
9705	Counter status is not configured in I/	Counter status is not configured in I/O mapping of the encoder axis.	Configure Counter status in I/O mapping of the encoder axis.

Error code	Error message	Detailed description	Solution
	O mapping of the encoder axis.		
9706	Encoder present position is not configured in I/O mapping of the encoder axis.	Encoder present position is not configured in I/O mapping of the encoder axis.	Configure Encoder present position in I/O mapping of the encoder axis.
9707	Pulse rate is not configured in I/O mapping of the encoder axis.	Pulse rate is not configured in I/O mapping of the encoder axis.	Configure Pulse rate in I/O mapping of the encoder axis.
9708	The positive limit of the encoder axis is not greater than the negative limit.	The positive limit of the encoder axis is not greater than the negative limit.	Ensure that the positive limit of the encoder axis is greater than the negative limit.
9709	The positive limit of the encoder axis is greater than 2147483647 after being converted into the pulse unit.	The positive limit of the encoder axis is greater than 2147483647 after being converted into the pulse unit.	Ensure that the positive limit of the encoder axis is less than or equal to 2147483647 after being converted into the pulse unit.
9710	The negative limit of the encoder axis is less than -2147483648 after being converted into the pulse unit.	The negative limit of the encoder axis is less than -2147483648 after being converted into the pulse unit.	Ensure that the negative limit of the encoder axis is greater than or equal to -2147483648 after being converted into the pulse unit.
9711	The revolution cycle of the encoder axis in ring mode is greater than 2147483647 after being converted into the pulse unit.	The revolution cycle of the encoder axis in ring mode is greater than 2147483647 after being converted into the pulse unit.	Ensure that the revolution cycle of the encoder axis in ring mode is less than or equal to 2147483647 after being converted into the pulse unit.
9712	The encoder axis is changed while the ENC_Counter instruction is still active.	The encoder axis is changed while the ENC_Counter instruction is still active.	Do not change the encoder axis while the ENC_Counter instruction is still active.
9713		GR10-2HCE module fault	

Error code	Error message	Detailed description	Solution
	GR10-2HCE module fault		Check the fault code object dictionary of the GR10-2HCE module and troubleshoot the fault according to the fault code.
9714	Failed to reset the encoder axis fault.	Failed to reset the encoder axis fault.	1. The current fault of the encoder axis does not support reset. 2. The encoder shaft enters the faulty state immediately after the fault is reset. Check the axis fault codes and slave fault codes to further determine the fault type.
9715	The ENC_Reset instruction is called when the encoder axis is not faulty.	The ENC_Reset instruction is called when the encoder axis is not faulty.	Do not call the ENC_Reset instruction when the encoder axis is not faulty.
9716	The value of TriggerMode of the ENC_Preset instruction is out of range.	The value of TriggerMode of the ENC_Preset instruction is out of range.	Ensure that the parameter value is within the allowable range.
9717	The Position of the ENC_Preset instruction is set to a value greater than 9999999.	The Position of the ENC_Preset instruction is set to a value greater than 9999999.	Set Position of the ENC_Preset instruction to a value less than or equal to 9999999.
9718	Physical output command is not configured in I/O mapping of the encoder axis.	Physical output command is not configured in I/O mapping of the encoder axis.	Configure Physical output command in I/O mapping of the encoder axis.
9719	The preset position or comparison output position of the encoder axis instruction is greater than the positive limit.	The preset position or comparison output position of the encoder axis instruction is greater than the positive limit.	Ensure that the preset position or comparison output position of the encoder axis instruction is less than or equal to the positive limit.
9720	The preset position or comparison output position of	The preset position or comparison output position of the encoder axis	Ensure that the preset position or comparison output position of the

Error code	Error message	Detailed description	Solution
	the encoder axis instruction is less than the negative limit.	instruction is less than the negative limit.	encoder axis instruction is greater than the negative limit.
9721	The preset position or comparison output position of the encoder axis instruction is greater than 2147483647 or less than -2147483648 after being converted into the pulse unit.	The preset position or comparison output position of the encoder axis instruction is greater than 2147483647 or less than -2147483648 after being converted into the pulse unit.	Ensure that the preset position or comparison output position of the encoder axis instruction is between -2147483648 and +2147483647 after being converted into the pulse unit.
9722	The preset position or comparison output position of the encoder axis instruction is greater than or equal to the revolution cycle in ring mode.	The preset position or comparison output position of the encoder axis instruction is greater than or equal to the revolution cycle in ring mode.	Ensure that the preset position or comparison output position of the encoder axis instruction is less than the revolution cycle in ring mode.
9723	The value of ProbeID of the ENC_TouchProbe instruction is out of range.	The value of ProbeID of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9724	The value of TriggerEdge of the ENC_TouchProbe instruction is out of range.	The value of TriggerEdge of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9725	The value of TerminalSource of the ENC_TouchProbe instruction is out of range.	The value of TerminalSource of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9726			

Error code	Error message	Detailed description	Solution
	The value of TriggerMode of the ENC_TouchProbe instruction is out of range.	The value of TriggerMode of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9727	The probe status word is not associated in I/O mapping of the encoder axis.	The probe status word is not associated in I/O mapping of the encoder axis.	Ensure that the probe status word is associated in I/O mapping of the encoder axis.
9728	The probe feedback position is not associated in I/O mapping of the encoder axis.	The probe feedback position is not associated in I/O mapping of the encoder axis.	Ensure that the probe feedback position is associated in I/O mapping of the encoder axis.
9729	The control word is not associated in I/O mapping of the encoder axis.	The control word is not associated in I/O mapping of the encoder axis.	Ensure that the control word is associated in I/O mapping of the encoder axis.
9730	The probe window function of the encoder axis is enabled, but the start position of the window is not less than the end position.	The probe window function of the encoder axis is enabled, but the start position of the window is not less than the end position.	Ensure that the start position of the probe window is less than the end position.
9731	The Xn0 terminal is not assigned with the probe function.	The Xn0 terminal is not assigned with the probe function.	Assign the probe function to the Xn0 terminal.
9732	The Xn1 terminal is not assigned with the probe function.	The Xn1 terminal is not assigned with the probe function.	Assign the probe function to the Xn1 terminal.
9742	Compare mode is not configured in I/O mapping of the encoder axis.	Compare mode is not configured in I/O mapping of the encoder axis.	Configure Compare mode in I/O mapping of the encoder axis.
9743	Compare pulse/time is not configured in I/O	Compare pulse/time is not configured in I/O mapping of the encoder axis.	Configure Compare pulse/time in I/O mapping of the encoder axis.

Error code	Error message	Detailed description	Solution
	mapping of the encoder axis.		
9744	Compare size/step is not configured in I/O mapping of the encoder axis.	Compare size/step is not configured in I/O mapping of the encoder axis.	Configure Compare size/step in I/O mapping of the encoder axis.
9745	Compare point value 1 is not configured in I/O mapping of the encoder axis.	Compare point value 1 is not configured in I/O mapping of the encoder axis.	Configure Compare point value 1 in I/O mapping of the encoder axis.
9746	Compare point value 2 is not configured in I/O mapping of the encoder axis.	Compare point value 2 is not configured in I/O mapping of the encoder axis.	Configure Compare point value 2 in I/O mapping of the encoder axis.
9747	Physical output status is not configured in I/O mapping of the encoder axis.	Physical output status is not configured in I/O mapping of the encoder axis.	Configure Physical output status in I/O mapping of the encoder axis.
9748	Compare error code is not configured in I/O mapping of the encoder axis.	Compare error code is not configured in I/O mapping of the encoder axis.	Configure Compare error code in I/O mapping of the encoder axis.
9749	Current compare number/position is not configured in I/O mapping of the encoder axis.	Current compare number/position is not configured in I/O mapping of the encoder axis.	Configure Current compare number/position in I/O mapping of the encoder axis.
9750	Failed to obtain the start address of the array of the single-axis array comparison output instruction.	Failed to obtain the start address of the array of the single-axis array comparison output instruction.	1. Check whether the PLC memory is sufficient. 2. Check whether the software tool and board software match. 3. Check whether the array of the instruction is out of bounds.
9751	Failed to obtain the start address of the axis group of the	Failed to obtain the start address of the axis group of the axis group	1. Check whether the PLC memory is sufficient.

Error code	Error message	Detailed description	Solution
	axis group array comparison output instruction.	array comparison output instruction.	2. Check whether the software tool and board software match. 3. Check whether the array of the instruction is out of bounds.
9752	The bus encoder axis is not associated with any slave.	The bus encoder axis is not associated with any slave.	Associate the bus encoder axis with a slave.
9753	The x-axis and y-axis of the axis group array comparison instruction are not associated with the same slave.	The x-axis and y-axis of the axis group array comparison instruction are not associated with the same slave.	Associate the x-axis and y-axis of the axis group comparison output instruction with the same slave.
9754	The x-axis of the axis group array comparison instruction is not associated with the first channel of the slave.	The x-axis of the axis group array comparison instruction is not associated with the first channel of the slave.	Associate the x-axis of the axis group comparison output instruction with the first channel of the slave.
9755	The y-axis of the axis group array comparison instruction is not associated with the second channel of the slave.	The y-axis of the axis group array comparison instruction is not associated with the second channel of the slave.	Associate the y-axis of the axis group comparison output instruction with the second channel of the slave.
9756	The Yn0 terminal is not assigned with the one-dimensional comparison output function.	The Yn0 terminal is not assigned with the one-dimensional comparison output function.	Assign the one-dimensional comparison output function to the Yn0 output terminal corresponding to the channel.
9757	The absolute value of the start value of the encoder axis step comparison output instruction is	The absolute value of the start value of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.

Error code	Error message	Detailed description	Solution
	greater than 9999999.		
9758	The absolute value of the end value of the encoder axis step comparison output instruction is greater than 9999999.	The absolute value of the end value of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9759	The absolute value of the step of the encoder axis step comparison output instruction is greater than 9999999.	The absolute value of the step of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9760	The absolute value of Parameter of the encoder axis step comparison output instruction is greater than 9999999.	The absolute value of Parameter of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9761	The value of Mode of the encoder axis comparison output instruction is out of range.	The value of Mode of the encoder axis comparison output instruction is out of range.	Ensure that the parameter value is within the allowable range.
9762	The time for time control of the encoder axis comparison output is out of range.	The time for time control of the encoder axis comparison output is out of range.	Ensure that the parameter value is within the allowable range.
9763	The step of the encoder axis step comparison output instruction is 0.	The step of the encoder axis step comparison output instruction is 0.	Set the step of the step comparison output instruction to a value other than 0.

Error code	Error message	Detailed description	Solution
9764	The start position of the step comparison output instruction of the encoder axis is equal to the end position.	The start position of the step comparison output instruction of the encoder axis is equal to the end position.	Ensure that the start position of the step comparison output instruction is not equal to the end position.
9765	The start position of the step comparison output instruction of the encoder axis is less than the end position, but the step is negative.	The start position of the step comparison output instruction of the encoder axis is less than the end position, but the step is negative.	Set the step to a positive value.
9766	The start position of the step comparison output instruction of the encoder axis is greater than the end position, but the step is positive.	The start position of the step comparison output instruction of the encoder axis is greater than the end position, but the step is positive.	Set the step to a negative value.
9767	The value of Size of the encoder axis array comparison output instruction is out of range.	The value of Size of the encoder axis array comparison output instruction is out of range.	Ensure that the parameter value is within the allowable range.
9768	The absolute value of the target position of the encoder axis array comparison output instruction is greater than 9999999.	The absolute value of the target position of the encoder axis array comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9769	The axis is performing one-dimensional comparison output and must not be aborted by a two-	The axis is performing one-dimensional comparison output and must not be aborted by a two-dimensional comparison output instruction.	Wait for the one-dimensional comparison output to complete or stop the one-dimensional comparison output before executing the two-dimensional comparison output instruction.

Error code	Error message	Detailed description	Solution
	dimensional comparison output instruction.		
9770	The EtherCAT slave is disconnected during operation.	The EtherCAT slave is disconnected during operation.	Check whether the EtherCAT slave is disconnected during operation.
9771	The bus encoder axis is in offline commissioning mode.	The bus encoder axis is in offline commissioning mode.	The bus encoder axis does not support the offline commissioning mode.
9772	The DI terminal is not assigned with the preset position function.	The DI terminal is not assigned with the preset position function.	Assign the preset position function to the DI terminal before calling the preset position instruction.
9773	The value of Parameter in the comparison instruction is out of range when the pulse output mode is selected.	The value of Parameter in the comparison instruction is out of range when the pulse output mode is selected.	Do not set Parameter to 0 or a negative value when the pulse output mode is selected in the comparison instruction.
9774	The 2HCE module fails when the comparison output instruction is called.	The 2HCE module fails when the comparison output instruction is called.	1. Ensure that the input parameters are within the allowable range. 2. Check whether I/O mapping of the encoder axis is manually modified and whether it meets the I/O mapping configuration requirements of the comparison output instruction.
9775	The set position in ring mode is less than 0.	The set position in ring mode is less than 0.	Set the position in ring mode to a value greater than or equal to 0.
9776	The Y00 terminal is not assigned with the two-dimensional comparison output function.	The Y00 terminal is not assigned with the two-dimensional comparison output function.	Assign the two-dimensional comparison output function to the Y00 output terminal corresponding to the channel.
9777	The axis is performing two-dimensional	The axis is performing two-dimensional comparison output and cannot be aborted by a one-	Wait for the two-dimensional comparison output to complete, or stop the two-dimensional comparison

Error code	Error message	Detailed description	Solution
	comparison output and cannot be aborted by a one-dimensional comparison output instruction.	dimensional comparison output instruction.	output before calling the one-dimensional comparison output instruction.
9800	Failed to read the number of motion control axes.	Failed to read the number of motion control axes.	Change the software tool version.
9801	The number of motion control axes is out of range.	The number of motion control axes is out of range.	Reduce the number of axes because the H5U supports at most 32 axes.
9802	The axis failed to request internal storage space.	The axis failed to request internal storage space.	1. Check whether the memory runs out. 2. Contact the manufacturer.
9803	Failed to obtain axis parameters.	Failed to obtain axis parameters.	Change the software tool version.
9804	Failed to obtain the slave.	Failed to obtain the slave.	Change the software tool version.
9805	Failed to obtain the system variable.	Failed to obtain the system variable.	1. Check whether the memory runs out. 2. Return the machine to the manufacturer for analysis.
9806	Parameters related to the gear ratio are set improperly.	Parameters related to the gear ratio are set improperly.	1. Ensure that the numerator and denominator of the gear ratio are greater than 0. 2. Ensure that the number of pulses per revolution of the motor/encoder is greater than 0. 3. Ensure that the displacement per revolution of the rotary table is between 0.000001 and 9999999.
9807	The software limiting parameters are set improperly.	The software limiting parameters are set improperly.	1. Ensure that the positive limit is not greater than 9999999. 2. Ensure that the negative limit is not greater than 9999999. 3. Ensure that the negative limit is not greater than the positive limit.

Error code	Error message	Detailed description	Solution
9808	The linear/rotary mode parameter is set improperly.	The linear/rotary mode parameter is set improperly.	Note that only the linear mode and rotary mode are supported.
9809	The revolution cycle is set improperly.	The revolution cycle is set improperly.	Ensure that the revolution cycle is between 0.01 and 9999999.
9810	The encoder mode is set improperly.	The encoder mode is set improperly.	Ensure that the encoder mode is set properly. Note that only the incremental mode and absolute value mode are supported.
9811	The homing parameter is set improperly.	The homing parameter is set improperly.	1. Do not modify the homing mode of the bus servo axis. If you want to modify the homing mode of the bus servo axis, write to the SDO. 2. Check whether the homing mode is set properly. Note that only the values 17 to 30 and 35 are supported.
9812	The Modbus address setting of the limit, home, or probe terminal is out of range.	The Modbus address setting of the limit, home, or probe terminal is out of range.	1. Check whether the set address is out of the range of Modbus addresses. 2. Select an address among X0 to X7 for the home signal. 3. Select an address among X0 to X7 for the probe signal.
9813	The pulse output mode of the local pulse axis is set improperly.	The pulse output mode of the local pulse axis is set improperly.	Check whether the pulse output mode of the local pulse axis is set improperly.
9814	The limiting deceleration is set improperly.	The limiting deceleration is set improperly.	Ensure that the limiting deceleration is between 0.0001 and 9999999.
9815	The deceleration upon axis fault is set improperly.	The deceleration upon axis fault is set improperly.	Ensure that the maximum velocity is between 0.0001 and 999999.
9816	The maximum velocity is set improperly.	The maximum velocity is set improperly.	Ensure that the maximum velocity is between 0.0001 and 999999.

Error code	Error message	Detailed description	Solution
9817	The maximum positive torque is set improperly.	The maximum positive torque is set improperly.	Ensure that the maximum positive torque is between 1 and 65534.
9818	The maximum negative torque is set improperly.	The maximum negative torque is set improperly.	Ensure that the maximum negative torque is between 1 and 65534.
9819	The maximum jogging velocity is set improperly.	The maximum jogging velocity is set improperly.	Ensure that the maximum jogging velocity is between 0.0001 and the maximum velocity.
9820	The maximum acceleration is set improperly.	The maximum acceleration is set improperly.	Ensure that the maximum acceleration is between 0.0001 and 9999999.
9821	The following error threshold is set improperly.	The following error threshold is set improperly.	Ensure that the following error threshold is between 0.0001 and 9999999.
9822	The velocity reach threshold is set improperly.	The velocity reach threshold is set improperly.	Ensure that the velocity reach threshold is between 0.0001 and 9999999.
9823	The homing velocity is set improperly.	The homing velocity is set improperly.	1. Ensure that the homing velocity is between 0.0001 and 9999999. 2. Ensure that the homing velocity is not greater than the maximum velocity. 3. Ensure that the value obtained by multiplying the homing velocity by the gear ratio is between 1 and 2148483647.

Error code	Error message	Detailed description	Solution
9824	The homing approach velocity is set improperly.	The homing approach velocity is set improperly.	1. Ensure that the homing approach velocity is between 0.0001 and 9999999. 2. Ensure that the homing approach velocity is not greater than the maximum velocity. 3. Ensure that the value obtained by multiplying the homing approach velocity by the gear ratio is between 1 and 2148483647. 4. Ensure that the homing approach velocity is less than the homing velocity.
9825	The homing position mode setting is out of range.	The homing position mode setting is out of range.	Ensure that the parameter value is within the allowable range.
9826	The homing acceleration setting is improper.	The homing acceleration setting is improper.	1. Ensure that the homing acceleration is between 0.0001 and 9999999. 2. Ensure that the homing acceleration is not greater than the maximum acceleration.
9827	The homing timeout time is out of range.	The homing timeout time is out of range.	Ensure that the homing timeout time is greater than or equal to 1.

Service and Support

Should you encounter a safety accident during the use or operation of the product, or face challenges in operating and maintaining the equipment, which remain unresolved after the relevant documentation is consulted, we provide multiple channels to ensure prompt resolution:

- Channel #1: Contact service@inovance.com.
- Channel #2: Visit https://www.inovance.com/global to access document downloads, after-sales support, spare parts ordering, repair applications, and authenticity verification services.
- Channel #3: Download My Inovance app (https://zshc-eu.inovance.com/download-pc/) where you can access products info and documentation, and query product parameters.

We are committed to providing you with quick and professional technical support, and we look forward to your satisfaction and trust.

19012250A20

Copyright © Shenzhen Inovance Technology Co., Ltd.

Shenzhen Inovance Technology Co., Ltd.

www.inovance.com

Add.: Inovance Headquarters Tower, High-tech Industrial Park,
Guanlan Street, Longhua New District,
Shenzhen 518000, P.R. China

Tel: (0755) 2979 9595 Fax: (0755) 2961 9897

Suzhou Inovance Technology Co., Ltd.

www.inovance.com

Add.: No. 52, Tian E Dang Road, Wuzhong District, 215104,
Suzhou City, Jiangsu Province, P.R. China

Tel: (0512) 6637 6666 Fax: (0512) 6285 6720