VC Series Micro-PLC Programming Manual

Version: V1.1
Date: 2021-05-15

Suzhou Veichi Electric Co., Ltd. provides customers with a full range of technical support services. You can contact the local office or customer service center of our company, or directly contact our headquarters.

Suzhou Veichi Electric Co., Ltd.

All rights reserved.The content of the manualis subject to change without prior notice.

Suzhou Veichi Electric Co., Ltd.

Address: No.1000 of Songjia Road, Guoxiang Street, Wuzhong Economic and Technological Development Zone, Suzhou

Hotline: +86-0755-29685680

Website: www.veichi.com

Preface

- Intended readers

  This manual is intended for automation technicians to help them master the programming, system design, and commissioning of VEICHI programmable logic controllers (PLCs), providing reference for both new and senior learners of PLC programming.

- Content

  This manual describes the programming principles, software and hardware programming resources, supported programming languages, and detailed instructions of VC series PLCs; provides some technical reference, such as high-speed input/output and communication information; and introduces the application methods of all functions with application instances provided.

- Arrangement

  The chapters in this manual are organized from the whole to details. The content of each chapter is independent from that of another. You can read the manual through to gradually understand VC series PLCs to the full extent, or you can just read some chapters at any time to get some technical reference.

- Reading guide

  1. For readers unfamiliar with PLCs

     If you have never used PLCs before, it is recommended that you read chapters 1 to 4 first. These chapters explain the basics of PLCs, including function descriptions, programming languages, program elements, data types, addressing modes, soft element definitions, program annotation function and programming, application of main program and subprograms.

  2. For readers familiar with PLCs

     If you are familiar with the basic concepts and programming tools of PLCs, you can directly read Chapter 5 "Basic instructions" and Chapter 6 "Application instructions". These two chapters describe all the instructions of VEICHI VC series PLCs. To understand how to use sequential function diagrams, high-speed I/O, interruption, and communication functions, refer to chapters 7 to 10. To understand the functions of positioning control, refer to Chapter 11 "Positioning function guide". In addition, Appendix K "Instruction order index table" and Appendix L "Instruction classification index table" allow you to look up descriptions of corresponding instructions by instruction type and alphabetic ordering of the English names of the instructions, respectively, which makes the reading easier.

- Related documents and references

  You can also refer to the following manuals:

  - VC1 Series PLC User Manual

Contents

Preface 1

Chapter 1 Product overview........ 1

1.1 Product introduction........1
1.1.1 Product performance and specifications 1
1.1.2 Appearance of VC1, and VC1 series main modules 5
1.2 Auto Studio programming software 5
1.2.1 Basic configuration........6
1.2.2 Installation process of the Auto Studio programming software........6
1.2.3 Running interface of Auto Studio........6
1.2.4 Programming cable 7
1.3 Communication function........7
1.3.1 Modbus communication network........7
1.3.2 N:N communication network 7
1.3.3 Free-port network........8

Chapter 2 Function Description 11

2.1 Programming resources and principles 12
2.1.1 Programming resources........12
2.1.2 PLC operating mechanism (Scan cycle model)........14
2.1.3 User program running watchdog........15
2.1.4 Constant scan operation mode 15
2.1.5 User file download and storage........15
2.1.6 Element initialization 15
2.1.7 Saving data at power outage........15
2.1.8 Digital filtering for input points 16
2.1.9 No battery mode........16
2.1.10 User program protection........16
2.2 System configuration........17
2.2.1 System block........17
2.2.2 Data blocks 23
2.2.3 Global variable table 24
2.3 Running mode and state control........24
2.3.1 Concepts of system run and stop states 24
2.3.2 Run and stop state 24
2.3.3 Setting the state of output pointsin the stop state........25
2.4 System commissioning........25
2.4.1 Program download and upload 25
2.4.2 Error reporting mechanism........26
2.4.3 Modifying the user program online 28
2.4.4 Clearing and formatting........28
2.4.5 Checking PLC information online 29
2.4.6 Element value writing and forcing, and element monitoring table 29
2.4.7 Generating data blocks from RAM 32

Chapter 3 Soft element and data........34

3.1 Type and function of soft elements........35
3.1.1 Summary of soft elements........35
3.1.2 Soft elementlist........35

3.1.3 Input and output points........37
3.1.4 Auxiliary Relays........37
3.1.5 State relay........38
3.1.6 Timer........38
3.1.7 Counter........39
3.1.8 Data register........39
3.1.9 Special auxiliary relay........40
3.1.10 Special data register........40
3.1.11 Indexing register........41
3.1.12 Local auxiliary relay........41
3.1.13 Local data register........41
3.1.14 Bit-string combined addressing mode (Kn addressing mode)........41
3.1.15 Indexing mode (Z addressing mode)........42
3.1.16 Bit-string combined indexing mode........43
3.1.17 Storing and addressing 32-bit data in D, R, and V elements........44
3.2 Data........44
3.2.1 Data type........44
3.2.2 Correlation between elements and data types........44
3.2.3 Constant........45
Chapter 4 Programming concept........46
4.1 Introduction to programming languages........47
4.1.1 LAD........47
4.1.2 IL........48
4.1.3 SFC........48
4.2 Program Elements........49
4.2.1 User Program........49
4.2.2 System block........50
4.2.3 Data block........50
4.3 Block comment and variable comment of the program........50
4.3.1 Block comment........50
4.3.2 Variable Comment........51
4.4 Subprogram........52
4.4.1 Concept........52
4.4.2 Notestouse SBRs........52
4.4.3 Definition of the SBR variable table........53
4.4.4 SBR parameter transfer........54
4.4.5 SBR application instance........54
4.5 General information of the instructions........56
4.5.1 Instruction operands........56
4.5.2 Flag bit........56
4.5.3 Restrictions on the use of instructions........56
Chapter 5 Basic instructions........57
5.1 Contact logic instructions........59
5.1.1 LD: NO contact instruction........59
5.1.2 LDI: NC contact instruction........59
5.1.3 AND: NO contact and instruction........59
5.1.4 ANI: NC contact and instruction........60
5.1.5 OR: NO contact or instruction........60
5.1.6 ORI: NO contact or instruction........61

5.1.7 OUT: Coil output instruction 61
5.1.8 ANB: Energy flow block and instruction 61
5.1.9 ORB: Energy flow block or instruction 62
5.1.10 MPS: Output energy flow push instruction 62
5.1.11 MRD: Instruction for reading output energy flow stack top value 63
5.1.12 MPP: Output energy flow stack pop instruction 63
5.1.13 EU: Rising edge detection instruction 64
5.1.14 ED: Falling edge detection instruction 64
5.1.15 LDP: Rising edge of contact 65
5.1.16 LDF: Falling edge of contact 65
5.1.17 ANDP: Rising edge of contact 66
5.1.18 ANDF: Falling edge of contact 66
5.1.19 ORP: Rising edge of contact 67
5.1.20 ORF: Falling edge of contact 67
5.1.21 PLP: Rising edge output instruction 68
5.1.22 PLF: Falling edge output instruction 68
5.1.23 INV: Energy flow negation instruction 69
5.1.24 SET: Coil set instruction 69
5.1.25 RST: Coil reset instruction 69
5.1.26 NOP:No operation instruction 70
5.2 Main control instructions 70
5.2.1 MC: Main control instruction 70
5.2.2 MCR: Main control reset instruction 70
5.3 SFC instructions 71
5.3.1 STL: SFC state loadinginstruction 71
5.3.2 SET Sxx: SFC state transition instruction 72
5.3.3 OUT Sxx: SFC state jumpinstruction 72
5.3.4 RST Sxx: SFC state reset instruction 72
5.3.5 RET: SFC program segment end instruction 73
5.4 Timer instructions 73
5.4.1 TON: Turn-on delay timing instruction 73
5.4.2 TONR: Memory-type turn-on delay timing instruction 73
5.4.3 TOF: Turn-off delay timing instruction 74
5.4.4 TMON: Non-retriggerable monostable timing instruction 75
5.5 Counter instructions 76
5.5.1 CTU: 16-bit increment counterinstruction 76
5.5.2 CTR: 16-bit cycliccountinginstruction 76
5.5.3 DCNT:32-bit increment and decrement counting instruction 77

Chapter 6 Application instructions 79

6.1 Program flow control instructions 85
6.1.1 FOR: Cycle instruction 85
6.1.2 NEXT: Cycle return instruction 85
6.1.3 LBL: Jump label definition instruction 86
6.1.4 CJ: Conditional jump instruction 88
6.1.5 CFEND: Instruction for conditional return of user main program 88
6.1.6 WDT: Instruction for watchdog reset of user program 89
6.1.7 EI: Enable interrupt instruction 89
6.1.8 DI: Disable interrupt instruction 89
6.1.9 CIRET: Instruction for conditional return of user interrupt program 89
6.1.10 STOP: Instruction for stopping the user program 89

6.1.11 CALL: Instruction for calling the user subprogram 90
6.1.12 CSRET: Instruction for conditional return of user subprogram 90
6.2 Data transmission instructions 91
6.2.1 MOV: Word data transmissioninstruction 91
6.2.2 DMOV: Double word data transmission instruction 91
6.2.3 RMOV: Floating-point data transmission instruction 92
6.2.4 BMOV: Block data transmission instruction 92
6.2.5 FMOV: Data block fill instruction 92
6.2.6 DFMOV: Data block double word fill instruction 93
6.2.7 SWAP: MSB/LSB swop instruction 93
6.2.8 XCH: Word swop instruction 94
6.2.9 DXCH: Double word swop instruction 94
6.2.10 PUSH: Data push instruction 94
6.2.11 FIFO: First-in-first-out instruction 95
6.2.12 LIFO: Last-in-first-out instruction 96
6.2.13 WSFR: Word string shift right instruction 96
6.2.14 WSFL: Word string shift left instruction 97
6.3 Integer arithmetic operation instructions 98
6.3.1 ADD: Integer addition instruction 98
6.3.2 SUB: Integer subtraction instruction 99
6.3.3 MUL: Integer multiplication instruction 99
6.3.4 DIV: Integer division instruction 99
6.3.5 SQT: Instruction for extracting the square root of an integer 100
6.3.6 INC: Integer plus one instruction 100
6.3.7 DEC: Integer minus one instruction 101
6.3.8 VABS: Instruction for obtaining the absolute value of an integer 101
6.3.9 NEG: Integer negation instruction 102
6.3.10 DADD: Long integer addition instruction 102
6.3.11 DSUB: Long integer subtraction instruction 102
6.3.12 DMUL: Long integer multiplication instruction 103
6.3.13 DDIV: Long integer division instruction 103
6.3.14 DSQT: Instruction for extracting the square root of a long integer 104
6.3.15 DINC: Long integer plus one instruction 104
6.3.16 DDEC: Long integer minus one instruction 105
6.3.17 DVABS: Instruction for obtaining the absolute value of a long integer 105
6.3.18 DNEG: Long integer negation instruction 106
6.3.19 SUM: Integeraccumulation instruction 106
6.3.20 DSUM: Long integer accumulation instruction 108
6.4 Floating-point arithmetic operation instructions 108
6.4.1 RADD: Floating-point numberaddition instruction 108
6.4.2 RSUB: Floating-point numbersubtraction instruction 109
6.4.3 RMUL: Floating-point numbermultiplication instruction 109
6.4.4 RDIV: Floating-point numberdivision instruction 110
6.4.5 RSQT: Instruction for extracting the square root of a floating-point number 110
6.4.6 RVABS: Instruction for obtaining the absolute value of a floating-point number 111
6.4.7 RNEG: Floating-point number negation instruction 111
6.4.8 SIN: Instruction for obtaining SIN of a floating-point number 111
6.4.9 COS: Instruction for obtaining COS of a floating-point number 112
6.4.10 TAN: Instruction for obtaining TAN of a floating-point number 112
6.4.11 POWER: Instruction for exponentiation of a floating-point number 113

6.4.12 LN: Instruction for obtaining the natural logarithm of a floating-point number 113
6.4.13 EXP: Instruction for obtaining the natural number power of a floating-point number...........................114
6.4.14 RSUM: Floating-point number accumulation instruction 114
6.4.15 ASIN: Instruction for obtaining ASIN of a floating-point number..115
6.4.16 ACOS: Instruction for obtaining ACOS of a floating-point number 115
6.4.17 ATAN: Instruction for obtaining ATAN of a floating-point number 115
6.4.18 LOG: Instruction for obtaining the common logarithm of a floating-point number................................116
6.4.19 RAD: Instruction for floating-point number angle-radian conversion 116
6.4.20 DEG: Instruction for floating-point number radian-angle conversion 116
6.5 Value conversion instructions...117
6.5.1 DTI: Instruction for converting a long integer to an integer..117
6.5.2 ITD: Instruction for converting an integer to a long integer..117
6.5.3 FLT: Instruction for converting an integer to a floating-point number 118
6.5.4 DFLT: Instruction for converting a long integer to a floating-point number..118
6.5.5 INT: Instruction for converting a floating-point number to an integer...118
6.5.6 DINT: Instruction for convert a floating-point number to a long integer 119
6.5.7 BCD: Instruction for converting a word to a 16-bit BCD code...119
6.5.8 DBCD: Instruction for converting a double word to a 32-bit BCD code 120
6.5.9 BIN: Instruction for converting a 16-bit BCD code to a word 120
6.5.10 DBIN: Instruction for converting a 32-bit BCD code to a double word...121
6.5.11 GRY: Instruction for converting a word to a 16-bit gray code..121
6.5.12 DGRY: Instruction for converting a double word to a 32-bit gray code..121
6.5.13 GBIN: Instruction for converting a 16-bit gray code to a word 122
6.5.14 DGBIN: Instruction for converting a 32-bit gray code to a double word...122
6.5.15 SEG: Instruction for converting a word to a 7-segment code 123
6.5.16 ASC: ASCII code conversion instruction 123
6.5.17 ITA: Instruction for convertinga 16-bit hex data to an ASCII code...124
6.5.18 ATI: Instruction for converting an ASCII code to a 16-bit hex data...124
6.5.19 LCNV: Project conversion instruction...125
6.5.20 RLCNV: Floating-point project conversion instruction 126
6.6 Word logic operation instructions...127
6.6.1 WAND: Word AND instruction...127
6.6.2 WOR: INT OR instruction..129
6.6.3 WXOR: Word XOR instruction 129
6.6.4 WINV: Word INV instruction..130
6.6.5 DWAND: Double word AND instruction...130
6.6.6 DWOR: Double word OR instruction 130
6.6.7 DWXOR: Double word XOR instruction 131
6.6.8 DWINV: Double word negation instruction 131
6.7 Bit shift and rotate instructions 132
6.7.1 ROR: 16-bit rotate right instruction..132
6.7.2 ROL: 16-bit rotate left instruction..132
6.7.3 RCR: Instruction for 16-bit rotate right with carry flag bit...133
6.7.4 RCL: Instruction for 16-bit rotate left with carry flag bit..134
6.7.5 DROR: 32-bit rotate right instruction 134
6.7.6 DROL: 32-bit rotate left instruction 135
6.7.7 DRCR: Instruction for 32-bit rotate right with carry flag bit 135
6.7.8 DRCL: Instruction for 32-bit rotate left with carry flag bit 136
6.7.9 SHR: 16-bit shift right instruction...136
6.7.10 SHL:16-bit shift left instruction...137

6.7.11 DSHR: 32-bit shift right instruction 138
6.7.12 DSHL: 32-bit shift left instruction 138
6.7.13 SFTR: Bit string shift right instruction 139
6.7.14 SFTL: Bit string shift left instruction 139
6.8 Peripheral instructions 141
6.8.1 FROM: Instruction for reading words from a special module buffer register........ 141
6.8.2 DFROM: Instruction for reading double words from a special module buffer register 141
6.8.3 TO: Instruction for writing words from a special module buffer register........ 142
6.8.4 DTO: Instruction for writing double words from a special module buffer register 143
6.8.5 VRRD: Instruction for readingthe value of an analog potentiometer 143
6.8.6 REFF: Instruction for setting input filtering constant........ 144
6.8.7 REF: Instruction for immediately refreshing I/O........ 144
6.8.8 EROMWR: EEPROM write instruction 145
6.8.9 PR: Printing instruction........ 145
6.8.10 TKY: Numeric key input instruction 146
6.9 Real-time clock instructions........ 148
6.9.1 TRD: Real-time clock read instruction........ 148
6.9.2 TWR: Real-time clock write instruction........ 149
6.9.3 TADD: Clock addition instruction........ 150
6.9.4 TSUB: Clock subtraction instruction........ 151
6.9.5 HOUR: Chronograph instruction 152
6.9.6 DCMP: (=、<、>、<>、>=、<=)Date comparison instruction 153
6.9.7 TCMP: (=、<、>、<>、>=、<=)Time comparison instruction 154
6.9.8 HTOS: Instruction for converting hour-minute-second data to seconds 155
6.9.9 STOH: Instruction for converting seconds to hour-minute-second data 155
6.10 High-speed I/O instructions 156
6.10.1 HCNT: Instruction for driving the high-speed counter 156
6.10.2 DHSCS: Instruction of setting the high-speed count comparison........ 156
6.10.3 DHSCI: Instruction for triggering interrupt based on comparison of high-speed count........ 157
6.10.4 DHSPI:Triggering interrupt Instruction based on comparison of absolute high-speed output positions159
6.10.5 DHSCR: Instruction for resetting the high-speed count comparison 160
6.10.6 DHSZ: High-speed count range comparison instruction 161
6.10.7 DHST: High-speed count table comparison instruction........ 162
6.10.8 DHSP: Instruction for pulse output based on high-speed count table comparison........ 164
6.10.9 SPD: Frequency measuring instruction........ 166
6.10.10 PLSY: High-speed pulse output instruction........ 167
6.10.11 PLSR: Instruction for count pulse output with acceleration/deceleration........ 168
6.10.12 PLS: Envelopepulse output instruction........ 171
6.10.13 PWM: Pulse output instruction 172
6.11 Control calculation instructions........ 174
6.11.1 PID: Function instruction 174
6.11.2 RAMP: Ramp signal output instruction........ 178
6.11.3 HACKLE: Sawtooth wave signal output instruction 179
6.11.4 TRIANGLE: Triangle wave signal output instruction 180
6.11.5 ABSD: Cam absolute control instruction 181
6.11.6 DABSD: Double word cam absolute control instruction 183
6.11.7 ALT: Alternate output instruction 184
6.12 Communication instructions 184
6.12.1 MODBUS: Master station communication instruction........ 184
6.12.2 XMT: Free-port sending instruction 185

6.12.3 RCV: Free-port receiving instruction 186
6.12.4 MODRW: Modbus read/write instruction 187
6.13 Check instructions 191
6.13.1 CCITT: Check instruction 191
6.13.2 CRC16: Check instruction 191
6.13.3 LRC: Check instruction....................192
6.14 Enhanced bit processing instructions 193
6.14.1 ZRST: Instruction for resetting bits to 0 in batch 193
6.14.2 ZSET: Instruction for resetting bits in batch....................193
6.14.3 DECO: Decoding instruction 193
6.14.4 ENCO: Encoding instruction....................194
6.14.5 BITS: Instruction for counting on bit in word....................194
6.14.6 DBITS: Instruction for counting on bit in double word 195
6.14.7 BON: Instruction for judging on bit in word....................195
6.15 Word contact instructions 195
6.15.1 BLD: Word bit contact LD instruction 195
6.15.2 BLDI: Word bit contact LDI instruction 196
6.15.3 BAND: Word bit contact AND instruction....................196
6.15.4 BANI: Word bit contact ANI instruction....................197
6.15.5 BOR: Word bit contact OR instruction....................197
6.15.6 BORI: Word bit contact ORI instruction....................198
6.15.7 BOUT: Word bit coil output instruction 198
6.15.8 BSET: Word bit coil setinstruction 198
6.15.9 BRST: Word bit coil reset instruction....................199
6.16 Comparison contact instructions....................199
6.16.1 LD (=, <, >, <>, >=, <=): Integer comparison LD※instruction 199
6.16.2 AND (=, <, >, <>, >=, <=): Integer comparison AND ※instruction 200
6.16.3 OR (=, <, >, <>, >=, <=): Integer comparison OR※instruction 201
6.16.4 LDD (=, <, >, <>, >=, <=): Long integer comparison LDD※instruction 202
6.16.5 ANDD (=, <, >, <>, >=, <=): Long integer comparison ANDD※instruction 203
6.16.6 ORD (=, <, >, <>, >=, <=): Long integer comparison ORD※instruction....................204
6.16.7 LDR: Floating-point number comparison instruction 205
6.16.8 ANDR: Floating-point number comparison instruction 206
6.16.9 ORR: Floating-point number comparison instruction....................207
6.16.10 CMP: Instruction for setting integer comparison to ON 209
6.16.11 LCMP: Instruction for setting long integer comparison to ON....................209
6.16.12 RCMP: Instruction for setting floating-point number comparison to ON 210
6.17 Bulk data processing instructions 210
6.17.1 BKADD: Bulk data addition operation instruction 210
6.17.2 BKSUB: Bulk data subtraction operation instruction 211
6.17.3 BKCMP=,>,<,<>,<=,>=: Bulk data comparison instruction 211
6.18 Datasheet instructions 212
6.18.1 LIMIT: Upper/lower limit control instruction 212
6.18.2 DBAND: Deadband control instruction 213
6.18.3 ZONE: Zone control instruction 213
6.18.4 SCL: Coordinatesetting instruction....................214
6.18.5 SER: Data search instruction 215

6.19 Character string instructions........216
6.19.1 STRADD: String combination instruction........216
6.19.2 STRLEN: Instruction for detecting the string length........217
6.19.3 STRRIGHT: Instruction for reading a string from right........217
6.19.4 STRLEFT: Instruction for reading a string from left........218
6.19.5 STRMIDR: Instruction for reading any characters of a string........219
6.19.6 STRMIDW: Instruction for replacing any characters of a string........219
6.19.7 STRINSTR: String search instruction........220
6.19.8 STRMOV: String transmission instruction........221
6.20 Extension file register instructions........221
6.20.1 LOADR: Instruction for reading data from an extension file register........221
6.20.2 SAVER: Instruction for writing data to an extension file register........222
6.20.3 INITR: Instruction for initializing an extension register........223
6.20.4 LOGR: Instruction for logging on an extension register........224
6.20.5 INITER: Instruction for initializing an extension file register........225
6.21 Positioning instructions........227
6.21.1 ZRN: Zero return instruction........227
6.21.2 PLSV:Variable speed pulse output instruction........228
6.21.3 DRVI: Relative position control instruction........229
6.21.4 DRVA: Absolute position control instruction........230
6.21.5 DSZR: Instruction for zero return with DOG........231
6.21.6 DVIT: Interrupt positioning instruction........232
6.21.7 STOPDV: Pulse output stop instruction........233
6.21.8 LIN: Linear trajectory interpolation instruction........234
6.21.9 CW: Clockwise arc trajectory interpolation........237
6.21.10 CCW: Counterclockwise arc trajectory interpolation instruction........238
6.21.11 MOVELINK: Synchronous control instruction........241
6.21.12 GEARBOX: Electronic gear instruction........243
6.22 Data processing instructions........244
6.22.1 MEAN: Mean instruction........244
6.22.2 WTOB: Byte-unit data separation instruction........244
6.22.3 BTOW: Byte-unit data combination instruction........245
6.22.4 UNI: Instruction for combining 4bits of 16-bit data........246
6.22.5 DIS: Instruction for separating 4bitsof 16-bit data........247
6.22.6 ANS: Signal alarm set instruction........248
6.22.7 ANR: Signal alarm reset instruction........249
6.23 Other instructions........249
6.23.1 RND: Instruction for generating random numbers........249
6.23.2 DUTY: Instruction for generating timed pulses........250
Chapter 7 SFC Tutor........251
7.1 Introduction to SFC........252
7.1.1 What is SFC........252
7.1.2 What is SFC of VC series PLCs........252
7.1.3 Basic concept of SFC........252
7.1.4 Programming symbols and their usage........252
7.1.5 SFC program structure........253
7.1.6 Execution of SFC program........257
7.2 Relationship between SFC and LAD........258
7.2.1 STL instruction and step states........258
7.2.2 SET instruction........259

7.2.3 RET instruction and SFC program segment....259
7.2.4 OUT instruction and RST instruction....259
7.2.5 SFC selection branch, parallel branch and merge....259
7.3 How to program With SFC....259
7.4 Notesin SFC programming....260
7.4.1 Common programming errors....260
7.4.2 Programming tricks....263
7.5 Examples of SFC programming....264
7.5.1 Simple sequential structure....265
7.5.2 Selection branch structure....266
7.5.3 Parallel branch structure....269
Chapter 8 Operating guide for high-speed input function....274
8.1 High-speed counter....275
8.1.1 Configuration....275
8.1.2 Relationship between high-speed counter and SM auxiliary relay....276
8.1.3 How to use the high-speed counters....278
8.1.4 Notes about the VC1 series high-speed counters....281
8.2 External pulse capture function....282
8.3 Notes on High-speed input application....282
Chapter 9 Using Interrupts....283
9.1 Interrupt program....284
9.2 Processing interrupt event....286
9.3 Using timed interrupt....286
9.4 Using external interrupts....289
9.5 Using high-speed counter interrupt....290
9.6 Using PTO output completion interrupt....291
9.7 Using serial port-based interrupt....292
Chapter 10 Communication function guide....295
10.1 Communication resources....296
10.2 Programming port communication protocol....296
10.3 Free-port communication protocol....296
10.3.1 Introduction....296
10.3.2 Parameters setting of free-port....296
10.3.3 Free-port instruction....298
10.4 Modbus communication protocol....299
10.4.1 Introduction....299
10.4.2 Characteristics of links....299
10.4.3 RTU transmission mode....299
10.4.4 Supported Modbus function codes....300
10.4.5 Addressing mode of the PLC elements....300
10.4.6 Modbus slave station....301
10.4.7 Reading and writing elements....301
10.4.8 Processing of double word element....302
10.4.9 Processing of LONG INT data....302
10.4.10 Diagnostic function code....302
10.4.11 Error code....303
10.4.12 Modbus parameter setting....303
10.4.13 Modbus instruction....304
10.5 N:N communication protocol....306

10.5.1 N:N introduction 306
10.5.2 N:N network data transmission form 306
10.5.3 N:N network structure 307
10.5.4 N:N refresh mode......308
10.5.5 Enhanced refresh mode......312
10.5.6 N:N parameter setting......313
10.6 Several control strategies......315
10.6.1 Determination of the master station 315
10.6.2 Maximum number of inspection stations 315
10.6.3 Multi-master and slave (M: N) 315
10.6.4 Examples of using N:N......315
Chapter 11 Positioning function guide 316
11.1 Positioning control system......317
11.1.1 Absolute position system......317
11.1.2 Positioning control system......318
11.1.3 Process of positioning control 319
11.2 Overview of the VC series PLC positioning functions......319
11.3 Note of using the positioning instructions 322
11.4 Special elements related to the positioning instructions 323
11.4.1 Outputaxes of the VC series PLC 323
11.4.2 Outputaxes of the VC1 series PLC 324
11.5 Application instance......324
11.5.1 Configuration method of the PLS envelope instruction 324
Appendix A Special auxiliary relay......328
1. PLC working state flag......328
2. Clock running bit......328
3. User program execution error......329
4. Interrupt control 329
5. Peripheral instruction......331
6. 0peration flag bit......331
7. DHST/DHSP Table comparison completion flag 331
8. ASCII code conversion instruction flag......331
9. Pulse capture monitoring bit 331
10. Quad-frequency......332
11. Free port (PORT0)......332
12. Free port (PORT1)......332
13. Extended free port (PORT2)......333
14. N:N communication 334
15. System bus error flag 334
16. Real-time clock error flag......335
17. Enable flag of integrated analog channel 335
18. Count direction and monitoring of high-speed counters 335
19. High speed output and positioning instruction 336
1) Y0 Correlation flag bit......336
2) Y1 Correlation flag bit......337
3) Y2 Correlation flag bit......338
4) Y3 Correlation flag bit......338
5) Y4 Correlation flag bit......339
6) Y5 Correlation flag bit......340

7) Y6 Correlation flag bit......341
8) Y7 Correlation flag bit......341
20. Timed output instruction......342
21. Signal alarm......342
22. CANopen instruction......342

Appendix B Special data register......344

1. PLC working state data......344
2. Operation error code FIFO area......344
3. FROM/TO error......345
4. Scan time......345
5. Input filtering constant setup......345
6. Timed interrupt cycle......346
7. Real-time clock......346
8. Usage of DHSP and DHST instructions......346
9. Free-port receiving control and state (PORT0)......347
10. Free-port receiving control and state (PORT1)......348
11. Free-port receiving control and state (PORT2)......350
12. High-speed pulse output and positioning......352
1) Y0 Correlation register......352
2) Y1 Correlation register......353
3) Y2 Correlation register......354
4) Y3 Correlation register......355
5) Y4 Correlation register......355
6) Y5 Correlation register......356
7) Y6 Correlation register......357
8) Y7 Correlation register......358
13. Timed output instruction......358
14. Signal alarm instruction......359

Appendix C Reserved elements......360

Appendix D Modbus communication error codes......361

Appendix E System error codes......362

Appendix F Modbus communication protocols......364

1. Overview of Modbus communication protocol......364
2. Supported Modbus function code and element addressing mode......364
3. Modbus function code description......366
    3.1 Reading the coil state (0x01)......366
    3.2 (0x02)Reading the discrete input state (0x02)......366
    3.3 Reading holding registers (0x03)......366
    3.4 Forcing (writing) single coil (0x05)......367
    3.5 Presetting (writing) single register (0x06)......367
    3.6 Returning diagnostic check (0x08)......367
    3.7 Forcing (Writing) multiple coils(0x0F)......370
    3.8 Presetting (writing) multiple registers (0x10 Hex)......370
    3.9 Fault response frame (0x80+function code)......370
    3.10 Note......370
4. Example of Modbus communication control......371
5. Description of broadcast......372

Appendix G ASCII code character encoding table....374

Appendix H Instruction order index table....375

Appendix I Instruction classification index table....385

Chapter 1 Product overview

This chapter briefly describes the product elements, programming software platform, and network configuration and application of VC1 series PLCs.

Preface 1

Chapter 1 Product overview.. 1

- 1.1 Product introduction..1
  - 1.1.1 Product performance and specifications 1
  - 1.1.2 Appearance of VC1, and VC1 series main modules 5
- 1.2 Auto Studio programming software 5
  - 1.2.1 Basic configuration..6
  - 1.2.2 Installation process of the Auto Studio programming software..6
  - 1.2.3 Running interface of Auto Studio..6
  - 1.2.4 Programming cable 7
- 1.3 Communication function..7
  - 1.3.1 Modbus communication network..7
  - 1.3.2 N:N communication network 7
  - 1.3.3 Free-port network..8

1.1 Product introduction

VC series are integrated PLCs with built-in high-performance microprocessors and core computing control systems, integrating input and output points, expansion module bus, etc. I/O extension modules and special modules are also included in these series. The main module integrates 2 to 3 communication ports. VC2/3/5 series PLCs can be directly connected to the network while other series PLC main modules can be connected to the field bus network through the fieldbus expansion modules. The I/O configured on the main modules also includes high-speed counting and high-speed pulse output channels, which can be used for precise positioning. They are equipped with abundant built-in programming resources, adopt three standardized programming languages, and can implement commissioning and monitoring through the powerful Auto Studio programming software.Furthermore, optimal security protection mechanisms are provided for user programs.

1.1.1 Product performance and specifications

Table 1-1 Performance and specifications of PLC main modules

Name			VC5（Planning）	VC3（Planning）	VC2（Planning）	VC1	VC1S
I/O	Digital I/O point		16 in/16 out	16 in/16 out 32 in/32 out	16 in/16 out 32 in/32 out	8 in/6 out 12 in/8 out 14 in/10 out 16 in/14 out 24 in/16 out 36 in/24 out 16 in/14 out/ 2 AI/1 AO	8 in/6 out 12 in/8 out 14 in/10 out 16 in/14 out 24 in/16 out 36 in/24 out 16 in/14 out/ 2 AI/1 AO
	Max. number of logical I/O points		512	512	128	128	60
	Max. number of special modules		8	8	8	8	None
	High-speed pulse output		None	8×200 kHz (for transistor output only)	3×100 kHz (for transistor output only)	3×100 kHz (for transistor output only)	2×100 kHz (for transistor output only)
	Single-phase counting channel		8×200 kHz		2×50 kHz + 6×10 kHz		6×10 kHz
	Two-phase counting channel		4×200 kHz		1×30 kHz + 3×5 kHz		1×5 kHz
	Max.frequency sum of the high-speed counters		1600 kHz	1600 kHz	80 kHz	60 kHz	60 kHz
	Digital filtering		Applying digital filtering for X0 to X7. Input filtering constant range: 0–60 ms	Applying digital filtering for X0 to X7. Input filtering constant range: 0–60 ms	Applying digital filtering for X0 to X5. Input filtering constant range: 0–60000 us	Applying digital filtering for X0 to X7. Input filtering constant range: 0–60 ms	Applying digital filtering for X0 to X7. Input filtering constant range: 0–60 ms
	Max. current of the relay output point	Resistive load	2 A/ 1 point 8 A/ 4 point group common terminal 8 A/ 8 point group common terminal				
		Inductive load	220 V AC, 80 VA				
		Lamp load	220 V AC, 100 W				
	Max. current of the	Resistive load	Output point: 0.5 A/1 point Others: 0.5 A/1 point; 0.8 A/4 point; 1.6 A/8 point Above 8 points, the total current increases by 0.1 A for each addition point				

Name			VC5（Planning）	VC3（Planning）	VC2（Planning）	VC1	VC1S
	transistor output point	Inductive load	12 W/24 V DC	Y0–Y7: 7.2 W/24 V DC Others: 12 W/24 V DC	Y0, Y1,Y2: 7.2 W/24 V DC Others: 12 W/24 V DC		Y0, Y1: 7.2 W/24 V DC Others: 12 W/24 V DC
		Lamp load	1.5 W/24 V DC	Y0–Y7: 0.9 W/24 V DC Others: 1.5 W/24 V DC	Y0, Y1,Y2: 0.9 W/24 V DC Others: 1.5 W/24 V DC		Y0, Y1: 0.9 W/24 V DC Others: 1.5 W/24 V DC
Storage device	User program		64 kstep (128 kB)	64 kstep (128 kB)	32 kstep (64 kB)	16 kstep (32 kB)	6 kstep (12 kB)
	Permanent storage after power outage		Yes				
	Max. number of elements for which data is saved at power outage		All the soft elements except R	All the soft elements except R	All the soft elements	Bit element: full range Word element: 2000	Bit element: full range Word element: 2000
	Hardware support and endurance time		Backup battery for storage of 3 years	Backup battery for storage of 3 years	Backup battery for storage of 3 years	EEPROM, permanent storage	EEPROM, permanent storage
Soft element resources	Timer		100 ms precision: T0–T209 10 ms precision: T210–T479 1 ms precision: T480–T511		100 ms precision: T0–T209 10 ms precision: T210–T251 1 ms precision: T252–T255		
	Counter		16-bit increment counter: C0–C199 32-bit increment and decrement counter: C200–C235 32-bit high-speed counter: C236–C263		16-bit increment counter: C0–C199 32-bit increment and decrement counter: C200–C235 32-bit high-speed counter: C236–C263		
	Data register		D0–D7999, 0–R32767		D0–D7999		
	Local data register		V0–V63				
	Indexing register		Z0–Z15				
	Special data register		SD0–SD1023	SD0–SD1023	SD0–SD1023	SD0–SD511	SD0–SD511
	Auxiliary relay		M0–M10239	M0–M10239	M0–M2047		M0–M2047
	Local auxiliary relay		LM0–LM63				
	Special auxiliary relay		SM0–SM1023	SM0–SM1023	SM0–SM1023	SM0–SM511	SM0–SM511
	State relay		S0– S4095	S0– S4095	S0– S1023		S0– S1023
Interruption resource	Internal timed interruption		3	3	3		3
	External interruption		16	16	16		16
	High-speed counter-based interruption		8	8	8		8

Name		VC5（Planning）	VC3（Planning）	VC2（Planning）	VC1	VC1S
	Serial port-based interruption	12	12	6		8
	Interruption after PTO output	/	8	3		2
	Interruption after interpolation	/	1	/		/
	Interruption when passing a position	/	8	/		/
	Interruption at power outage	1	1	/		1
Regular	Running time of a basic instruction	0.09μS	0.09μS	0.09μS	0.2μS	0.2μS
	Real-time clock	Supporting a minimum of uninterrupted output of 3 years after power outage	Supporting a minimum of uninterrupted output of 3 years after power outage	Supporting a minimum of uninterrupted output of 3 years after power outage	Supporting a minimum of uninterrupted output of 3 years after power outage	None
	Analog potentiometer	None	None	None	None	None
Communication	Communication port	PORT0: RS232 PORT1: RS485 PORT2: RS485 PORT3: CAN PORT4: Network port PORT5: USB PORT : Network port	PORT0: RS232 PORT1: RS485 PORT2: RS485 PORT3: CAN PORT4: Network port PORT5: USB	PORT0: RS232 PORT1: RS485 PORT2: RS485 PORT3: USB （Note :Port 2 is expansion of communication port）		PORT0: RS232 PORT1: RS485
	Communication protocol	EtherCAT CANopen, Modbus-TCP, Modbus, free-port, N:N, programming port protocol	Modbus, free-port, N:N, and programing port protocols			
Access control and user program protection	Password type	Upload password Download password Monitoring password Subprogram password				
	Disable upload	Supported				
	Disable formatting	Supported				

Name		VC5（Planning）	VC3（Planning）	VC2（Planning）	VC1	VC1S
	Real-time clock, clock instruction	Available	Available	Available	Available	None
	Data and clock comparison instruction	Available	Available	Available	Available	None
	Floating-point number operation instruction	Available	Available	Available	Available	None
	Positioning instruction	Available	Available	Available	None	None
	High-speed I/O instruction	Available	Available	Available	Available	PLS instruction is not supported
	Modbus and inverter instruction	Available	Available	Available	Available	Available
	Read/write EEPROM instruction	None	None	None	None	None
	Control and calculation instruction	Available	Available	Available	Available	Support PID instruction only
	Character string instruction	Available	Available	None	None	None
	Batch data processing instruction	Available	Available	None	None	None
	Datasheet instruction	Available	Available	None	None	None
	Memory card instruction	None	None	None	None	None
Mean time between failures (MTBF)	Relay output	200,000 hours, fixed on the ground, with mechanical stress approximating zero and temperature and humidity control				
		100,000 hours, fixed on the ground, with mechanical stress approximating zero, no temperature and humidity control				
	Transistor output	300,000 hours, fixed on the ground, with mechanical stress approximating zero and temperature and humidity control				
		150,000 hours, with mechanical stress approximating zero, no temperature and humidity control				
Service life of the output relay contacts	220 V AC/15 VA/sensitive	1s ON/1s OFF, 3,200,000 times				
	220 V AC /30 VA/sensitive	1s ON/1s OFF, 1,200,000 times				
	220 V AC/72 VA/sensitive	1s ON/1s OFF, 300,000 times				
Power supply characteristics	Input voltage range	85 V AC–264 V AC (normal operation)				

Name	VC5（Planning）	VC3 （Planning）	VC2 （Planning）	VC1	VC1S
Notes: 1. For details about product specifications, installation instructions, and operation and maintenance of VC1 series PLCs, refer to the *VC Series PLC User Manual*. 2. For details about product specifications, installation instructions, and operation and maintenance of VC2 series PLCs, refer to the *VC2 Series PLC User Manual*. 3. For details about product specifications, installation instructions, and operation and maintenance of VC3 series PLCs, refer to the *VC3 Series PLC User Manual*. 4. For details about product specifications, installation instructions, and operation and maintenance of VC5 series PLCs, refer to the *VC5 Series PLC User Manual*. 5. The backup battery can support storage of 3 years in working environments of 25 °C.					

1.1.2 Appearance of VC1, and VC1 series main modules

Figure 1-1 shows the appearance and structure of VC1 series main modules (using VC1-1614MAT as an Application instance).

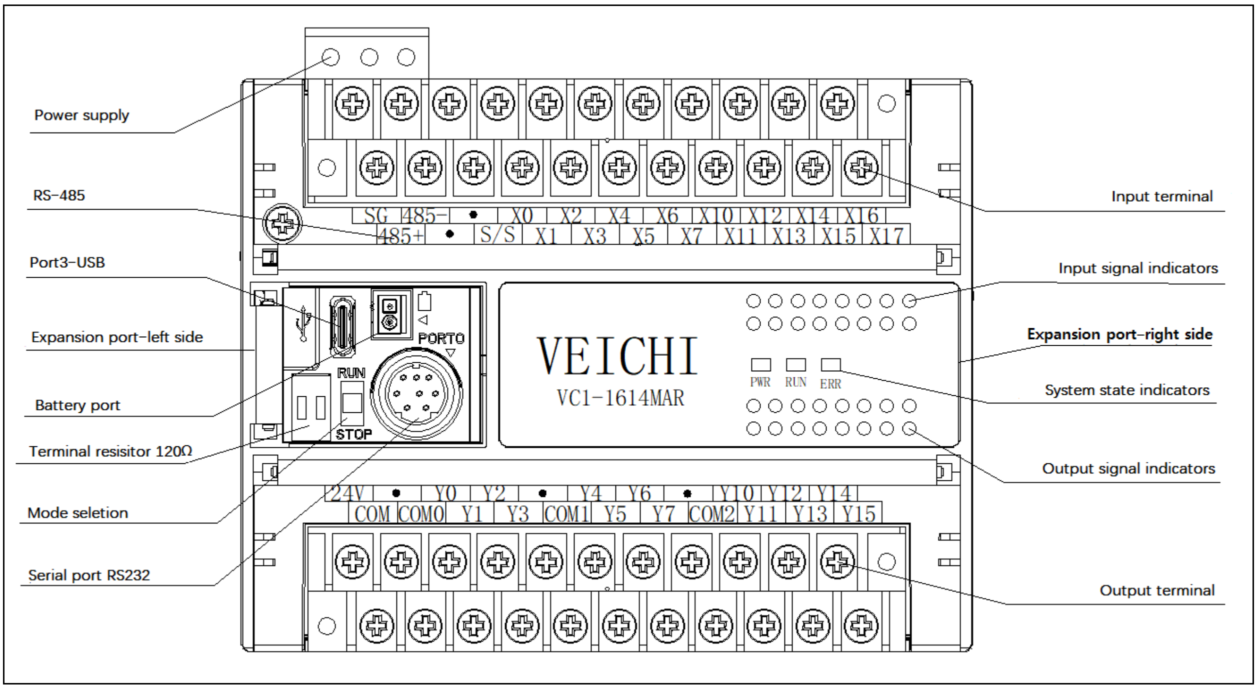

Power supply
RS-485
Port3-USB
Expansion port-left side
Battery port
Terminal resisitor 120Ω
Mode seletion
Serial port RS232
Input terminal
Input signal indicators
Expansion port-right side
System state indicators
Output signal indicators
Output terminal
SG 485- • X0 X2 X4 X6 X10 X12 X14 X16
485+ • S/S X1 X3 X5 X7 X11 X13 X15 X17
PORT0
RUN
STOP
VEICHI
VC1-1614MAR
PWR RUN ERR
24V • Y0 Y2 • Y4 Y6 • Y10 Y12 Y14
COM COM0 Y1 Y3 COM1 Y5 Y7 COM2 Y11 Y13 Y15

Figure 1-1 Appearance and structure of VC1, and VC1 series main modules

PORT0 and PORT1 are communication ports. PORT0 adopts the RS232 level and the Mini DIN8 socket. PORT1 and PORT2 of the VC1 series adopt the RS485 leve, PORT3 is Type-C port.

1.2 Auto Studio programming software

Auto Studio is programming software designed for VC1S, VC1, VC2, VC3 and VC5 series PLC products. This software can be downloaded at the company website.

Auto Studio programming software is standard Windows software and a graphical PLC programming tool, operated by using a mouse and keyboard. Three standard languages are available for programming: ladder diagram (LAD), instruction language (IL), and sequential function chart (SFC).

The Auto Studio programming software is connected to a PLC by using a serial programming cable. Modbus network programming can also be implemented through serial port conversion, and remote programming can be implemented through Modbus. For details about Modbus programming and remote monitoring, refer to the *Auto Studio Programming Software User Manual*.

1.2.1 Basic configuration

The Auto Studio programming software operates on IBM PC microcomputers or compatibles, and needs to be installed ina Microsoft Windows series operating system. Compatible operating systems include Windows 98, Windows Me, NT 4.0, Windows 2000, and Windows XP.

Table 1-2 describes the minimum and recommended configuration required by Auto Studio.

Table 1-2 Basic configuration conditions of Auto Studio

Item	Minimum configuration	Recommended configuration
CPU	Equivalent to Intel's Pentium 233 or higher	Equivalent to Intel's Pentium 1G or higher
Memory	64 MB	128 MB
Graphics card	Can operate at the resolution of 640 x 480 in 256-color mode	Can operate at the resolution of 800 x 600 in 65535-color mode
Communication interface	An RS232 serial port provided by a DB9 type socket is required (or you can use the USB interface, but a USB-RS232 converter is required).	
Other devices	Programming cables designed for VEICHI PLCs	

1.2.2 Installation process of the Auto Studio programming software

The Auto Studio installation package released by Suzhou Veichi Electric Co.,Ltd. is a standalone executable program. Double-click it to start the installation process and then install it step by step according to the installation wizard. You can choose an installation path as required.

After the installation is complete, the VEICHI program group appearson the Start Menu, and the installer also generates a shortcut icon of Auto Studio on the desktop. You can double-click the icon to run the program.

Uninstallation: The software can be uninstalled on the Windows Control Panel. To upgrade and install a new version of Auto Studio, the earlier version needs to be uninstalled first.

1.2.3 Running interface of Auto Studio

The main interface of the program generally consists of seven parts: menu, toolbar, project manager window, instruction tree window, message window, status bar, and workspace.

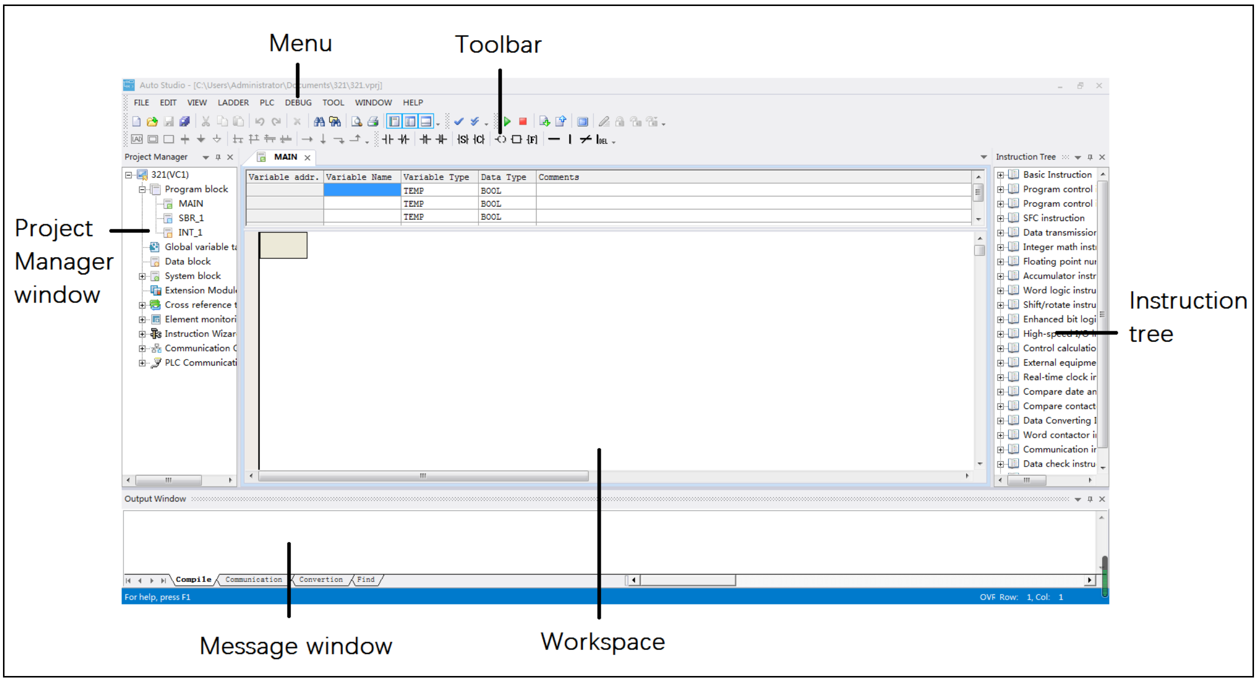

Menu
Toolbar
Project Manager window
Instruction tree
Message window
Workspace
FILE EDIT VIEW LADDER PLC DEBUG TOOL WINDOW HELP
Project Manager
MAIN
Instruction Tree
Output Window
Compile
Communication
Convertion
Find
For help, press F1

Figure 1-4 Main interface of Auto Studio

Refer to the *Auto Studio Programming Software User Manual* for details about how to use the Auto Studio programming software.

1.2.4 Programming cable

Customers can use the serial programming cable provided by VEICHI to program and perform commissioning on the PLC.

Figure 1-5 shows the connection of a programming cable.

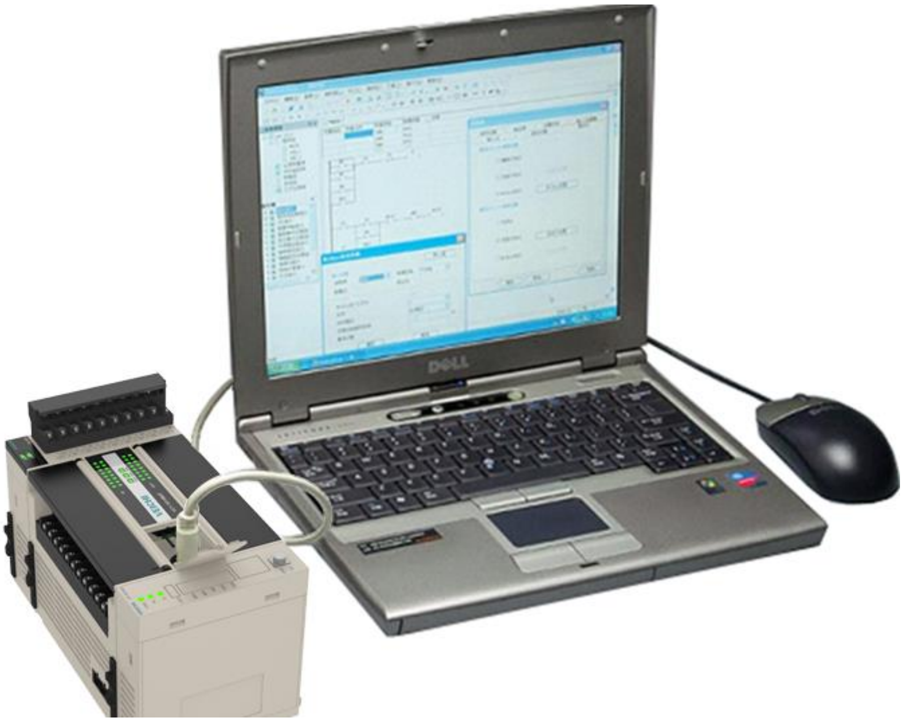


Figure 1-5 Programming cable connection

1.3 Communication function

The main module of VC1/VC1S series micro-PLC integrates two serial ports: PORT0 ,PORT1 and one USB port. PORT0 and PORT1 suppor firmware upgrade and programming.These serial ports adopt the Modbus and N: N communication protocols, and the user-defined free-port protocol is also applicable. The PORT0 port cannot be used for N:N network connection.

1.3.1 Modbus communication network

An VC series PLC can form a Modbus RS485 network with other intelligent instruments or devices, such as inverters and PLCs, through the RS485 interfaces PORT1 and PORT2 or through PORT0 (when using PORT0, a RS232-RS485 converter is required). The maximum communication distance is 1200 meters and the highest communication baud rate is 115200 bit/s. You can select the RTU transmission mode.

An VC series PLC can be connected to an inverter, PLC, touch screen, or instrument through the RS232 interface PORT0. The maximum communication distance is 15 meters and the highest communication baud rate is 115200 bit/s.

For details about Modbus communication,refer to Chapter 10"Communication function guide" and Appendix G "Modbus communication protocols".

1.3.2 N:N communication network

VC1S, VC1, VC2, VC3 and VC5 series PLCsare configured with the N:N communication protocol developed by VEICHI, which can be applied to form an N:N communication network through the RS485 interfaces PORT1 and PORT2. With the conversion of RS232 to RS485, the PORT0 port can also be used for N:N network connection.

The N:N communication protocol allows 2 to 32 PLC stations to exchange data with each other. The highest communication baud rate is 115200 bps. This protocol can be applied to form a single-layer or double-layer network.

For details about N:N communication, refer to Chapter 10 "Communication function guide".

1.3.3 Free-port network

The free-port protocol is a protocol mode in which the communication is performed based on user-defined communication data formats. It supports two data formats: ASCII and binary. In this communication port mode, a PLC can be used to communicate with various devices that use user-defined formats, such as inverters, barcode scanners, instruments, and other intelligent devices that adopt free communication protocols. A PLC can communicate with one device in RS232 or RS485 mode, and it can also form an RS485 network with multiple devices.

For details about the free-port protocol communication, refer to Chapter 10 "Communication function guide".

Chapter 2 Function Description

This chapter briefly describes the programming resources and principles and system configuration method of VC series PLCs, how to set the PLC running and operation mode, and the various functions and software tools related to system commissioning. For details, refer to the *Auto Studio Programming Software User Manual*.

Chapter 2 Function Description 11
2.1 Programming resources and principles 12
  2.1.1 Programming resources 12
  2.1.2 PLC operating mechanism (Scan cycle model) 14
  2.1.3 User program running watchdog 15
  2.1.4 Constant scan operation mode 15
  2.1.5 User file download and storage 15
  2.1.6 Element initialization 15
  2.1.7 Saving data at power outage 15
  2.1.8 Digital filtering for input points 16
  2.1.9 No battery mode 16
  2.1.10 User program protection 16
2.2 System configuration 17
  2.2.1 System block 17
  2.2.2 Data blocks 23
  2.2.3 Global variable table 24
2.3 Running mode and state control 24
  2.3.1 Concepts of system run and stop states 24
  2.3.2 Run and stop state 24
  2.3.3 Setting the state of output pointsin the stop state 25
2.4 System commissioning 25
  2.4.1 Program download and upload 25
  2.4.2 Error reporting mechanism 26
  2.4.3 Modifying the user program online 28
  2.4.4 Clearing and formatting 28
  2.4.5 Checking PLC information online 29
  2.4.6 Element value writing and forcing, and element monitoring table 29
  2.4.7 Generating data blocks from RAM 32

2.1 Programming resources and principles

2.1.1 Programming resources

Table 2-2 VC1 programming resources

Name		Specification and description
I/O Configuration	Max. number of I/O points	128 (theoretical value)
	Number of extension modules	15
User file capacity	User program capacity	16 ksteps
	Block size	8000 D elements
Instruction speed	Basic instruction	0.2 us/instruction
	Application instruction	A few us – hundreds us /instruction
Number of instructions	Basic instruction	32
	Application instruction	234
Soft element resources^note 7	I/O points	128 inputs/128 outputs (inputs: X0–X177; outputs: Y0–Y177)^note 1
	Auxiliary relay	2048 (M0–M2047)
	Local auxiliary relay	64 (LM0–LM63)
	Special auxiliary relay	512 (SM0–SM511)
	State relay	1024 (S0–S1023)
	Timer	256 (T0–T255)^note 2
	Counter	256 (C0–C255)^note 3
	Data register	8000 (D0–D7999)
	Local data register	64 (V0–V63)
	Indexing register	16 (Z0–Z15)
	Special data register	512 (SD0–SD511)
Interruption resources	External input-based interruption	16 (interruption triggering edges can be defined by users, corresponding to the rising and falling edges of the X0 to X7 terminals)
	High-speed counter-based interruption	8
	Internal timed interruption	3
	Serial port-based interruption	6
	Interruption after PTO output	2
	Interruption at power outage	None
Communication function	Communication interface	Two asynchronous serial communication ports: PORT0: RS232 PORT1 One USD port
	Communication protocol	Modbus, free-port, N:N (VEICHI proprietary protocol, which can be applied to form 1:N or N:N communication networks)

Name		Specification and description	
Special function	High-speed counter	X0, X1	Single input: 50 kHz Simultaneous inputs of X0 to X7: The total frequency is no more than 80 kHz.
		X2–X5	Single input: 10 kHz
	High-speed pulse output	Y0, Y1, Y2	Three independent outputs: 100 kHz (for transistor output type only)
	Digital filtering	Applying digital filtering for X0–X7 and hardware filtering for other ports	
	Analog potentiometer[note 4]	None	
	Subprogram call	A maximum of 64 user subprograms and six levels of nested subprograms are allowed. Local variables are supported, and each subprogram can provide a maximum of 16 parameters for information transmission. Variable aliases are also supported.	
Special function	User program protection	Upload password	Three types of passwords are provided. A password is no more than 8 characters, and each character is alphanumeric and case sensitive.
		Download password	
		Clock password	
		Subprogram encryption	A password is no more than 16 characters, and each character is alphanumeric and case sensitive.
		Other protective measures	The functions of disabling formatting and upload are provided.
	Programming mode[note 5]	Auto Studio programming software[note 6]	It needs to be installed and run on IBM PC microcomputer or compatibles.
	Real-time clock	Built-in, and power supply by backup battery.	

Notes:

Note 1: The addresses of the X and Y elements are numbered in octal, for Application instance: Address X10 represents the 8th input point.

Note 2: Addresses of the T elements are classified into three categories according to timing precision:

VC1

(1) 100 ms precision: T0 – T209

(2) 10 ms precision: T210 – T251

(3) 1 ms precision: T252 – T255

Note 3: Addresses of the C elements are classified into three categories according to the width and function of count values:

VC1

(1) 16-bit increment counter: C0 – C199

(2) 32-bit increment and decrement counter: C200 – C235

(3) 32-bit high-speed counter: C236 – C263

Note 5: The element forcing function is provided to facilitate commissioning and user program analysis, and thus improve commissioning efficiency. A maximum of 128 bit elements and 16 word elements can be forced simultaneously.

Note 6: The user program can be modified online when the PLC is running.

Note 7: Some internal soft element resources of PLCs have been reserved for internal use. Do not use these elements on the user program, if possible. For details, refer to Appendix C"Reserved elements".

2.1.2 PLC operating mechanism (Scan cycle model)

VC series PLC main modules operate according to the scan cycle model.

The system performs four tasks in the sequential and cyclical manner: executing the user program, communication, housekeeping, and refreshing I/O. Each round of tasks is called a scan cycle.

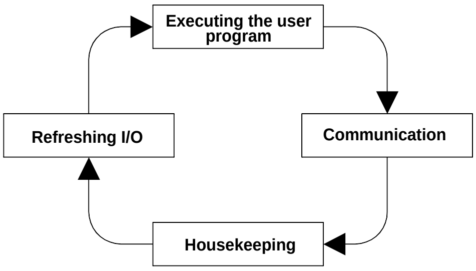


Figure 2-1PLC operating mechanism

- Executing the user program
  The system sequentially executes the instructions of the user program, starting from the first instruction to the end instruction of the main program.
- Communication
  The system communicates with the programming software and responds to the programming communication instructions, such as the download, run, and stop instructions sent by the programming software.
- Housekeeping
  The system handles various system housekeeping tasks such as refreshing panel indicators, updating timing value of the software timer, refreshing special auxiliary relaysand special data registers.
- Refreshing I/O
  I/O refresh includes an output refresh phase and an input refresh phase.

Output refresh phase: Switching the corresponding hardware output point on or off according to the value of the Y element (ON or OFF).

Input refresh phase: Converting the on/off state of the hardware input point to the corresponding X element value (ON or OFF).

2.1.3 User program running watchdog

The system monitors the running time of the user program in each scan cycle. Once detecting that the running time of the user program exceeds the set value, the system stops the user program. You can set the watchdog time on the **Set Time** tab of the system block dialog boxon the Auto Studio background software interface.

2.1.4 Constant scan operation mode

In the constant scan operation mode, the time of each scan cycle is the same when the system is running. You can activate the constant scan mode and set the constant scan time on the **Set Time** tab of the system block dialog box of the Auto Studio background software interface. The default constant scan cycle is 0, that is, constant scan is disabled. When the actual scan cycle is longer than the constant scan cycle, the program runsin the actual scan cycle.

🕮 Note

The setting of the constant scan time cannot be longer than the watchdog time.

2.1.5 User file download and storage

You can program and control a main module by downloading a specific user file to it.

There are four types of user files: user program files, data block files, system block files, and user assistance information files. User assistance information files includeglobal variable tables and user data source files.

You can choose to download a user program file, data block file, or system block file. When a file is selected, the corresponding user assistance information file is bundled and downloaded.

All user files of the VC1S and VC1 series are written into the FLASH area of the main module for permanent storage.

🕮 Note

1. Ensure that the power supply of the main module works properly in a period of time (longer than 30s) after the file is downloaded, so that the file can be properly written into the main module.

2.1.6 Element initialization

When the PLC enters the running state (STOP→RUN), the device initializes the related soft elements according to the data saved at power outage, data stored on EEPROM, data blocks, and element values. Table 2-6 describes the priorities of the data.

Table 2-6 PLC data initialization priorities

Storage device type	Power OFF→ON	STOP→RUN
Data saved at power outage	Highest	Highest
Data storage on EEPROM	High	High
Data blocks (when "Datablock enabled" is selected in the advanced settings of the system block)	Medium	Medium
Element values (when "Element value retained" is selected in the advanced settings of the system block)	—	Low

2.1.7 Saving data at power outage

- Conditions for saving data at power outage

  When detecting that a power outage occurs, the system stops the user program and saves the data values of the elements set within the storage range in the system block to the power outage backup file.
- Element recovery at power-on

If the power outage backup file is correct after power-on, the values of the specified soft elements are restored to the values saved at the last power outage.

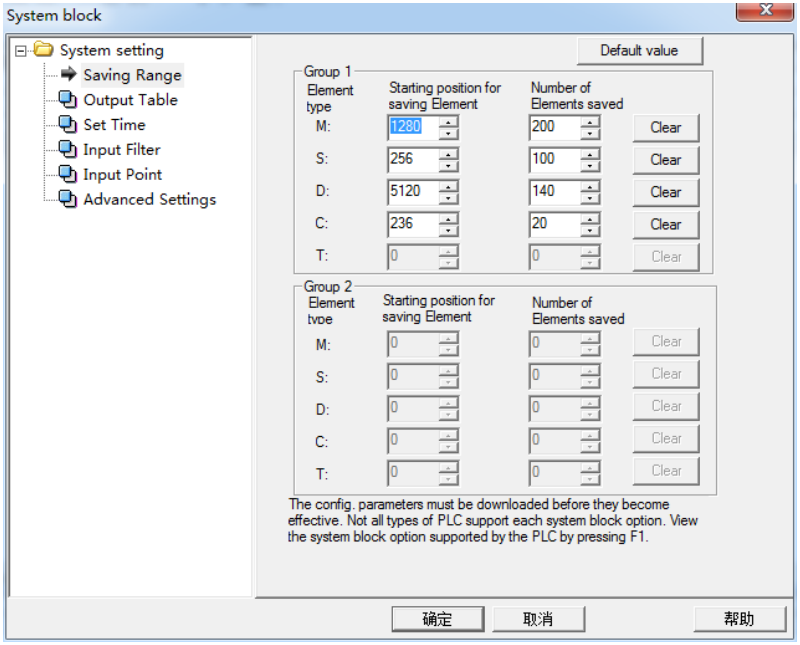
After power-on, the system deletes the data values of the elements that are out of the storage range.

If the backup file is lost or incorrect, data of all elements are deleted.

- Storage range setting

The range of elements for which data values are to be saved can be set in the **Saving Range** of the system block, as shown in Figure 2-2.

The storage range of the VC1S and VC1 series can be set only to a group.



Group 1

Element type	Starting position for saving Element	Number of Elements saved
M:	1280	200
S:	256	100
D:	5120	140
C:	236	20
T:	0	0

Group 2

Element tvoe	Starting position for saving Element	Number of Elements saved
M:	0	0
S:	0	0
D:	0	0
C:	0	0
T:	0	0

The config. parameters must be downloaded before they become effective. Not all types of PLC support each system block option. View the system block option supported by the PLC by pressing F1.

Figure 2-2 Setting the saving range

📖 Note

1. When an VC1S/VC1 series PLC encounters a power outage, data of its elements set within the storage range is stored in EEPROM.

2.1.8 Digital filtering for input points

Input points X0 to X7 of VC1S, VC1, VC2,VC3 and VC5 series main modules are configured with the digital filtering function which can be used to filter the interfering signal of the ports. You can change the input filtering constant by configuring the **Input Filter** in the system block.

2.1.9 No battery mode

VC1 series main modules can operate without batteries. When you select the no battery mode, the system reports no system error caused by the lack of battery (loss of element data, loss of forcing tables, user program file error).

Refer to the description of the **No battery mode** configuration in the **Advanced Settings** of the system block.

2.1.10 User program protection

VC Series PLCs are designed with multi-level password protection and other security strategies.

Table 2-7 User program protection

User program protection	Description
Disable formatting	After disabling formatting in the system block configuration and downloading the system block to a PLC, you cannot delete the user program, system block, and data block on the PLC by formatting. To enable formatting, you need to download a new system block in which formatting is not disabled.

User program protection	Description
Download password	Used to restrict the download function
Disable upload	In the download operation, select the upload disabled option in the download dialog box, and then you cannot upload the data later even you enter the upload password. To enable upload, you need to download the user data again and select the upload enabled option in the download dialog box.
Upload password	Used to restrict the upload function
Clock password	Used to restrict the clock function
Program password	Programmers can set passwords to encrypt the main program, subprograms and interruption subprograms. When opening the project in the programming software, you cannot view and edit the content of the encrypted programs. To view and edit it, you need to open the decryption dialog box and enter the correct password for decryption. Encryption method: Right-click the program to be encrypted, select **Encrypt/Decrypt** in the right-click menu, and then enter the password and confirm the password. Decryption method: Right-click the program to be decrypted, select **Encrypt/Decrypt** in the right-click menu, and then enter the correct password.

📖 Note

If you enter a wrong password for five consecutive times, the VC series micro-PLC disables password input for five minutes.

2.2 System configuration

2.2.1 System block

PLC configuration informationconfigured in a system block is compiled as a system block file, which is an important PLC user file. Before operating a PLC, you need to compile and download a system block file.

The system block configuration includes the following items:

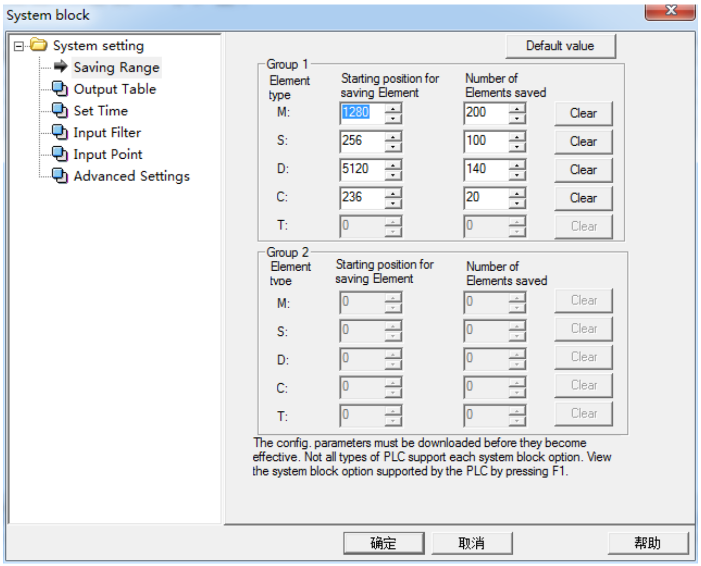
- Saving range (element saving range)
- Set time (watchdog, constant scan time and power outage detection time)
- Input point (startup mode of the input point)
- Communication ports (communication ports and protocol settings)
- Interrupt priority (interrupt priority configuration)
- Inverter configuration
- Ethernet configuration
- Output table (output table settings)
- Input filter
- Advanced settings (data block, element value retained, no battery mode, anddisable formatting)
- Special module configuration
- Communication module
- CANopen configuration

After configuring a system block, you can select **PLC/Compile All** menu to compile a system block file and then download it.

- Saving range

  Up on power outage, VC series PLCs can save data of the elements that are set within the storage range to the power-outage storage area, and the data can be retained and used after the machine is powered on again.

  Figure2-3 shows the address range of the elements for which data is to be saved on the first tab of the dialog box.



Group 1

Element type	Starting position for saving Element	Number of Elements saved
M:	1280	200
S:	256	100
D:	5120	140
C:	236	20
T:	0	0

Group 2

Element type	Starting position for saving Element	Number of Elements saved
M:	0	0
S:	0	0
D:	0	0
C:	0	0
T:	0	0

The config. parameters must be downloaded before they become effective. Not all types of PLC support each system block option. View the system block option supported by the PLC by pressing F1.

Figure 2-3 Configuring the address range of the elements for which data are to be saved

---

📖 Note

The address ranges and groups of elements for which data can be saved vary according to PLC models.

By default, a certain storage range is automatically set for each of the D, M, S, T, and C elements.

You can change the address range of the elements for which data is to be saved on the tab as you need. Click the Clear button on the right to set the number of the elements (for which data is to be saved) to zero.

Only one group can be set for each of the VC1S and VC1 series.

📖 Note

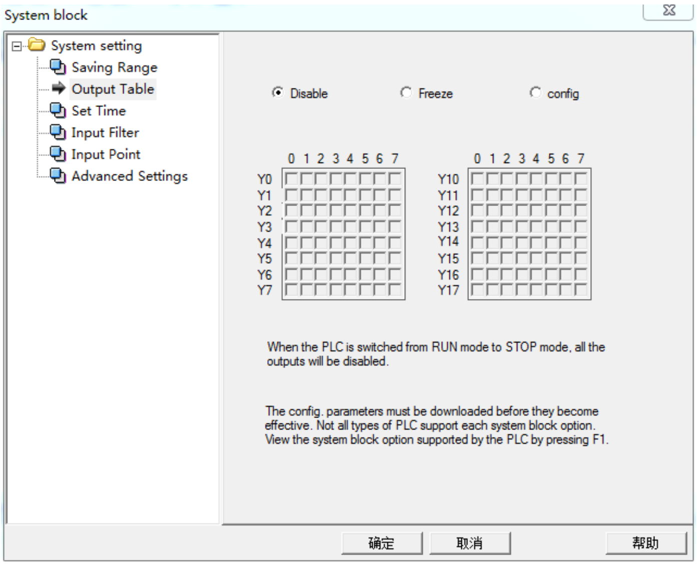
For VC1S and VC1 series PLCs, no T element can be set within the storage range.

System operation at power outage: The PLC saves the data of the elements set within the storage range to the power-outage backup file.

System operation at power-on: The PLC checks data in power-outage storage area. If the data saved in power-outage storage area is correct, the storage area on SRAM remains unchanged. If the data is incorrect, the PLC deletes data of all elements (including the elements that are within and out of the storage range) on SRAM.

- Output table

Click the **Output Table** tab and then you can set the state an output point enters after the PLC is stopped, as shown in Figure 2-4.


System block
System setting
Saving Range
Output Table
Set Time
Input Filter
Input Point
Advanced Settings
0 1 2 3 4 5 6 7
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
Y10
Y11
Y12
Y13
Y14
Y15
Y16
Y17
确定
取消
帮助

● Disable ○ Freeze ○ config

When the PLC is switched from RUN mode to STOP mode, all the outputs will be disabled.

The config. parameters must be downloaded before they become effective. Not all types of PLC support each system block option. View the system block option supported by the PLC by pressing F1.

Figure 2-4 Setting the output table

The output table setting function is used to set the state an output point enters after the PLC is stopped. An output point can enter one of the following state after the CPU of the PLC is stopped:

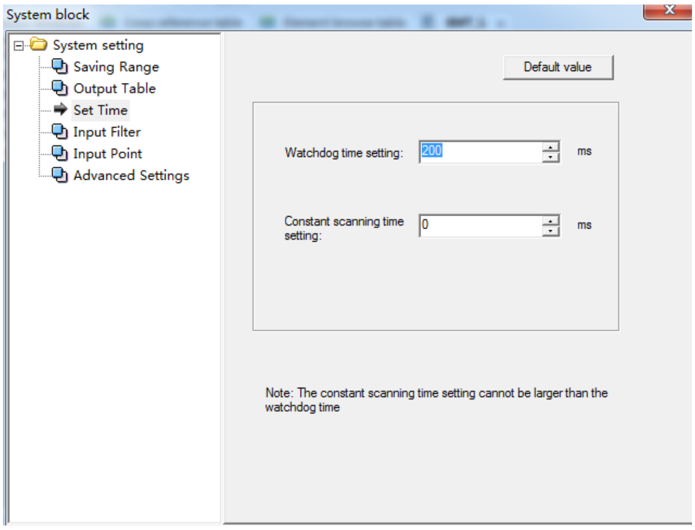
(1) Disable:The PLC disables all the output points when switching from the running state to the stopped state.

(2) Freeze: The PLC freezes all the output points in their final state when being stopped.

(3) Configuration: The PLC sets all the output points to a known state when being stopped. The default state of all the output points is off (0).

- Set time

Figure 2-5 shows the set time tab.


System block
System setting
Saving Range
Output Table
Set Time
Input Filter
Input Point
Advanced Settings
Default value

Watchdog time setting: 200 ms

Constant scanning time setting: 0 ms

Note: The constant scanning time setting cannot be larger than the watchdog time

Figure 2-5 Set time

1. Watchdog time setting

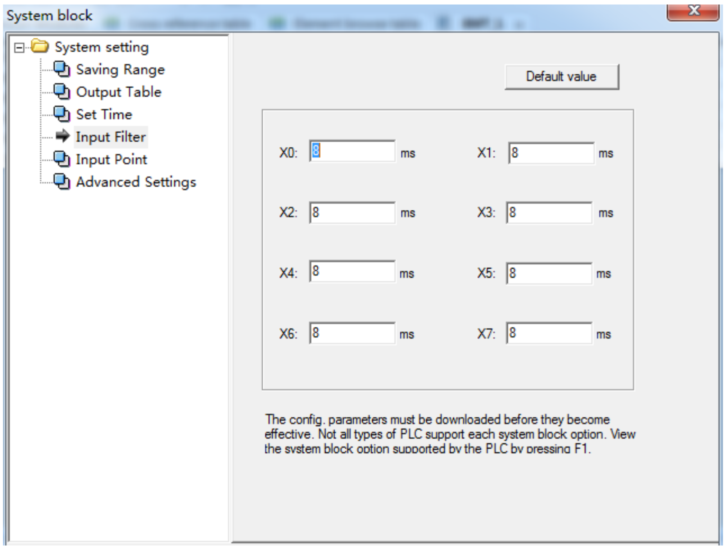
You can set the user program running watchdog time. The watchdog time is the allowed maximum time for the user program to run. When the actual execution time of the user program exceeds the watchdog time, the PLC stops the execution of the user program, turns on the program alarm indicator (red), and provides output according to the system configuration. The watchdog time can be set from 0ms to 1000ms and the default value is 200ms.

2. Constant scan time setting

Constant scan time is a constant time when the system scans a register. The system's constant scan time setting register is read and the user program is scanned only once at the constant time. The constant time can be set from 0ms to 1000ms and the default is 0ms. The default value is 0ms and constant scan time is not enabled.Enables the set constant scan time when it is non-zero.

- Input filter

Clicking the **Input Filter** tab to set an input filtering constant for the PLC input points. The external interfering signals introduced by the input points can be filtered out by digital filtering. X0 to X7 are digital input points with the digital filtering function. Other digital input points adopt hardware filtering. VC1S/VC1 series input filtering can be set separately for each input port and the filtering constants can be set consecutively from 0us to 60ms. Figure 2-6 shows the input filtering settings for the VC1 series.


System block
System setting
Saving Range
Output Table
Set Time
Input Filter
Input Point
Advanced Settings
Default value

X0: 8 ms X1: 8 ms

X2: 8 ms X3: 8 ms

X4: 8 ms X5: 8 ms

X6: 8 ms X7: 8 ms

The config. parameters must be downloaded before they become effective. Not all types of PLC support each system block option. View the system block option supported by the PLC by pressing F1.

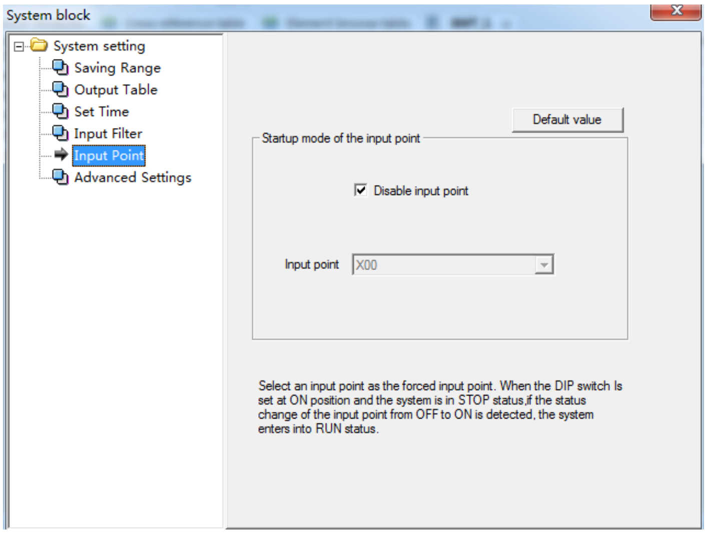
Figure 2-6 Setting the input filter

- Input point

Figure 2-7 shows the input point setting tab.

1. Set the input point for starting the PLC

When the **Disable input point** is not selected, you can set an input point (among X0 to X17) to force the PLC to enter RUN state. The PLC switches from the STOP state to RUN state when it is detected that the input point is ON.


System block
System setting
Saving Range
Output Table
Set Time
Input Filter
Input Point
Advanced Settings
Default value

Startup mode of the input point

☑ Disable input point

Input point X00

Select an input point as the forced input point. When the DIP switch Is set at ON position and the system is in STOP status,if the status change of the input point from OFF to ON is detected, the system enters into RUN status.

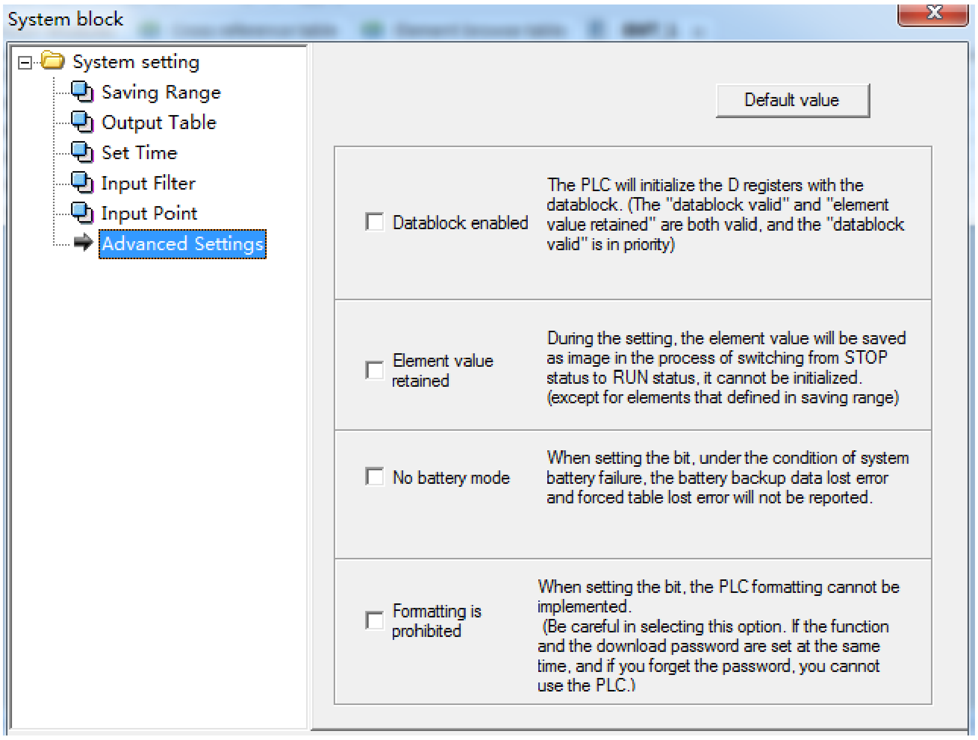
Figure 2-7 Input point setting

2. Disable input point

Select the **Disable input point** to disable the input point-based start function.

- Advanced settings

The advanced settings include Datablock enabled, Element value retained, no battery mode, etc.



☐ Datablock enabled — The PLC will initialize the D registers with the datablock. (The "datablock valid" and "element value retained" are both valid, and the "datablock valid" is in priority)

☐ Element value retained — During the setting, the element value will be saved as image in the process of switching from STOP status to RUN status, it cannot be initialized. (except for elements that defined in saving range)

☐ No battery mode — When setting the bit, under the condition of system battery failure, the battery backup data lost error and forced table lost error will not be reported.

☐ Formatting is prohibited — When setting the bit, the PLC formatting cannot be implemented. (Be careful in selecting this option. If the function and the download password are set at the same time, and if you forget the password, you cannot use the PLC.)

Figure 2-8 Advanced settings

4. Datablock enabled

Select Datablock enabledand the PLC uses data blocks to initialize D elements when switching from STOP to RUN.

5. Element value retained

Select Element value retained,and the PLC does not initialize the element value but save them as image when switching from STOP to RUN.

📖 Note

When "**Datablock enabled**" and "**Element value retained**" are both valid, "**Datablock enabled**"prevails. For details, refer to section 2.1.6 "Element initialization".

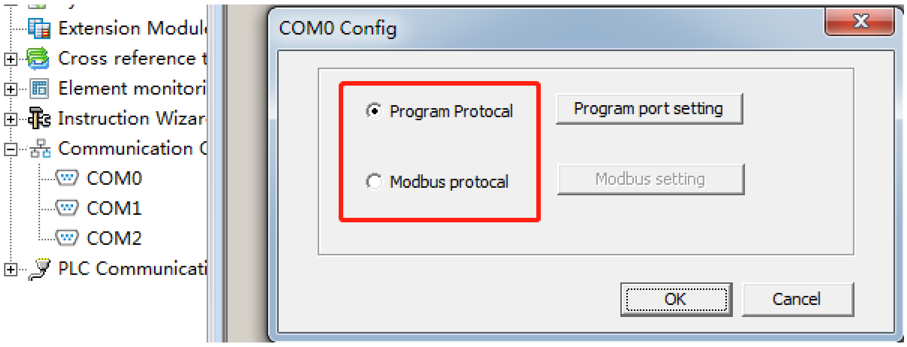
---

6. No battery mode

Select **No battery mode**, and the system does not report errors caused by loss of battery backup data or loss of forcing tables upon backup battery failure.

- Communication port

You can set three communication ports on the communication port tab of the system block. Figure 2-9 shows the communication port settings. The item includes communication protocol selection and specific protocol parameter settings.



● Program Protocal

○ Modbus protocal

Figure 2-9 Communication port setting

By default, PORT0 adopts the programming port protocol while PORT1 and PORT2 adopt no protocol. You can set PORT0, PORT1, and PORT2 as you need.

1. Programming port protocol

By default, PORT0 adopts the programming port protocol, a dedicated protocol for the communication of the VC series PLC programming software. With this protocol, you can set the communication baud rate between a PC and PORT0 through the serial port configuration tool of Auto Studio.

2. Free-port protocol

The free-port protocol is a communication modeusing user-defined data file formats. The free-port communication mode supports two data formats, namely ASCII and binary code. A PLC can adopt free-port communication only in the RUN state. When adopting the free-port mode, the PLC cannot communicate with programming devices.

The configurable parameters include baud rate, data bit, parity check, stop bit, start character detection, end character detection, intercharacter timeout, and interframe timeout.

3. Modbus communication protocol

Modbus communication devices include master and slave stations. A master station can communicate witha slave station (such as an inverter) and send control frames to the slave station according to the function codes of the Modbus communication protocol, and the slave station responds to the request of the master station.

PORT0 can be set as a slave station while PORT1 and PORT2 can be set as a master or slave station.

The configurable parameters include baud rate, data bit, parity check, stop bit, master-slave mode, station number, transmission mode, and timeout time and retry times of the master mode.

4. N:N communication protocol

N:N is an N:N communication protocol developed by Suzhou Veichi Electric Co.,Ltd. for micro-PLC networks. PLCs in the N:N network can automatically exchange some of their D and M element values.

All of the PORT0, PORT1, and PORT2 can adopt the N:N communication protocol.

📖 Note

For details about how to use the free-port, Modbus, and N:N protocols, refer to Chapter 10"Communication function guide".

- Expansion module configuration

You can set the module property on the expansion module configuration tab, as shown in Figure 2-10.

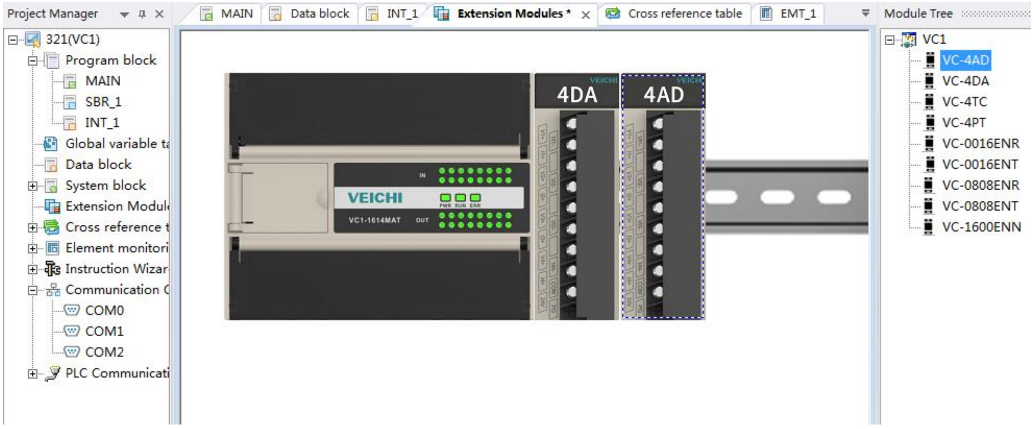

Project Manager
321(VC1)
Program block
MAIN
SBR_1
INT_1
Global variable ta
Data block
System block
Extension Modul
Cross reference t
Element monitori
Instruction Wizar
Communication C
COM0
COM1
COM2
PLC Communicati
MAIN
Data block
INT_1
Extension Modules *
Cross reference table
EMT_1
4DA
4AD
VEICHI
VC1-1614MAT
Module Tree
VC1
VC-4AD
VC-4DA
VC-4TC
VC-4PT
VC-0016ENR
VC-0016ENT
VC-0808ENR
VC-0808ENT
VC-1600ENN

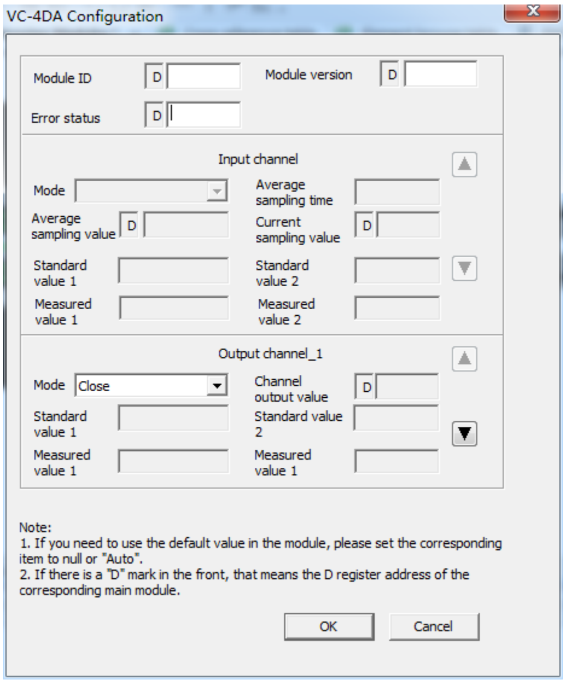
Figure 2-10 Expansion module configuration

1. Module type

You can select the module type for VC-4AD,VC-4DA,VC-4PT,VC-4PT and other module, as shown in Figure 2-10.

2. Module property

After selecting the **Module Type**, the corresponding **Module Property** is activated and the following dialog box may be displayed.


VC-4DA Configuration
Module ID
D
Module version
D
Error status
D
Input channel
Mode
Average sampling time
Average sampling value
D
Current sampling value
D
Standard value 1
Standard value 2
Measured value 1
Measured value 2
Output channel_1
Channel output value
D
Standard value 1
Standard value 2
Measured value 1
Measured value 1
OK
Cancel

Mode: Close

Note:
1. If you need to use the default value in the module, please set the corresponding item to null or "Auto".
2. If there is a "D" mark in the front, that means the D register address of the corresponding main module.

Figure 2-11 Module property setting

In the dialogue box shown in Figure 2-11, you can configure the channel for the special module, including **Mode** (signal features), **Digital value at zero**, **Upper limit of digital value**, **Average sampling** and **Current sampling value**. For details about the meanings and configuration methods of the parameters, refer to user manuals of the corresponding special modules.

2.2.2 Data blocks

Data blocks are used to set the default values in D elements. If you download the compiled data block settings to a PLC, the PLC uses the data block to initialize the related D elements upon the startup.

The data block editor enables you to assign initial data to a D register (data memory). You can assign data to words or double words, but not to bytes. You can also add comments by inputting "//" to the front of a character string.

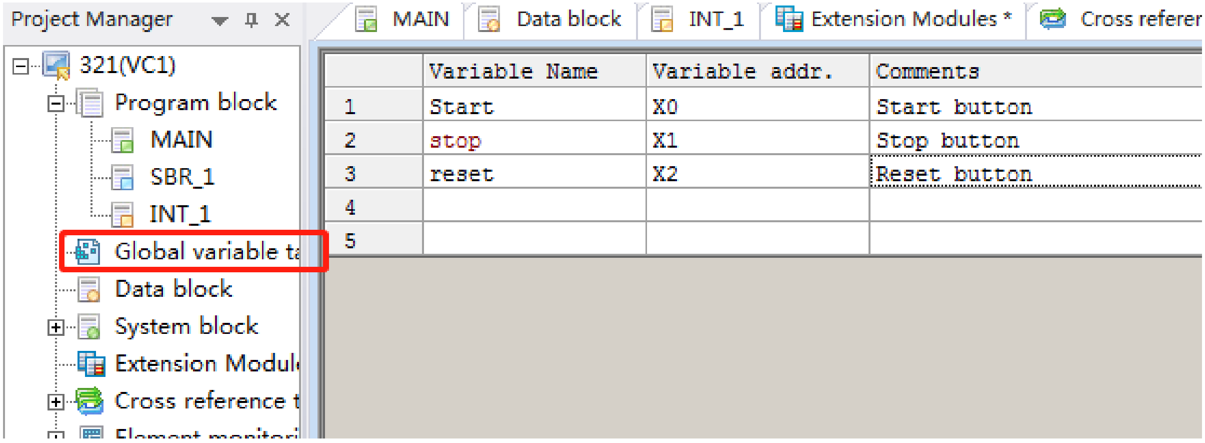
For details about operation instructions of data blocks, refer to the *Auto Studio Programming Software User Manual*.

2.2.3 Global variable table

Global variable is a meaningful symbol name defined for a certain PLC address, and it can be visited within the whole project range, which is equivalent to using the corresponding element of the variable. Global variable is defined in the global variable table. Global variable includes three attributes, namely **Variable name**, **Variable address**, and **Comments**.

The definition rule of global variable is: comprised of A–Z, a–z, 0–9, underline, and Chinese characters, and the variable name cannot start with numbers nor can it be independent numbers. The name can be capitalized or lower-case, and the length cannot exceed 8 bytes. No element type letters+numbers can be adopted as the name of the program and variable. There should be neither blank space nor any name which is the same with keywords. The reserved keywords include basic data type name, instruction name, and operation symbol in instruction list language.

VC1 series micro-PLCs are allowed to be downloaded no more than 140 global variables respectively (according to the maximum comment amount). Otherwise, it can only be saved locally but cannot be downloaded. Figure 2-16 shows the global variable table.



	Variable Name	Variable addr.	Comments
1	Start	X0	Start button
2	stop	X1	Stop button
3	reset	X2	Reset button
4			
5			

Figure 2-16 Global variable table

2.3 Running mode and state control

You can start or stop PLCs in any of the following three ways.

1. Using the mode selection switch.

2. Setting the startup mode of input points and external terminals in the system blocks, and controlled by the specified terminals.

3. Using the programming software to run or stop the PLC if the mode selection switch is set to ON.

2.3.1 Concepts of system run and stop states

The working states of main modulesinclude run and stop states.

- RUN
  When a main module is in Run state, a PLC executes the user program. That is to say, a scan cycle includes four tasks, namely executing the user program, communication, housekeeping, and refreshing I/O.
- STOP
  When a main module is in Stop state, a PLC does not execute the user program, but executes other three tasks in each scan cycle, namely communication, housekeeping, and refreshing I/O.

2.3.2 Run and stop state

- How to change from STOP to RUN
  1. Resetting the PLC

If the mode selection switch is set to ON and the PLC is reset (including system power-on reset), the system enters the RUN state automatically.

Note

If the system configuration item of the **Input point control mode** in the main module is valid, the state of the corresponding input terminal is ON. Otherwise, the system cannot enter the run state after reset.

---

2. Setting the mode selection switch manually

The system is changed from STOP to RUN state when you set the mode selection switch from OFF to ON.

3. Startup mode of input points

When system configuration item of **Input point startup mode** in system block tab is valid, the main module is changed from STOP to RUN once the system detects that the specified input points (X0 to X17) change from OFF to ON.

Note

When you select the input point control mode, a mode selection switch should be set to ON at the same time, otherwise the PLC cannot enter the run state.

- How to change from RUN to STOP

1. Resetting the PLC

If the mode selection switch is set to OFF, reset the PLC (including power-on reset),and the system enters RUN state automatically.

Note

When the mode selection switch is set to ON, the system can also enter the stop state after reset if the system configuration item of the **Input point control mode** is valid and the designated input point is in OFF.

---

2. Setting the mode selection switch manually

The system is change from RUN to STOP state when you set the mode selection switch from ON to OFF.

3. Using the instruction control mode

The system changes from RUN to STOP after executing the STOP instruction in the user program.

4. Auto-stop upon faults

The system stops the execution of the user program when detecting that a serious error occurs (such as user program errors oruser program execution timeout).

2.3.3 Setting the state of output pointsin the stop state

You can set the output state of the output point (Y) when the PLC is stopped. There are three modes for your choice:

Disable the output mode –When the PLC is stopped, all output points are OFF.

Freeze the output mode— When the PLC is stopped, all output points are frozen at the last state.

Configuration output mode— You can set the state of output points as needed when the PLC is stopped.

You can operate the above settings in the Output Table of the system block tab. For details, refer to the output table settings in section 2.2.1 "System block".

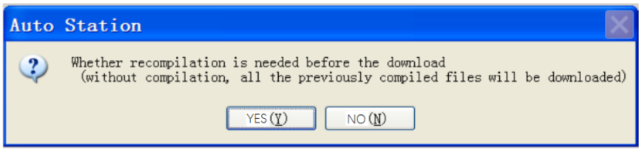
2.4 System commissioning

2.4.1 Program download and upload

- Download

  The download function is used to download a system block, data block, and user program generated by the Auto Studio software to a PLC through serial ports, and the PLC is required to stop running when downloading.

When downloading, if the program is changed after last compilation, you are informed whether you need to recompile the program, as shown in Figure 2-17.



Auto Station

Whether recompilation is needed before the download
(without compilation, all the previously compiled files will be downloaded)

Figure 2-17 Tips for recompiling the program

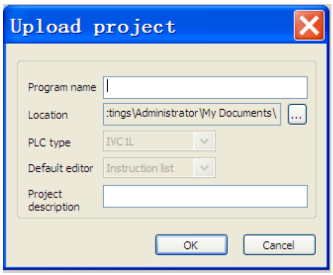
Note

Selecting **N** button means that you have no need to recompile it and the software adopts the previous edited results, but the running program downloaded to the PLC is different from the one displayed on the software interface.

When downloading a program, the software pops up a password window asking for entering a download password if it's configured with a download password and the downloading password has not been entered after starting the software. The download starts after a password entered is verified successfully, otherwise, the system prompts you to re-enter your password and click the Cancel button to exit the download.

- Upload

The upload function is used to upload a system block, data block, and user program in a PLC to PC through serial ports, and save them for new projects. When the battery backup data is valid, a file is selected and then the corresponding user assistance information file is bundled and uploaded. Figure 2-18 shows the upload dialog box.



Upload project

Program name

Location :tings\Administrator\My Documents\

PLC type IVC1L

Default editor Instruction list

Project description

Figure 2-18 Upload the dialog box

When uploading a program, you can upload the program directly if it's not configured with a download password. The software pops up a password window asking for entering an upload password if it's configured with upload password and an upload password has not been entered after starting the software. Upload starts after a password entered is correct, otherwise, the software prompts you and returns to the upload dialog interface.

If you select the **Disable Upload** function when downloading a program, a PLC cannot upload the program later, unless you enter a correct password to remove the Disable Upload function.

2.4.2 Error reporting mechanism

A system can detect and report two types of errors: system error and user program execution error.

A system error is caused by abnormal system operation.

A user program execution error is caused by the abnormal execution of the user program.

All errors are numbered uniformly, and each error code represents an error. For details, refer to Appendix F"System error codes".

- The reporting mechanism for system error

When the system detects that a system error occurs, the system error code is written to a special data register SD3, and the special relay SM3 is set at the same time. You can obtain the system error information by assessing the error code stored in SD3.

If multiple system errors occur at the same time, the system only writes the code of the most serious error in SD3.

When a serious system error occurs, a user program stops, and a ERR indicator on the main module turns on for a long time.

- The reporting mechanism for user program execution error

When a user program execution error occurs, the system sets a special relay SM20 and writes the current error code into a special data register SD20.

When the next application instruction is executed correctly, the SM20 is reset while SD20 still keeps the previous error code.

The system records the user program execution errors sequentially in a stack. Special data registers SD20 to SD24 form an error stack that records error codes of the latest five user program execution errors.

When a user program execution error occurs and the current error code is different from one stored in SD20, the error code is added to the error stack. Figure 2-19 shows the stack push process of error codes when a user program execution error occurs.

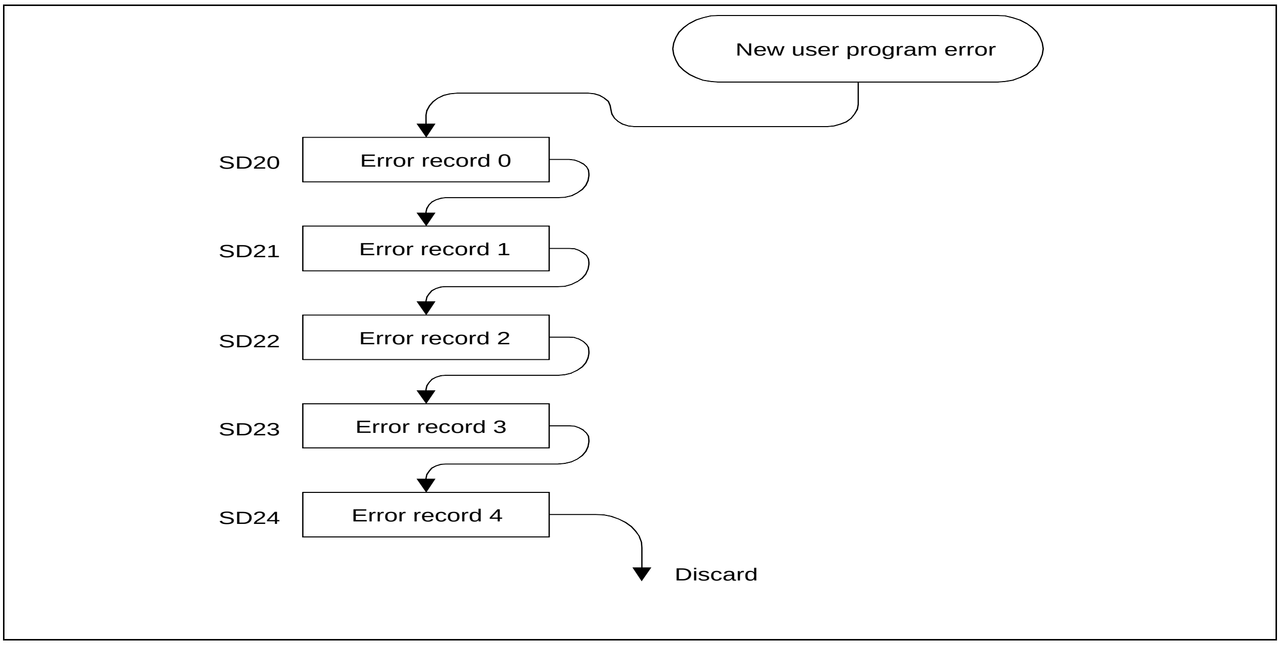


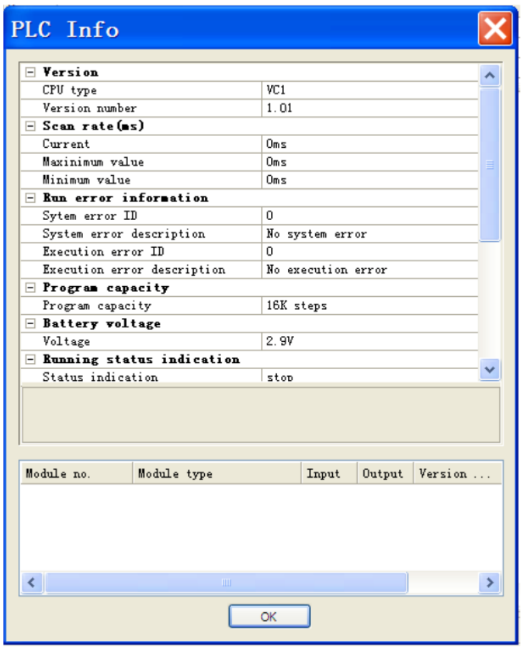
Figure 2-19 Push stack process of error codes

When a serious user program execution error occur, a user program stops and a ERR indicator on the main module turns on for a long time. In less serious cases, a ERR indicator on the main module does not turn on.

- Checking the error information online

Connecting the PLC with your PC through a serial port, and you can read various PLC state information through Auto Studio, including codes and descriptions of the above-mentioned system errors and user program execution errors.

In the main interface of Auto Studio, you can click **PLC**->**PLC Info** item to check the PLC information, as shown in Figure 2-20.



Version	
CPU type	VC1
Version number	1.01
Scan rate(ms)	
Current	0ms
Maximinum value	0ms
Minimum value	0ms
Run error information	
Sytem error ID	0
System error description	No system error
Execution error ID	0
Execution error description	No execution error
Program capacity	
Program capacity	16K steps
Battery voltage	
Voltage	2.9V
Running status indication	
Status indication	stop

Module no.	Module type	Input	Output	Version ...

Figure 2-20 PLC information

The **System error ID** is the code of the system error stored in SD3 while the **Execution error ID** is the code of the user program execution error stored in SD20. The relevant error descriptions displayed by both are for your reference.

2.4.3 Modifying the user program online

You can use the online modification function when you need to modify the user program without stopping the PLC.

🕮 Warning

On the occasions that casualties or property loss may occur, the online program modification function should be used by professionals with sufficient protection measures.

- Operation method

  After making sure that PC-PLC communication has been set up successfully and the PLC is in the run state, you can click the **Debug** ->**Online modification** in the Auto Studio main interface to enter the online modification state.

  In the online modification state, you can modify the main program, subprogram and interrupt subprogram as usual. After the modification, you click **PLC**->**Download** menu,and the modified program is compiled and downloaded to the PLC automatically. When the download completes, the PLC executes the new downloaded program.

- Limits

  1. In the online modification state, the global variable table and local variable table of any program cannot be modified, nor can add or delete any subprogram or interruption subprogram.

  2. When a program is in the online modification state, Auto Studio automatically exits the online modification state if a PLC is stopped.

2.4.4 Clearing and formatting

The clearing operations include PLC element value clear, PLC program clear, and PLC data block clear.

Formatting is to clear all data and programs in PLCs.

- PLC element value clear

  The PLC element value clear function can clear all element values in the PLC. Element values should be cleared when the PLC is in the stop state.

  Clearing the element values in the PLC causes the PLC to operate improperly or lose the intermediate working data, so you need to use this function with caution. During the operation process, the software displays a confirmation window for you to choose whether to continue or cancel the current operation.

- PLC program clear

  The PLC program clear function can clear the user programs in the PLC. User programs should be cleared when the PLC is in the stop state.

  Clearing the user programs in the PLC leads to no-execution of the user programs by PLC, so you need to use this function with caution. During the operation process, the software displays a confirmation window for you to choose whether to continue or cancel the current operation.

- PLC data block clear

  The PLC data block clear function can clear all the data block setups in the PLC. Data blocks should be cleared when the PLC is in the stop state.

  Clearing the data blocks in the PLC causes the PLC to stop using the preset value of the data block for initializing D elements, so you need to use this function with caution. During the operation process, the software displays a confirmation window for you to choose whether to continue or cancel the current operation.

- PLC formatting

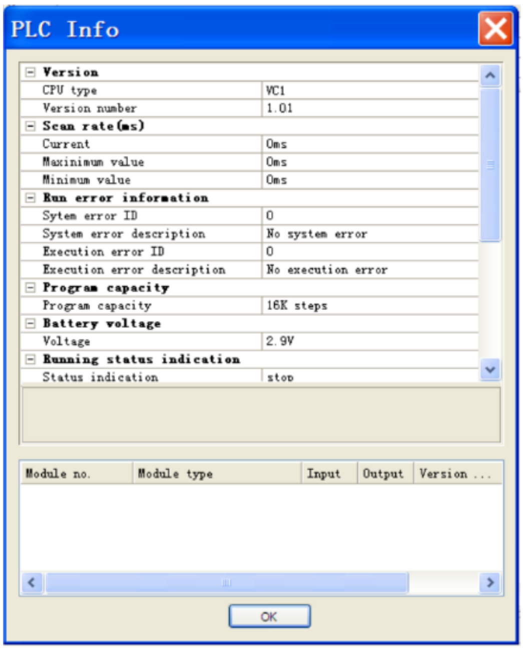
  The PLC formatting function can format all the PLC data, including clearing the user program, restoring the default configuration, and clearing data blocks. Data blocks should be cleared when the PLC is in the stop state.

  This operation clears all the downloaded and setup data, so you need to use this function with caution. During the operation process, the software displays a confirmation window for you to choose whether to continue or cancel the current operation.

2.4.5 Checking PLC information online

- PLC information

The PLC information function can be used to obtain and display various operation data and important information of PLCs. You can see the important information about the current operation of the PLC on the information display window, as shown in Figure 2-21.



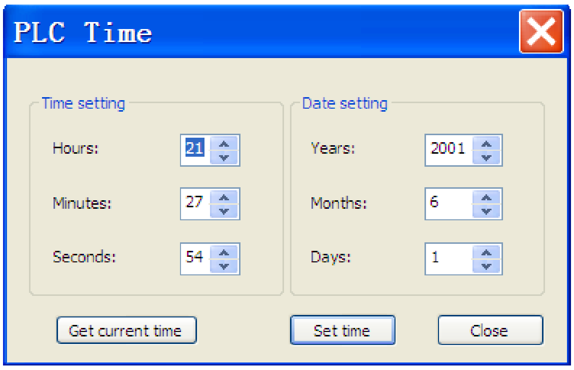
Version	
CPU type	VC1
Version number	1.01
Scan rate(ms)	
Current	0ms
Maximinum value	0ms
Minimum value	0ms
Run error information	
Sytem error ID	0
System error description	No system error
Execution error ID	0
Execution error description	No execution error
Program capacity	
Program capacity	16K steps
Battery voltage	
Voltage	2.9V
Running status indication	
Status indication	stop

Module no.	Module type	Input	Output	Version ...

Figure 2-21 Current operation information of the PLC

- PLC time

The PLC time function can be used to display and set the current time of PLCs, as shown in Figure 2-22.



Hours: 21
Minutes: 27
Seconds: 54
Years: 2001
Months: 6
Days: 1

Figure 2-22 Setting the PLC time

The window displays the current date and time from the PLC. You can adjust the time setting and click the **Set Time** button to set a new time to the PLC.

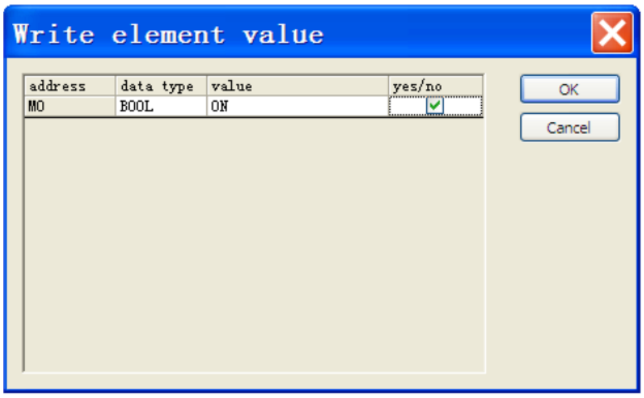
2.4.6 Element value writing and forcing, and element monitoring table

- Write and forced values of the elements

During the commissioning process, you can manually modify the value of certain soft elements to achieve some conditions, the element value write and force offer you this function. Difference between the write and force is that: the write element value is valid for once, and the value being written may change with the running of the program, but the forced element value is recorded in the PLC hardware until being cancelled by you.

To execute the write or force function, you need to select the elements to be written or forced firstly, and right click the menu to select **Write** or **Force**. Then a corresponding dialog box pops up and lists all the soft element addresses being referred to by the selected elements. You can selectively write or force certain soft element values, after confirmation, these values are sent to the PLC hardware. When these values take effect in the hardware, you can see the change results during the subsequent commissioning process.

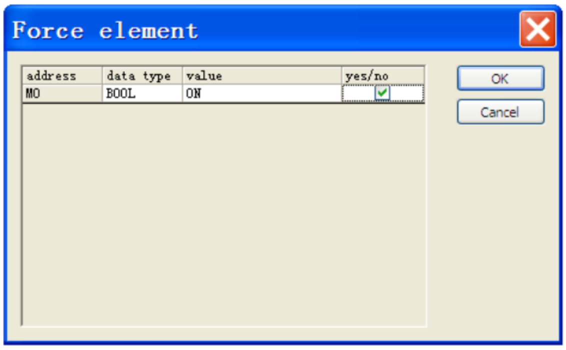
Figure 2-23 shows the dialog box for writing an element value.



address	data type	value	yes/no
M0	BOOL	ON	☑

Figure 2-23 Dialog box for writing an element value

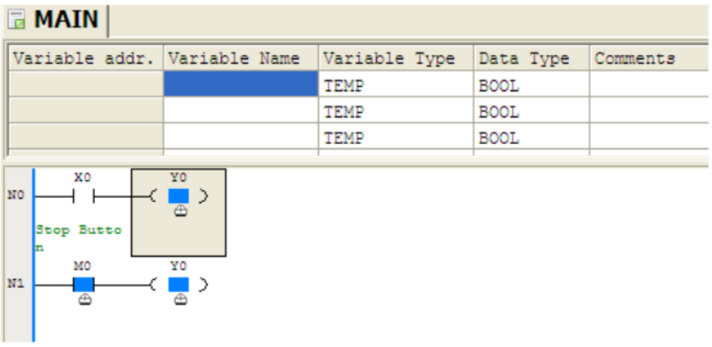
Figure 2-24 shows the dialog box for forcing an element value.



address	data type	value	yes/no
M0	BOOL	ON	☑

Figure 2-24 Dialog box for forcing an element value

The forced soft element carries a lock mark in LAD, as shown in Figure 2-25:

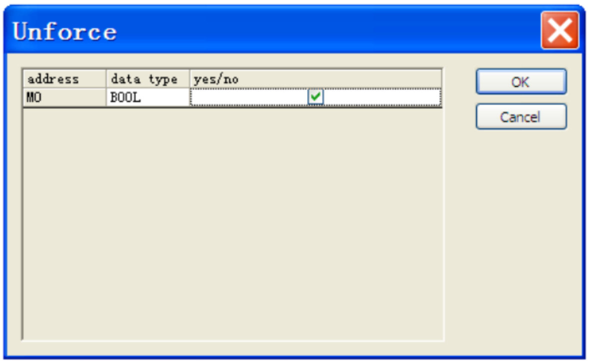


Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	
		TEMP	BOOL	
		TEMP	BOOL	

Figure 2-25 Locks mark of the forced soft element

- Element value unforcing

For element values that no long need to be forced, you can unforcing them. When you remove the force function, you need to first select target elements, right click and select **Unforce**, and then a dialog box pops up and displays all forced soft elements. You can selectively unforce certain forced soft elements, click OK button to confirm, and then these forced values are deleted from the PLC, so are the corresponding lock marks. Figure 2-26 shows the dialog box of unforce.



address	data type	yes/no
M0	BOOL	☑

Figure 2-26 Unforce

- Element monitoring table

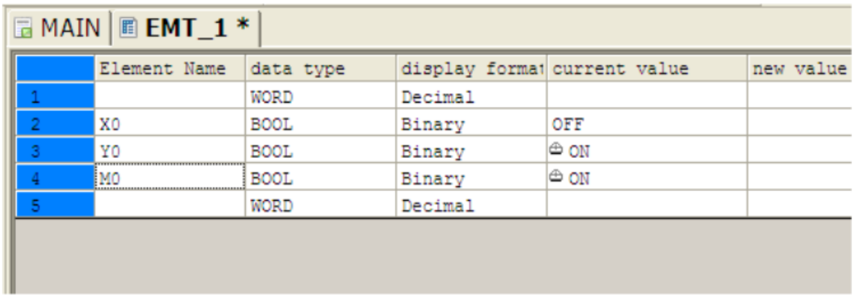
The element monitoring table provides element value monitoring function during commissioning. Program input and output elements, registers, and word elements can be added to the element monitoring table so that they can be tracked after the program is downloaded to PLCs.

The element monitoring table works in two modes: editing and monitoring modes. In the editing mode, all editing functions can be carried out but no monitoring function can be carried out. In the monitoring mode, both of the monitoring and editing functions can be carried out.

In the monitoring mode, displayed element values are refreshed automatically, and then modified or forced element values can be updated timely.

The element state monitoring table provides the functions includes editing, sequencing, searching, auto-refresh and display of the current values of designated elements, writing element values, forcing element/variable values, and unforcing values.

Figure 2-27 shows the element monitoring table.



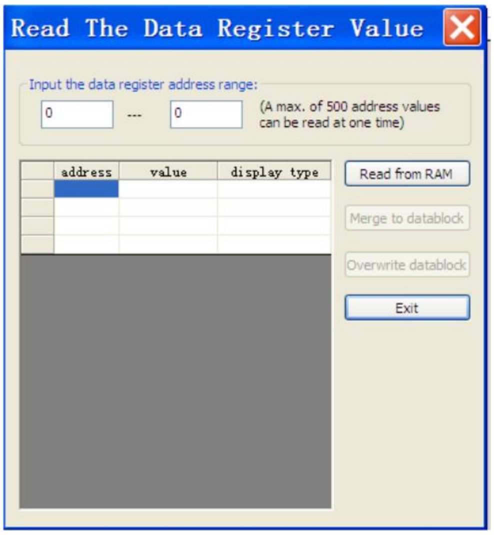
	Element Name	data type	display format	current value	new value
1		WORD	Decimal		
2	X0	BOOL	Binary	OFF	
3	Y0	BOOL	Binary	🔒 ON	
4	M0	BOOL	Binary	🔒 ON	
5		WORD	Decimal		

Figure 2-27 Element state monitoring table

2.4.7 Generating data blocks from RAM

The data values of 500 D registers (at most) read from PLC consecutively are displayed, and the results can be merged to the data block or overlay the original data block.

Open the window for generating data blocks from RAM, as shown in Figure 2-28.


Read The Data Register Value
Input the data register address range:
0

0
(A max. of 500 address values can be read at one time)
address
value
display type
Read from RAM
Merge to datablock
Overwrite datablock
Exit

Figure 2-27Window for generating data blocks from RAM

Entering the range of data blocks to be read, click the **Read from RAM** button, and data is read into the list after the instruction is correctly executed.

You can select hex, decimal, octal, or binary in the field of **display typ**e to display the data.

After reading the data successfully, the **Merge to datablock** and **Overwrite datablock** button become available. Click the **Merge to datablock** button to add the generated results to the rear part of the current data blocks and click the **Overwrite datablock** button to replace the existed contents in the data block with the generated results. After exiting the register value reading window, the software prompts that the data block has been changed and the data block window is open automatically.

Chapter 3 Soft element and data

This chapter detailedly describes the definition, classification and functions of the soft elements of VC series PLCs.

Chapter 3 Soft element and data 34
- 3.1 Type and function of soft elements 35
  - 3.1.1 Summary of soft elements 35
  - 3.1.2 Soft elementlist 35
  - 3.1.3 Input and output points 37
  - 3.1.4 Auxiliary Relays 37
  - 3.1.5 State relay 38
  - 3.1.6 Timer 38
  - 3.1.7 Counter 39
  - 3.1.8 Data register 39
  - 3.1.9 Special auxiliary relay 40
  - 3.1.10 Special data register 40
  - 3.1.11 Indexing register 41
  - 3.1.12 Local auxiliary relay 41
  - 3.1.13 Local data register 41
  - 3.1.14 Bit-string combined addressing mode (Kn addressing mode) 41
  - 3.1.15 Indexing mode (Z addressing mode) 42
  - 3.1.16 Bit-string combined indexing mode 43
  - 3.1.17 Storing and addressing 32-bit data in D, R, and V elements 44
- 3.2 Data 44
  - 3.2.1 Data type 44
  - 3.2.2 Correlation between elements and data types 44
  - 3.2.3 Constant 45

3.1 Type and function of soft elements

3.1.1 Summary of soft elements

PLCs are configured with a variety of virtual elements in the system design to replace the real generic relays, time relays, and other devices in the relay control circuits.These virtual elements are collectively referred to as soft elements. PLCs adopt soft elements for program operation and system function configuration to implement all operation and control functions. Due to their virtual nature, the soft elements can be used repeatedly in the program without quantity limit the oretically (actually related to the program capacity). Besides, the soft elements have no mechanical or electric problems of the real devices. Such features make PLCs much more reliable than relay control circuits. In addition, it is easier to program and modify the logic.

The types and functions of VC series PLC soft elements are shown in the following figure.

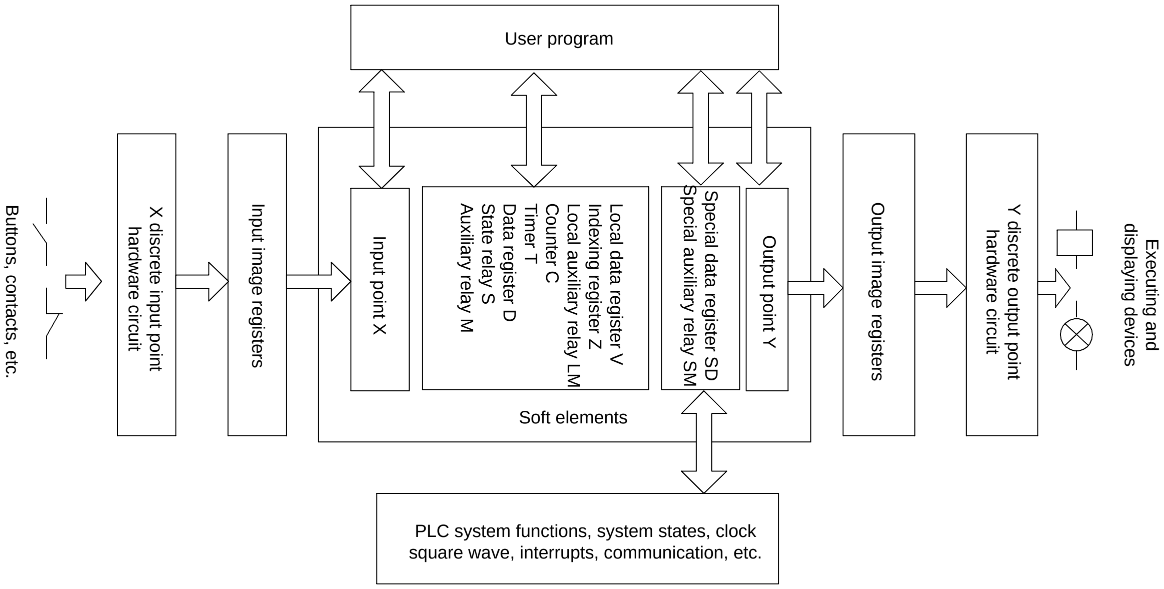

User program
Buttons, contacts, etc.
X discrete input point hardware circuit
Input image registers
Input point X
Local data register V
Indexing register Z
Local auxiliary relay LM
Counter C
Timer T
Data register D
State relay S
Auxiliary relay M
Special data register SD
Special auxiliary relay SM
Output point Y
Soft elements
Output image registers
Y discrete output point hardware circuit
Executing and displaying devices
PLC system functions, system states, clock square wave, interrupts, communication, etc.

Figure 3-1 Types and functions of soft elements

In this manual, soft elements are named according to their types. For example:

- Input point X, or "X element" for short
- Output point Y, or "Y element" for short
- Auxiliary relay M, or "M element" for short
- Data register D, or "D element" for short
- State relay S, or "S element" for short

3.1.2 Soft elementlist

The soft elements of VC series micro-PLCs are compiled and classified according to their functions. Different elements execute different functions and are easily accessible.

Figure 3-1 shows the list of VC series micro-PLC soft elements.

Figure 3-1 VC series PLC soft element list

		VC1S	VC1	VC2(Pianning)	VC3(Pianning)	VC5(Pianning)	Numbered in
Soft element resources note 4	I/O points	36 inputs 24 outputs (inputs:X0–X47 outputs:Y0–Y27) note 1	128 inputs 128 outputs (inputs:X0–X177; outputs:Y0–Y177) note 1	256 inputs 256 outputs (inputs:X0–X377; outputs:Y0–Y377) note 1	512 inputs 512 outputs (inputs:X0–X777; outputs:Y0–Y777)$^{\text{note 1}}$	512 inputs 512 outputs (inputs:X0–X777; outputs:Y0–Y777)$^{\text{note 1}}$	Octal

		VC1S	VC1	VC2(Pianning)	VC3(Pianning)	VC5(Pianning)	Numbered in
	Auxiliary relay	2048 (M0–M2047)	2048 (M0–M2047	2048 (M0–M2047)	10240 (M0–M10239)	10240 (M0–M10239)	Decimal
	Local auxiliary relay[note 5]	64 (LM0–LM63)	64 (LM0–LM63)	64 (LM0–LM63)	64 (LM0–LM63)	64 (LM0–LM63)	Decimal
	Special auxiliary relay	512 (SM0–SM511)	512 (SM0–SM511)	1024 (SM0–SM1023)	1024 (SM0–SM1023)	1024 (SM0–SM1023)	Decimal
	State relay	1024 (S0–S1023)	1024 (S0–S1023)	1024 (S0–S1023)	4096 (S0–S4095)	4096 (S0–S4095)	Decimal
	Timer	256 (T0–T255)[note 2]	256 (T0–T255)[note 2]	256 (T0–T255)[note 2]	512 (T0–T511)[note 2]	512 (T0–T511)[note 2]	Decimal
	Counter	256 (C0–C255)[note 3]	264 (C0–C263)[note 3]	256 (C0–C255)[note 3]	307 (C0–C307)[note 3]	307 (C0–C307)[note 3]	Decimal
	Data register	8000(D0–D7999)	8000 (D0–D7999)	8000 (D0–D7999)	8000 (D0–D7999)	8000 (D0–D7999)	Decimal
	Data register R	/	/	/	32768 (R0–R32767)	32768 (R0–R32767)	Decimal
	Local data register[note 5]	64 (V0–V63)	64 (V0–V63)	64 (V0–V63)	64 (V0–V63)	64 (V0–V63)	Decimal
	Indexing register	16 (Z0–Z15)	16 (Z0–Z15)	16 (Z0–Z15)	16 (Z0–Z15)	16 (Z0–Z15)	Decimal
	Special data register	512 (SD0–SD511)	512 (SD0–SD511)	1024 (SD0–SD1023)	1024 (SD0–SD1024)	1024 (SD0–SD1024)	Decimal

Notes:

Note 1: The addresses of X and Y elements are numbered in octal, for example, address X10 represents the 8th input point. The max. value of I/O points here is the system capacity. The actual extendable hardware points should depend on the PLC system configuration (including the available extension module types and points, power capacity limit, etc.).

Note 2: T element address is classified into three categories based on timing precision:

- 100 ms precision: T0–T209
- 10 ms precision: T210–T251
- 1 ms precision: T252–T255

Note 3: C element address is classified into three categories based on the width and function of count values:

- 16-bit increment counter: C0–C199
- 32-bit increment and decrement counter: C200–C235
- 32-bit high-speed counter: C236–C263

Note 4: Some internal soft element resources of PLCs have been reserved for internal use only. You should not use these elements on the user program, if possible. For details, refer to Appendix C "Reserved elements".

Note 5: These two types of soft elements are local variables which cannot be defined in the global variable table. When the user program calls subprograms or returns to the main program, they are rest to zero obtain the parameter values or states according to the interface parameter transfer function.

3.1.3 Input and output points

- Element mnemonics
    - X elements (discrete input points)
    - Y elements(discrete output points)
- Function

  The X and Y elements represent respectively the input state of the hardware X terminal and output state of hardware Y terminal.

  The state of X elements is obtained through the inputimage register, while the state of Y elements is output through the output circuit driven by the output image register. The two operations are carried out in the I/O refresh phase of PLC scan cycle mode, as shown in Figure3-2. For details, refer tosection 2.1.2 "PLC operating mechanism (Scan cycle model)". It is obvious that thereis a brief delay in PLC's response to the I/O. The delayis related to the input filter, communication, housekeeping, and scan cycle.

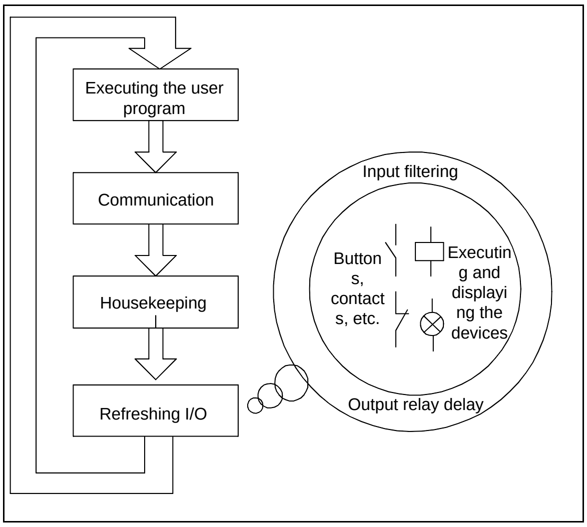


Figure 3-2 Schematic diagram of I/O refresh

- Classification

  Output channel corresponding to X elements: X0 to X17 are configured withthe digital filtering function. You can set the filtering time through the system block. Other X input points use the hardware filtering. X0 to X7 can be used as counting input points for soft elements of high-speed counters. Besides, X0 to X7 can also be used as input terminals for external interrupt, pulse capture, and the SPD instruction.

  Y elements are used as high-speed output terminals or general output terminals.
- Elements addressing mode

  Octal,started with 0. X and Y elements of the main module and I/O extension modules are addressed contiguously. X elements are addressed contiguously in X0﹣X7, X10﹣X17, X20﹣X27, and so on. Y elements are addressed contiguously in Y0–Y7, Y10–Y17, Y20–Y27, and so on.
- Data type

  Both the X and Y elements are Boolean (element values are ON or OFF).
- Available forms

  You can adopt NO and NC contacts of X elements during programming (depending on two instructions). NO and NC contacts have opposite state values.They are sometimes referred to as "a contact" and "bcontact".

  You can also adopt NO and NC contacts of Y elements during programming.
- Value assignment

  1. X elements accept only hardware input state and forced operation state values which cannot be changed through output or setting instructions, nor be set during the system commissioning.

  2. You can assign state values to Y elements with the OUTinstruction, set the state values of Y elements, oreven force or write state values of Y elements during the system commissioning.

  3. Through the system block, you can set the output states of Y elements in STOP state.

3.1.4 Auxiliary Relays

- Element mnemonic

  M elements
- Function

  The system provides a type of discrete state elements to you, which is similar to an intermediate relay in a real electrical control circuit. You can use them to store various intermediate states in the user program.
- Elements addressing mode

  Octal, started with 0.
- Data type

M elements are Boolean (the element values are ON or OFF).

- Available forms
  NO and NC contacts
- Value assignment
  1. Through the instruction operations. 2. Force or write the state values during the system commissioning.
- Saving dataat power outage

State	M elements for which data is saved at power outage	M elements for which data is not saved at power outage
Power outage	Data saved	Date deleted
RUN → STOP	Data saved	Date deleted
STOP → RUN	Unchanged	Cleared
Note: The storage range at power outage is set through the system block. For details, refer to section 2.2.1 "System block".		

Note

When using the N:N protocol, some M elementsare called by the system. You need to pay attention to it when programming and modifying the program.

3.1.5 State relay

- Element mnemonic
  S elements
- Alias
  Step flag
- Function
  As the step flag, S elements are used in the SFC. For details, refer to Chapter 7 "SFC user guide".
- Classification
  S0–S19: initial step flag
  Others: general step flag
- Elements addressing mode
  Decimal, starting with 0
- Data type
  Boolean elements (element values are ON or OFF).
- Available forms
  1. Representing the step state (when the STL instruction is programmed through SFC)
  2. NO and NC contacts (when the STL instruction is not programmed for SFC). Similar to characteristics of M elements, NO and NC contacts of S elements can be used for programming.
- Value assignment
  1. Through the instruction operations. 2. Force or write the state values during the system commissioning.
- Uninterrupted output after power outage

State	S elements for which data is saved at power outage	S elements for which data is not saved at power outage
Power outage	Unchanged	Cleared
RUN → STOP	Unchanged	Unchanged
STOP → RUN	Unchanged	Cleared
Note: The storage range at power outage is set through the system block. For details, refer to 2.2.1 "System block".		

3.1.6 Timer

- Element mnemonic
  T elements
- Function
  T element is a composite soft element that contains a word element (2 bytes) and a bit element. The T word element records a 16-bit timing value that can be used as a value in the program. The T bit element reflects the state of the timer coil, which can be used for logic control.

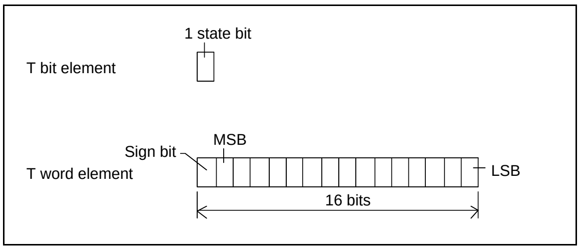


Figure 3-3 T elements

- Classification
  According to the timing precision, T elements are classified into three types.The following table describes the T elements and corresponding timing precisions in different address segments.You need to pay attention to it when using.

T element	Timing precision
T0–T209	100 ms precision
T210–T251	10 ms precision
T252–T255	1 ms precision

For T elements with timing precision of 1 ms, they areactivated by interrupt trigger and unrelated to the PLC scancycle. Therefore, their action time is the most accurate. For T elements with timing precision of 10 ms and 100 ms, refresh and action time of their timing values are related to the PLC scan cycle.

- Elements addressing mode
  Decimal, starting with 0
- Data type
  Boolean (element values are ON or OFF), and word.

- Available forms
  The timing and action modes of T elements are determined by timing instructions. There are four types of timing instructions, including TON, TOF, TONR, and TMON instructions. For details, refer to Chapter 5 "Basic instructions".
- Value assignment
  1. Through the instruction operations. 2. Force or write the state values during the system commissioning.
- Uninterrupted output after power outage

State	T elements for which data is saved at power outage (for VC2/3/5series only)	T elements for which data is not saved at power outage
Power outage	Unchanged	Cleared
RUN → STOP	Unchanged	Unchanged
STOP → RUN	Unchanged	Cleared
Note: The storage range at power outage is set through the system block. For details, refer to section 2.2.1 "System block".		

Note

The maximum timing value of T element is 32767. The preset value is -32768 to -32767. Because T element acts only when the count value reaches or exceeds the preset value. Therefore, it is actually meaning less to set the preset value as a negative number.

3.1.7 Counter

- Element mnemonic
  C elements
- Function
  A C element is a composite soft element that contains a bit element, and a word or adouble word element (2 or 4 bytes). The C word element records a 16-bit or 32-bit timing value that can be used as a value in the program while C bit element reflects the timer coil state for logic control.

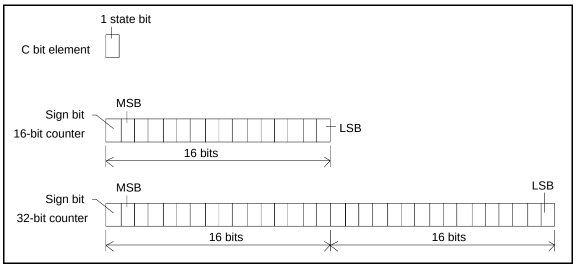


Figure 3-4 C elements

- Classification
  Two types: 16-bit counter and 32-bit counter
- Elements addressing mode
  Decimal, starting with 0
- Data type
  Boolean(element values are ON or OFF), single word or double word.
- Available forms
  The counting instructions that call C elements are classified into 4 types, namely 16-bit increment counting instruction, 16-bit cycle counting instruction, 32-bit increment and decrement counting instruction, and high-speed I/O instruction. For details, refer to Chapter 5 "Basic instructions" and Chapter 6 "Application instructions".
  The classification of C elements is shown in the table below:

C elements	Counting function	Applicable instruction type
C0–C199	16-bit increment counter	16-bit increment counting instruction 16-bit cyclic counting instruction
C200–C235	32-bit increment and decrement counter	32-bit increment and decrement counting instruction
C236–C263	32-bit high-speed counter	High-speed I/O instruction

- Value assignment
  1. Through the instruction operations. 2. Force or write the state values during the system commissioning.
- Uninterrupted output after power outage

State	C elements for which data is saved at power outage	C elements for which data is not saved at power outage
Power outage	Unchanged	Cleared
RUN → STOP	Unchanged	Unchanged
STOP → RUN	Unchanged	Cleared
Note: The storage range at power outage is set through the system block. For details, refer to section 2.2.1 "System block".		

3.1.8 Data register

- Element mnemonic
  D and R elements
- Function
  As data elements, D or R elements are used as operandsin manycalculation and control instructions.
- Elements addressing mode
  Decimal, starting with 0
- Data type
  Each D or R element is a 16-bit register that can store 16-bit data, such as 16-bit integer.

Two D or R elements can form a double-word element that can store32-bit data, such as long integer data or floating-point data.

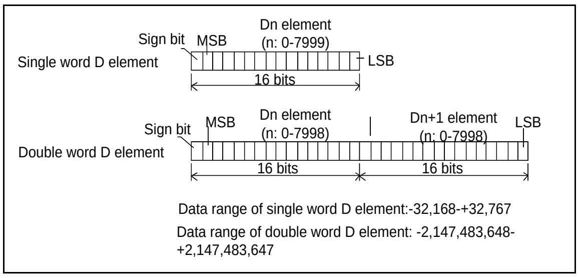


Figure 3-5 D or R elements

Note

In a double-word D or R element, the higher 16-bit is in the first D or R element, and the lower 16-bit is in the second D or R element.

- Available forms
  D or R elements are used as operands in many calculation and control instructions.
- Value assignment
  1. Data block initialization. 2. Through the instruction operations. 3. Force or write the state values during the system commissioning.
- Uninterrupted output after power outage

State	D elements for which data is saved at power outage	D elements for which data is not saved at power outage
Power outage	Unchanged	Cleared
RUN → STOP	Unchanged	Unchanged
STOP → RUN	Unchanged	Cleared
Note: The storage range at power outage is set through the system block. For details, refer to section 2.2.1 "System block". *R elements cannot be saved at power outage*		

Note

Some D elements are called by the system when inverter instruction or N:N protocol is adopted. You need to pay attention to it when programming and modifying the program.

3.1.9 Special auxiliary relay

- Element mnemonic
  SMelements
- Function
  SM elements are closely related to the PLCsystem function, reflectingthe system function and state of the PLCs. For details, refer to Appendix A "Specialauxiliary relay" in this manual.
- Classification
  The frequently used SM elements include:
  - SM0: PLC operation monitor bit. It is ON whenthe PLC is in RUN state.
  - SM1: initial operation pulse bit. It is ON in the firstscan cycle of PLC operation.
  - SM3: system error. It is ON if any system error is detected after the PLC is powered on or when the PLC changes from STOP to RUN.
  - SM10–SM12: respectively the clock square-wave cycled at 10 ms, 100 ms, and 1 s (flipping-overtwice in a cycle).
  - SM25﹣SM71: Interrupt control flag bit. Setting these SM elements enables the corresponding interrupt functions.
- Elements addressing mode
  Decimal, started with 0.
- Data type
  Boolean elements (element values are ON or OFF).
- Available forms
  NO and NC contacts
- Value assignment
  1. Through the instruction operations. 2. Force or write the state values during the system commissioning.

Note

You cannot assign values to read-only SD elements.

3.1.10 Special data register

- Element mnemonic
  SD elements
- Function
  SD elements are closely related to the PLC system function, reflecting the system function parameters, state code values, and instruction operation data of PLCs. For details, refer to Appendix B "Special data register" in this manual.
- Elements addressing mode
  Decimal, started with 0.
- Data type
  Word, and doubleword (integer)elements.
- Available forms
  Storage and calculation of integers
- Value assignment

1. Through the instruction operations. 2. Force or write state values during system commissioning.

Note

You cannot assign values to read-only SD elements.

3.1.11 Indexing register

- Element mnemonic
  Z elements
- Function
  16-bit register elements can be used to store symbolic integer data.For details about indexing, refer to section 3.1.15 "Indexing mode (Z addressing Mode)".
- Elements addressing mode
  Decimal, started with 0.
- Data type
  Word elements
- Available forms
  Z elements are used for indexing. You need to write the address offset in Z elements before using them.
- Value assignment
  1. Through the instruction operations. 2. Force or write state values during system commissioning.

3.1.12 Local auxiliary relay

- Element mnemonic
  LM elements
- Function
  LM elements are local variants that can be applied in the main program and subprograms. They are variable elements that are locally valid in each individual program body (main program, subprogram, and interrupt program). Therefore, it is not possible to directly share the state of any LM element between different program bodies. When the system leaves a program body in the execution of the user program, the LM element is redefined. When returning to the main program or calling a subroutine, the value of the redefined LM element is deleted or the corresponding state needs to be obtained according to the interface parameter transfer function.
  LM elements can be used to define the interface parameters of subprograms to realize the interface parameter transfer function. For details, refer to section 4.4 "Subprogram".
- Elements addressing mode
  Decimal, started with 0.
- Data type
  Boolean(element values are ON or OFF).
- Available forms
  NO and NC contacts
- Value assignment
  1. Through the instruction operations.

3.1.13 Local data register

- Element mnemonic
  V elements
- Function
  V elements are local variants that can be applied in the main program and subprograms. They are variable elements that are locally valid in each individual program body (main program and subprogram). Therefore, it is not possible to directly share the data of any V element between different program bodies. When the system leaves a program body in the execution of the user program, the V element is redefined. When returning to the main program or calling a subroutine, the value of the redefined V element is deleted or the corresponding data needs to be obtained according to the interface parameter transfer function.
  V elements can be used to define the interface parameters of subprograms to realize interface parameter transfer function. For details, refer to section 4.4 "Subprogram".
- Elements addressing mode
  Decimal, started with 0.
- Data type
  Boolean elements (element values are ON or OFF).
- Available forms
  Word element, which can be used to store numeric information
- Value assignment
  1. Through the instruction operations.

3.1.14 Bit-string combined addressing mode (Kn addressing mode)

- Concept
  The bit-string combined addressing mode,or Kn addressing mode, combines bit element strings into words or long words.

- Method

  The format is: "K (n) (U)", where the "n" is an integer among one to eight, indicating that the length of the bit string is n×4 bits. U indicates the start bit element address of the element string.

  Application instance:

  1. K1X0 indicates a word made up of a bit string of 4 bit elements (X0, X1, X2 and X3) starting from X0.

  2. K3Y0 indicates a word made up of a bit string of 12 bit elements (Y0, Y01, Y02 and Y03), (Y04, Y05, Y06 and Y07), and (Y10, Y11, Y12 and Y13) starting from Y0.

  3. K4M0 indicates a word made up of a bit string of 16 bit elements: M0, M1, M2, M3… and M15.

  4. K8M0 indicates a double word made up of a bit string of 32 bit elements: M0, M1, M2, M3… and M31.

- Data storage format in the Kn addressing mode

  The following example describes how data is stored in the Kn addressing mode:

  MOV 2#10001001 K2M0 (which is the equivalent of MOV 16#89 K2M0, or MOV 137 K2M0). After the instruction is executed, K2M0 is stored. The following table describes the specific storage format of K2M0.

Data	MSB	Intermediate bit						LSB
K2M0	M7	M6	M5	M4	M3	M2	M1	M0
16#89	1	0	0	0	1	0	0	1

- Note of the bit-string combined addressing mode

  If the destination operand of the instruction uses the Kn addressing mode, while the width of the data to be stored in the destination operand is greater than the width defined in the Kn addressing mode, the system keeps the LSB and discards the MSB of the data.

  For example:

  Executing the instruction "DBITS 16# FFFFFFF0 K1M0".

  After the instruction is executed, the operand 2 (K1M0) needs to store the calculation result 16#1c (28). However, the K1M0 is only 4-bit wide, which is not enough for 16#1c. By discarding the higher bits, the actual calculation result of operand 2 is K1M0=16#c (12).

3.1.15 Indexing mode (Z addressing mode)

- Concept of indexing

  VC series PLCs provides the indexing mode (Z addressing mode). You can use Z elements (indexing registers) to get indirect access to the target elements.

- Z addressing method:

  Target address=Basic element address+Address offset stored in Z element.

  For example:

  In the indexing mode, for D0Z0 (in which Z0=3), the target address is D3. D0 is the basic address, and the offset address stored in element Z0 is 3.

  Therefore, when Z0=3, the instruction "MOV 45 D0Z0" is equal to "MOV 45 D3" in effect, and in both cases D3 is assigned the value 45 after the instruction is executed.

- Indexing example

  1. Bit element indexing example

  LD M01

  MOV 6 Z1

  SFTR X0Z1 M0 8 2

  The preceding instructions are in effect equal to:

  LD M01

  SFTR X6 M0 8 2

  The addressing process is as follows:

Z1=6

X0Z1 = X (0＋Z1) = X6

2. Word element indexing example

LD M01

MOV 30 Z20

MOV D100Z20 D0

The preceding instructions are in effect equal to:

LD M01

MOV D130 D0

The addressing process is as follows:

Z20=30

D100 Z20 = D (100＋Z20)= D130

- Notes of the indexing mode

  1. In the indexing mode (Z addressing mode), Z elements store the address offset. They support signed integers, which means negative offset is supported.

  For example:

  MOV -30 Z20

  MOV D100Z20 D0

  The preceding instructions are in effect equal to:

  MOV D70 D0

  2. The SM and SD elements do not support the indexing mode.

  3. You need to pay attentrion to the address range when using the Z addressing mode. For example, D7999Z0 (Z0=9) is outside the address range of the D elements (The max. address of D elements is 7999.).

3.1.16 Bit-string combined indexing mode

The bit-string combined addressing mode can be used in conjunction with the indexing mode. For example: K1X0Z10.

In this mode, you need to determine the start bit element address through the Z addressing, and determine the length of the bit string through the Kn addressing.

For example:

LD M1

MOV 3 Z10

MOV K1X0Z10 D0

The preceding instructions are in effect equal to:

LD M1

MOV K1X3 D0

The addressing process is as follows:

Z10 = 3

K1X0Z10=K1X (0+Z10) = K1X3

3.1.17 Storing and addressing 32-bit data in D, R, and V elements

- Storing 32-bit data in D, R, and V elements

  The DINT, DINT, and REAL type data are 32-bit, anda D, R, or V element is only 16-bit. Two consecutive D, R, or V elements are needed to store the 32-bit data.

  VC series PLCs store the 32-bit data in the Big Endian mode, that is to say,the elements with small addressesare used to store the higher bits of the 32-bit data, while the elements with big addresses are used to store the lower bits of the 32-bit data.

  For Application instance, unsigned long integer "16# FEA8_67DA" is stored in the element (D0, D1). The actual storage format is as follows:

D0	0xFEA8
D1	0x 67DA

- Addressing 32-bit data in D, R, and V elements

  You can use a D or V element to locate a 16-bit data, such as an INT or WORD type data, or a 32-bit data, such as a DINT or DINT data.

  If a D, R, or V element address is used in an instruction operand, the operand data type determines whether the data is 16-bit or 32-bit.

  For example:

  In the instruction "MOV 16#34 D0", the address D0 indicates a single D0 element, because operand 2 of the MOV instruction is of the WORD data type.

  In the instruction "DMOV 16# FEA867DA D0", the address D0 indicates two consecutive words: D0 and D1,because operand 2 of the DMOV instruction is of the DINT data type.

3.2 Data

3.2.1 Data type

All instruction operands are of a certain data type. There are altogether six data types, as listed in the following table.

Table 3-2 Operand data types

Data type	Type description	Data width	Range
BOOL	Bit	1	ON, OFF (1, 0)
INT	Signed integer	16	－32768–+32767
DINT	Signed long integer	32	－2147483648–+2147483647
REAL	Floating-point number	32	±1.175494E-38–±3.402823E＋38

3.2.2 Correlation between elements and data types

The element types used by instruction operands needs to match their data types. The correlations between the applicable elements and data types are listed in the following table.

Table 3-3 Mapping between elements and data types

Data type	Soft elements													
BOOL										C	T			
	X	Y	M	S	LM	SM								
INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R
DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R
REAL	Constant							D				V		R

Data type	Soft elements													

If an instruction uses an operand with an unsuitable data type, the instruction is deemed illegal. For example,instruction "MOV 10 X0" is illegal because operand 2 of the MOV instruction is of signed integer type, while the X0 element can store only Boolean data.

Note

1. When the operand is of INT type, the applicable elements include KnX, KnY, KnM, KnS, KnLM, and KnSM, where 1 ≤ n ≤ 4.
2. When the operand is of DINT type, the applicable elements include KnX, KnY, KnM, KnS, KnLM, and KnSM, where 5 ≤ n ≤ 8.
3. When the operand is of INT type, the applicable C elements are C0–C199.
4. When the operand is of DINT type, the applicable C elements are C200–C263

3.2.3 Constant

You can use constants as the instruction operands. VC series PLCs support a variety of constant mode inputs. Expressions of constants are shown in the following table.

Table 3-4 Expressions of constants

Constant type	Application instance	Valid range	Remarks
Decimal constant (16-bit signed integer)	－8949	－32768–+32767	
Decimal constant (32-bit signed integer)	－2147483646	－2147483648–+2147483647	
Hex constant (16-bit)	16#1FE9	16#0–16#FFFF	Hex, octal, and binary constants have no positive or negative meaning. If you select hex, octal or binary constants used as operands, positive and negative natures of these constants, and their sizes are determined by data types of these operands.
Hex constant (32-bit)	16#FD1EAFE9	16#0–16#FFFFFFFF	
Octal constant (16-bit)	8#7173	8#0–8#177777	
Octal constant (32-bit)	8#71732	8#0–8#37777777777	
Binary constant (16-bit)	2#10111001	2#0–2#1111111111111111	
Binary constant (32-bit)	2#101110011111	2#0–2#1111111111111111 1111111111111111	
Single-precision floating-point constants	－3.1415E-16 3.1415E＋3 0.016	±1.175494E-38–±3.402823E＋38	Compliance with the IEEE-754 standard. The programming software can display and input floating-point constants with 7-bit effective precision.

Chapter 4 Programming concept

This chapter detailedly describes the contents of VC series micro-PLC programming, including the programming language and programelements. The programming and usage of subprograms arealso introduced, and finally, it describes some general explanation of instructions.

Chapter 4 Programming concept 46
- 4.1 Introduction to programming languages 47
  - 4.1.1 LAD 47
  - 4.1.2 IL 48
  - 4.1.3 SFC 48
- 4.2 Program Elements 49
  - 4.2.1 User Program 49
  - 4.2.2 System block 50
  - 4.2.3 Data block 50
- 4.3 Block comment and variable comment of the program 50
  - 4.3.1 Block comment 50
  - 4.3.2 Variable Comment 51
- 4.4 Subprogram 52
  - 4.4.1 Concept 52
  - 4.4.2 Notestouse SBRs 52
  - 4.4.3 Definition of the SBR variable table 53
  - 4.4.4 SBR parameter transfer 54
  - 4.4.5 SBR application instance 54
- 4.5 General information of the instructions 56
  - 4.5.1 Instruction operands 56
  - 4.5.2 Flag bit 56
  - 4.5.3 Restrictions on the use of instructions 56

4.1 Introduction to programming languages

There are three programming languages: LAD, IL, and SFC.

4.1.1 LAD

- Concepts

  LAD is a widely-used graphical PLC programming language similar to the electrical (relay) control diagram. Its main features include:

  1. It is configured with the left bus, and the right bus is omitted.

  2. All control output elements (coils) and function blocks (application instructions) share the same energy flow input terminal.

  The electrical control diagram and LAD are equivalent to a certain degree, as shown in the following figure.

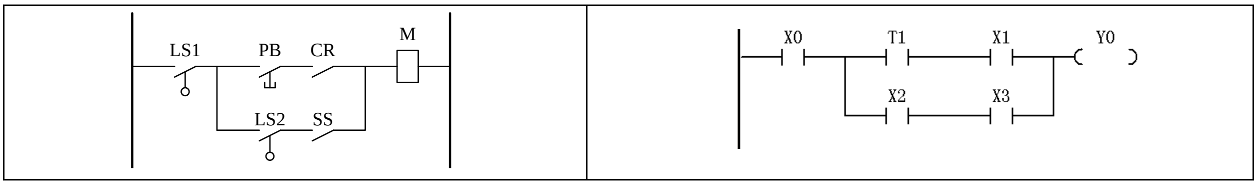


Figure 4-1 The equivalence between electrical control diagram and LAD

- LAD basic programming elements

  According to the principles in the electrical (relay) control diagram, several basic programming elements are abstracted for theLAD:

  1. Left bus: It corresponds to the control bus in electrical control diagram, and provides control power for the control circuit.

  2. Connecting line (— |): It represents the electrical connection in the electrical control diagram, and is used to conduct other inter-connected elements.

  3. Contact (⊣⊢): It represents the input contact in the electrical control diagram, it controls the ON/OFF and direction of control currents in the circuit. The parallel and serial connection of the contacts essentially represents the operational relationship of the input logic of the control circuit, and controls the transmission of the energy flow.

  4. Coil (-( )): It represents the relay output in the electrical control diagram.

  5. Function block (□): It is also known as the application instruction, and corresponds to the actuator or functional device that completes the special functions in the electrical control diagram. It can perform the specific control functions or control calculation functions (such as data transmission, data calculation, timer, counter, etc.).

- Energy flow

  Being an important concept in LAD, the energy flow is used to drive coil elements and application instructions, which is similar tothe control current output by the driving coil, and executed by the execution unit in the electrical control diagram.

  In LAD, the coils or the front end of the application instruction need to be connected to the energy flow. The coil element can be output and the application instructions can be effectively executed only when the energy flow is valid.

  The following figure demonstrates the energy flow transferin LAD and the drive of the energy flow to the coils or function blocks.

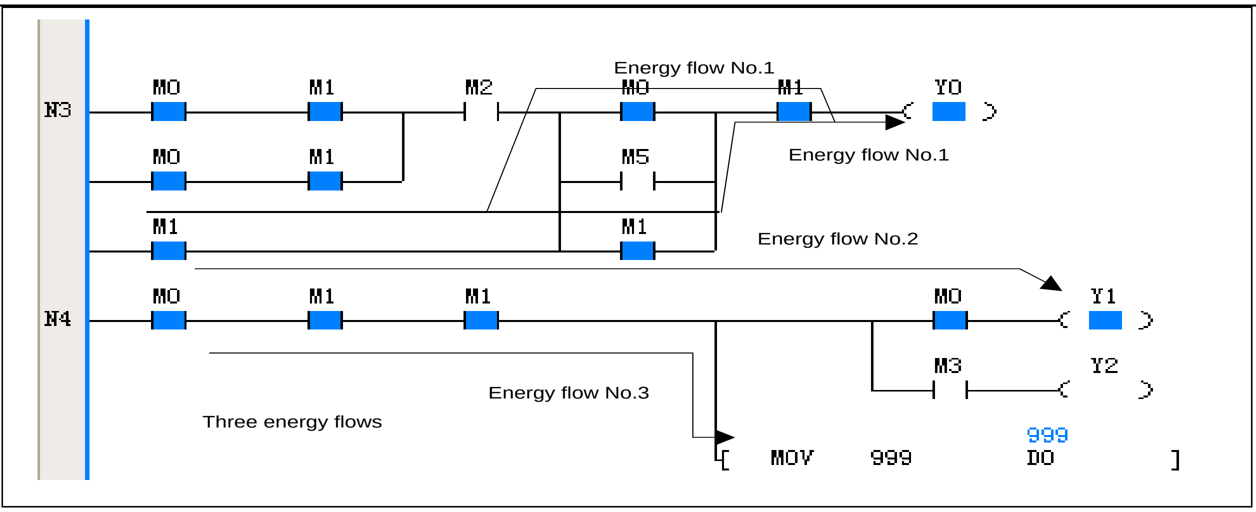

N3
M0
M1
M2
Energy flow No.1
M0
M1
Y0
M0
M1
M5
Energy flow No.1
M1
M1
Energy flow No.2
N4
M0
M1
M1
M0
Y1
M3
Y2
Energy flow No.3
Three energy flows
MOV
999
999
D0

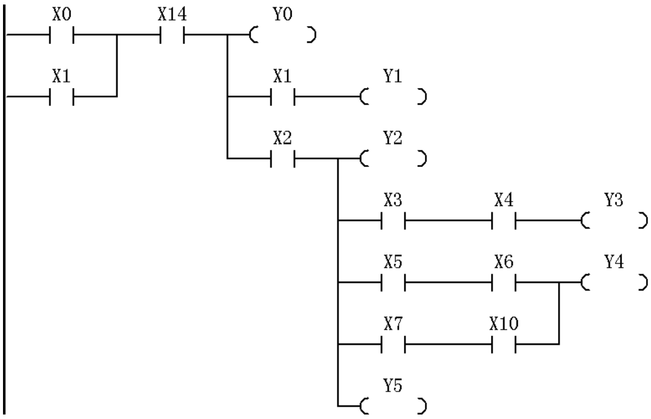
Figure 4-2 Energy flow and its driving function

4.1.2 IL

The IL is a textual user program, or a set of the instruction sequences written by the users.

The user program stored in the PLC main module for execution is actually the instruction sequence recognizable by the main module, and the system executes each instruction in the sequence one by one to realize the control function of the user program.

The following figure is an application instance of converting a LAD into an IL.

LAD	1L
 X0 X14 Y0 X1 X1 Y1 X2 Y2 X3 X4 Y3 X5 X6 Y4 X7 X10 Y5	LD X0 OR X1 AND X14 MPS OUT Y0 AND X1 OUT Y1 MPP AND X2 MPS OUT Y2 AND X3 AND X4 OUT Y3 MRD LD X5 AND X6 LD X7 AND X10 ORB ANB OUT Y4 MPP OUT Y5

4.1.3 SFC

The SFC is a graphical design language for the user program framework that is commonly used to implement sequential control functions.

Sequence control is a control process that can be divided into multiple procedures (processing steps) and proceed them according to the certain working sequence.

The user program designed with SFC is relatively straight forward and clear because its program structure is consistent with the actual sequence control process.

The following figure is an application instance of a simple SFC.

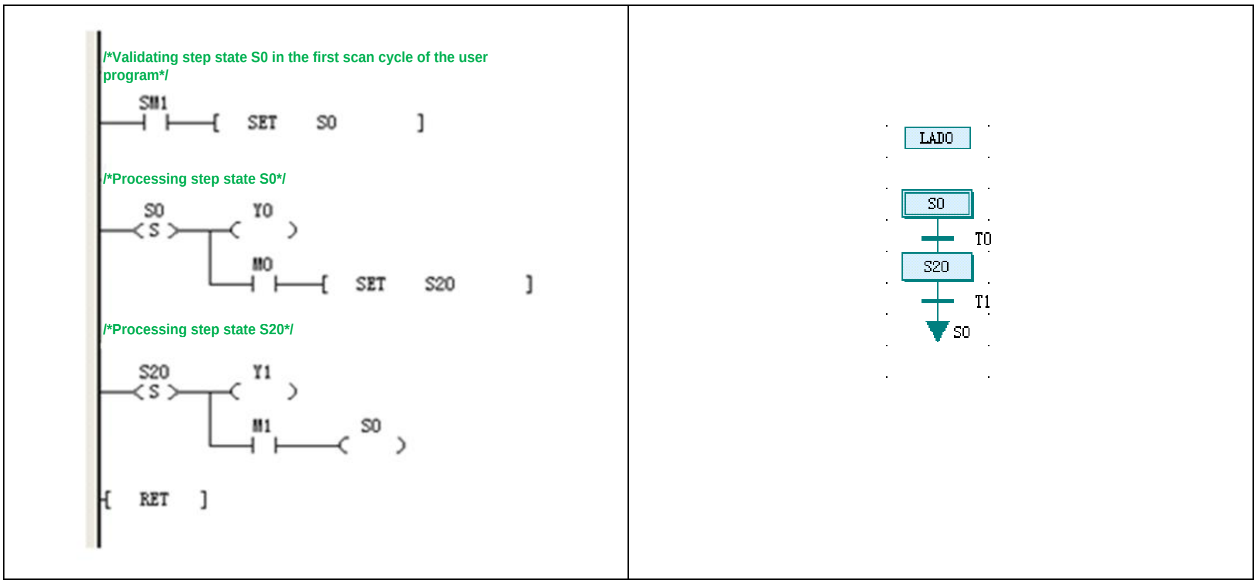

/*Validating step state S0 in the first scan cycle of the user program*/
SM1
SET S0
/*Processing step state S0*/
S0
S
Y0
M0
SET S20
/*Processing step state S20*/
S20
S
Y1
M1
S0
RET
LAD0
S0
T0
S20
T1
S0

Figure 4-3 Application instance of SFC

4.2 Program Elements

The program elements include user program, system block and data block. You can change these program elements by programming.

4.2.1 User Program

The user program is the program code written by the user. It is compiled into an executable instruction sequence,downloaded to the controller, and the controller executes the control function of the user program.

The user program consists of three program organization units (POU): main program (MAIN), subprogram (SBR) and interrupt program (INT).

- MAIN
  The main program is the main body and framework of the user program. When the system is in RUN, the main program is executed cyclically.

  One user program has only one main program.

- SBR
  A subprogram is a structurally and functionally independent user program that can be called by other program bodies. Subprograms generally have call operand interfaces that are executed only when being called.

  A user program can have random number of subprograms, or no subprogram at all.

- INT
  An interrupt program is a user program segment that handles a specific interrupt event. A specific interrupt event always corresponds to aspecific interrupt program.

  Upon the occurrence of an interrupt event, a normal scan cycle is interrupted, and the user program flow automatically jumps to the execution of the interrupt program until the interrupt return instruction system returns to the normal scan cycle.

  A user program can have random number of interrupt programs, or no interrupt program at all.

4.2.2 System block

The system block contains multiple system configuration options. You can modify, compile and download the system block to configure the operation mode of the main module.

For details about how to use the system configuration items, refer tosection 2.2.1 "System block"in the manual or the related description in *Auto Studio Programming Software User Manual*.

4.2.3 Data block

The data block contains the set value of D or R element. When the data block is downloaded to the controller, the designated D or R element is assigned a set value, thereby achieving the purpose of batch setting the D or R element value.

If the "Datablock enabled" in the advanced settings tab of the system block, the D or R elements are initialized by the data block before the user program is in RUN.

4.3 Block comment and variable comment of the program

4.3.1 Block comment

When programming, you can add the block comments to the program. Block comments textually describe the relevant programs. Each block comment takes up an entire line of space.

In the program, right click and select **Insert Row** to insert a row above the current row. You can use an empty row toseparate two program blocks.

To make a block comment, you need to first select an empty row, then right click and select **Insert Block Comment**, as shown below.

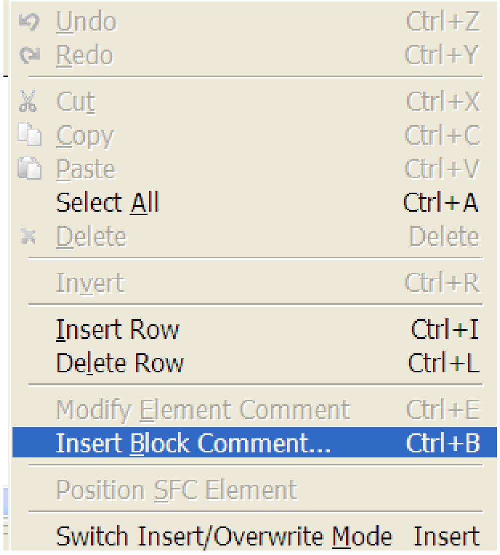
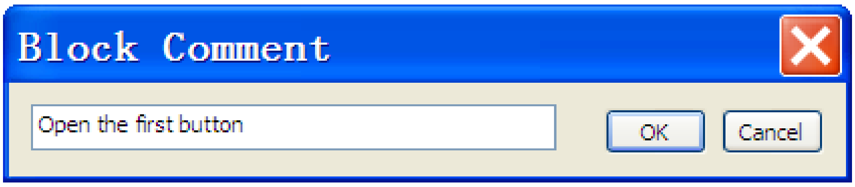

Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Select All Ctrl+A
Delete Delete
Invert Ctrl+R
Insert Row Ctrl+I
Delete Row Ctrl+L
Modify Element Comment Ctrl+E
Insert Block Comment... Ctrl+B
Position SFC Element
Switch Insert/Overwrite Mode Insert

Figure 4-4 Adding block comment

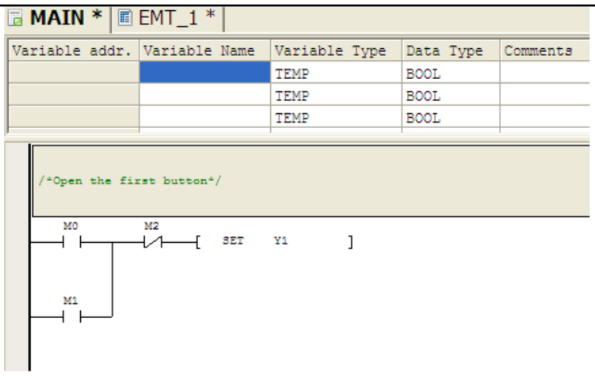
Entering your comment into the block comment dialogue box that pops out and clicking the OK button.


Block Comment
OK
Cancel

Open the first button

Figure 4-5 Block comment dialogue box

The software automatically adds "/*" and "*/" to both sides of the entered text, and displays them in green color, as shown below:



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	
		TEMP	BOOL	
		TEMP	BOOL	

/*Open the first button*/

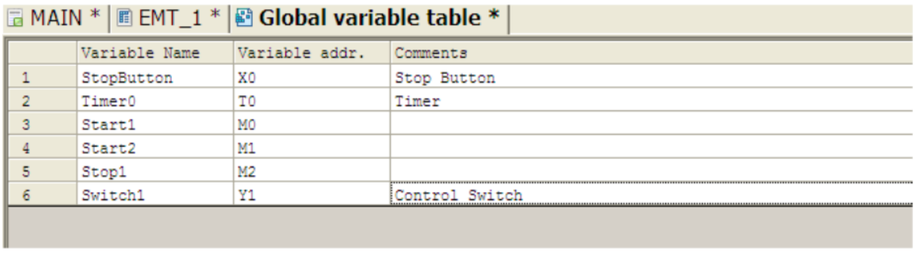
Figure 4-6 The clock comment in the program

A block comment occupies a whole row. You cannot add a block comment to an occupied row, nor can a row occupiedby a comment be used for other purposes.

4.3.2 Variable Comment

You can define the variables in the global variable table and local variable table. (For details, refer to section 2.2.3 "Global variable table" and section 4.4.3 "Definition of the SBR variable table"), and correctly defined variables can be used in the LAD. A variable can stand for a

certain address to make the program more readable. Figure 4-7 shows some variables defined in a global variable table.



	Variable Name	Variable addr.	Comments
1	StopButton	X0	Stop Button
2	Timer0	T0	Timer
3	Start1	M0	
4	Start2	M1	
5	Stop1	M2	
6	Switch1	Y1	Control Switch

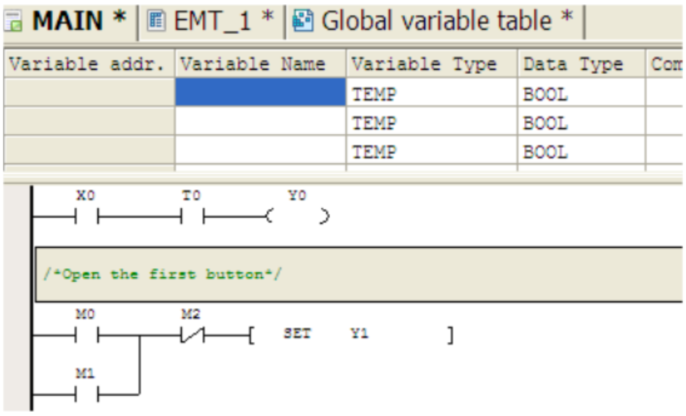
Figure 4-7 Variables defined in the global variable table

- Symbol addressing

When the defined variables are used, you can switch between the variable name and the element address by selecting **Symbol addressing** menu.

The following figure shows the same LAD in both display modes.

The following figure shows the state of the LAD when the Symbol Addressing is not selected.

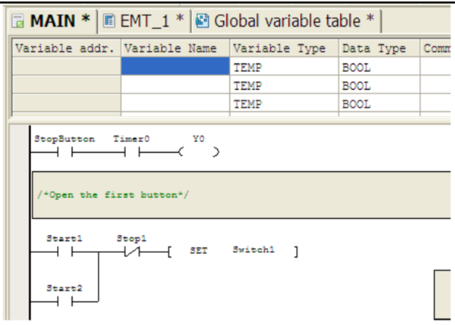


Variable addr.	Variable Name	Variable Type	Data Type	Com
		TEMP	BOOL	
		TEMP	BOOL	
		TEMP	BOOL	

/*Open the first button*/

Figure 4-8 The state of symbol addressing is not selected

The following figure shows the state of the LAD when the Symbol Addressing is selected.

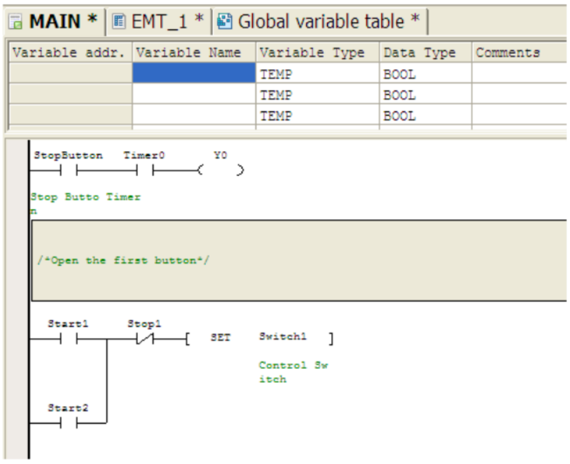

MAIN *
EMT_1 *
Global variable table *
StopButton
Timer0
Y0
/*Open the first button*/
Start1
Stop1
SET
Switch1
Start2

Variable addr.	Variable Name	Variable Type	Data Type	Comm
		TEMP	BOOL	
		TEMP	BOOL	
		TEMP	BOOL	

Figure 4-9 The state of symbol addressing is selected

- Elementcomment

You can control whether the element comments are displayed in the LAD by selecting the **Element comment** menu, as shown in Figure4-10.


MAIN *
EMT_1 *
Global variable table *
StopButton
Timer0
Y0
Stop Butto Timer
n
/*Open the first button*/
Start1
Stop1
SET
Switch1
Control Sw
itch
Start2

Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	
		TEMP	BOOL	
		TEMP	BOOL	

Figure 4-10 The LAD program displaying the element comments

4.4 Subprogram

4.4.1 Concept

Being an optional part of the user program, a subprogram (SBR) is an independent program organization unit (POU) that can be called by the main program or other SBRs.

You can use SBRs in your user program to:

1. Reduce the size of the user program. You can write a repeated program section as a SBR and call it whenever necessary.

2. Clarify the program structure, particularly the structure of the main program.

3. Make the user program more transplantable.

4.4.2 Notestouse SBRs

When writing or calling a SBR, You need to note the following:

1. The PLCs support a maximum of 6 levels of SBR nesting calling.

The following is an fine application instance of 6-level of SBR nesting calling:

MAIN→SBR1→SBR2→SBR3→SBR4→SBR5→SBR6

(where the "→" represents calling the corresponding SBRs with the CALL instruction)

2. The PLCs do not support recursive call and cyclic call of SBRs.

The following two application instances show two illegal SBR callings.

- MAIN→SBR0→SBR0 (recursive call, illegal)
- MAIN→SBR0→SBR1→SBR0 (cyclic call, illegal)

3. A maximum of 64 SBRs can be defined in a user program.

4. A maximum of 16 bit and 16 word types of variables can be defined in the variable table of a SBR.

5. When calling a SBR, you need to note that the operand type of the CALL instruction needs to match the variable type defined in the variable table of theSBR, and the compiler checks whether the match is correct.

6. The interrupt programs are not allowed to call SBRs

4.4.3 Definition of the SBR variable table

- SBR variabletable
  The SBR variable table is used to displays all SBR interface parameters and local variables (collectively referred to as variables) and specify their properties.
- SBR variable properties
  The SBR variables (including interface parameters and local variables) have the following properties:

  1. Variable address

  Based on the variable data type, the software automatically assign a fixed LM or V element address to each SBR variable in sequence.

  2. Variable name

  You can give each SBR variable a name (alias). You can use a variable in the program by quoting its name.

  3. Variable type

  The SBR variables are classified into the four types:IN, OUT, IN_OUT, and TEMP.

  - IN: The IN type variable is used to transfer the input value of the SBR when a SBR is being called.
  - OUT: The OUT type variable is used to call the return value for transferring the SBR when a SBR returns.
  - IN_OUT: The IN type variable is used to transfer the input value of the SBR when a SBR is being called, or call the return value for transferring the SBR when a SBR returns.
  - The TEMP type variables are only used as local variables that are valid only within scope of the SBR.

4. Variable data type

The properties of the variable data types specify the data widthand range of the variables. The variable data types are listed in the following table.

Table 4-1 Variable data types

Data type	Description	Occupied LM/V element address
BOOL	Bit type	One LM element address
INT	Signed integer type	One V element address
DINT	Signed double integer type	Two consecutive V element addresses
REAL	Floating-point type	Two consecutive V element addresses

4.4.4 SBR parameter transfer

If local input or output variables are defined in the SBR when a SBR is called in the main program, you need to input the corresponding variable values, global or temporary variable elements into the SBR interface parameters. You need to note that the local variable need to be of the same data type with the interface parameter.

4.4.5 SBR application instance

Here is an application instance of how to write and call a SBR.

- Introduction to the function of the application instance
  Calling SBR_1 in the main program to perform the addition calculation (3+2) of two integer constants, and assigning the operation result 5 to D0.
- Operation procedures of the application instance
  Step 1: Creating a SBR in the project and naming it as SBR_1.

Step 2: Writing the SBR_1.

1. Creating the SBR calling operand interface through the variable table of the SBR_1.

1) Defining variable 1: Naming it as IN1 (variable type: IN). It is used as the INT type data and sequentially assigned a V elementaddress V0.

2) Defining variable 2: Naming it as IN2 (variable type: IN). It is used as the INT type data and sequentially assigned a V elementaddress V1.

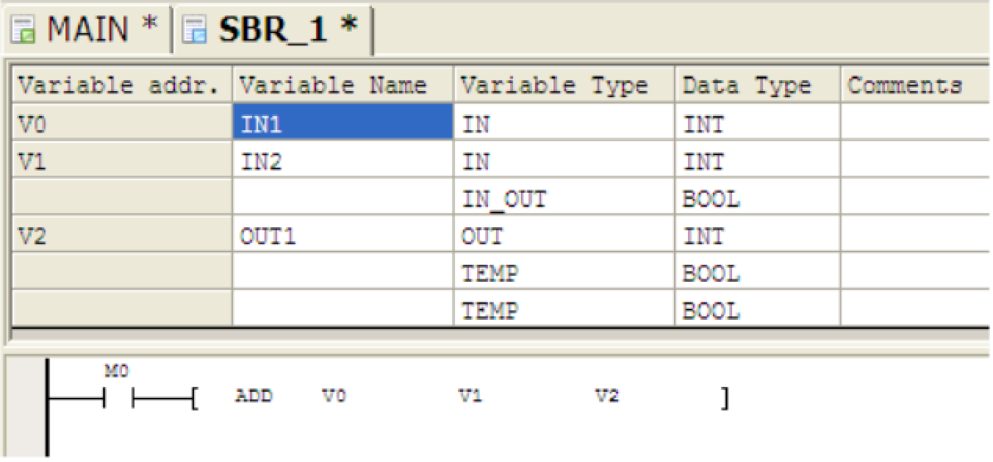
3) Defining variable 3: Naming it as OUT1 (variable type: OUT). It is used as the INT type data and sequentially assigned a V elementaddress V2.

2. Writing the SBR_1 as:

```
LD   SM0

ADD   #IN1   #IN2   #OUT1
```

The writing process of the SBR_1 is shown in the following figure.



Variable addr.	Variable Name	Variable Type	Data Type	Comments
V0	IN1	IN	INT	
V1	IN2	IN	INT	
		IN_OUT	BOOL	
V2	OUT1	OUT	INT	
		TEMP	BOOL	
		TEMP	BOOL	

```
M0
[ ADD V0 V1 V2 ]
```

Figure 4-11 Writing process of the SBR_1

Step 3: Writing the main program and calling the SBR

Using the CALL instruction in the main program to call the SBR_1.

The corresponding main program is as follows:

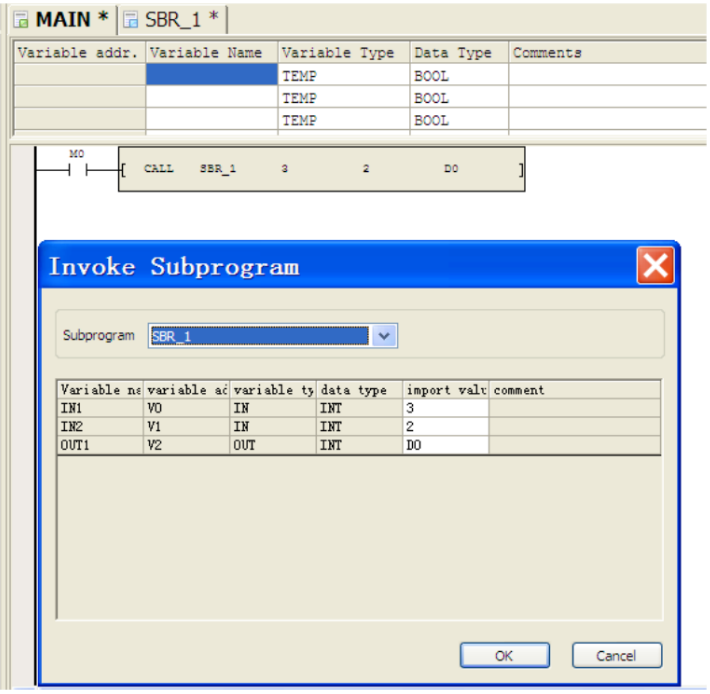
LD M0

CALL SBR_1 3 2 D0

You can use the parameters to transfer the corresponding relationship table and filling in the parameters that are brought in or returned when the SBR is called.

- Parameter IN1 is brought in to transfer the constant integer 3
- Parameter IN2 is brought in to transfer the constant integer 2
- The return value OUT1 is stored in D0

The above program is shown in the following figure.



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	
		TEMP	BOOL	
		TEMP	BOOL	

```
M0
[ CALL SBR_1 3 2 D0 ]
```

Subprogram: SBR_1

Variable na	variable ac	variable ty	data type	import valu	comment
IN1	V0	IN	INT	3	
IN2	V1	IN	INT	2	
OUT1	V2	OUT	INT	D0	

Figure 4-12 Calling the SBR

Step 4: Compiling, downloading and running the user program to verify the logic of the subprogram.

- Execution result of the application instance

When M0 is ON, SBR_1 is called. Values 2 and 3 are transferred to the operands IN1 and IN2 to carry out the addition operation. The result 5 is then returned to the main program, that is, D0 is 5.

4.5 General information of the instructions

4.5.1 Instruction operands

The instruction operands can be classified into the following two types:

- Source operands: Or S (or S1, S2, S3 … when there are more than one of them in the same instruction). The instruction reads values from source operands for calculation.
- Destination operands: or D (or D1, D2, D3 … when there are more than one of them in the same instruction). The instruction controls or outputs values to the destination operands.

The operands could be bit elements, word elements, double word elements, or constants. For details about the instruction description, refer to Chapter 5 and Chapter 6.

4.5.2 Flag bit

The instruction operations may affect the following three kinds of flags bits.

- Zero flag SM80
  Setting the zero flag when the instruction operation result is zero.
- Carry flag SM81
  Setting the carry flagwhen the instruction operation result is a carry.
- Borrow flag SM82
  Setting the borrow flag when the instruction operation result is a borrow.

4.5.3 Restrictions on the use of instructions

There are some restrictions on the application of some instructions, some of which are listed below. For details, refer to the detailed instructions of the relevant instructions.

- Exclusive hardware resources
  Some instructions requires hardware resources. When a specific hardware is being used by a certain instruction, the access to the hardware is denied to other instructions, because the occupation of the resource is exclusive.

  Taking the high-speed counting instructions and SPD instruction for application instance. Any of these instructions can occupy certain input points among X0 – X7. The limited resources make it impossible to execute these instructions at the same time.
- Exclusive time
  The execution of certain instructions may take a period of time. Therefore, when using these instructions, you need to ensure that the instructions have enough time to complete the function. Only one can be executed at a certain time when the system is running.

  Taking the XMT instruction for application instance. Due to the timeliness of communication, the XMT instruction is sent to the free port, and only one can be executed at the same time. Similarly, the free port can execute only one RCV instruction once. Everytime when a Modbusinstruction is being executed, the communication channel is unavailable to other instructions for a while. The same applies to other instructions such as high-speed output instruction, positioning instructions.
- Application limit of the instructions
  Some instructions cannot be used in certain situations due to their limited application scope.

  For example, the instruction pair MC/MCR cannot be used in the step state programmed by SFC.

Chapter 5 Basic instructions

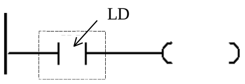
This chapter detailedly describes basic instructions of VC series micro-PLCs, including the instruction format (form), operand,influenced flag bit, function, application instance, and sequence diagram.

Chapter 5 Basic instructions 57
- 5.1 Contact logic instructions....59
  - 5.1.1 LD: NO contact instruction 59
  - 5.1.2 LDI: NC contact instruction....59
  - 5.1.3 AND: NO contact and instruction 59
  - 5.1.4 ANI: NC contact and instruction 60
  - 5.1.5 OR: NO contact or instruction 60
  - 5.1.6 ORI: NO contact or instruction 61
  - 5.1.7 OUT: Coil output instruction 61
  - 5.1.8 ANB: Energy flow block and instruction 61
  - 5.1.9 ORB: Energy flow block or instruction....62
  - 5.1.10 MPS: Output energy flow push instruction 62
  - 5.1.11 MRD: Instruction for reading output energy flow stack top value 63
  - 5.1.12 MPP: Output energy flow stack pop instruction....63
  - 5.1.13 EU: Rising edge detection instruction 64
  - 5.1.14 ED: Falling edge detection instruction....64
  - 5.1.15 LDP: Rising edge of contact....65
  - 5.1.16 LDF: Falling edge of contact 65
  - 5.1.17 ANDP: Rising edge of contact....66
  - 5.1.18 ANDF: Falling edge of contact 66
  - 5.1.19 ORP: Rising edge of contact....67
  - 5.1.20 ORF: Falling edge of contact....67
  - 5.1.21 PLP: Rising edge output instruction 68
  - 5.1.22 PLF: Falling edge output instruction....68
  - 5.1.23 INV: Energy flow negation instruction 69
  - 5.1.24 SET: Coil set instruction....69
  - 5.1.25 RST: Coil reset instruction....69
  - 5.1.26 NOP:No operation instruction....70
- 5.2 Main control instructions....70
  - 5.2.1 MC: Main control instruction....70
  - 5.2.2 MCR: Main control reset instruction 70
- 5.3 SFC instructions 71
  - 5.3.1 STL: SFC state loadinginstruction....71
  - 5.3.2 SET Sxx: SFC state transition instruction 72
  - 5.3.3 OUT Sxx: SFC state jumpinstruction....72
  - 5.3.4 RST Sxx: SFC state reset instruction....72
  - 5.3.5 RET: SFC program segment end instruction....73
- 5.4 Timer instructions 73
  - 5.4.1 TON: Turn-on delay timing instruction....73
  - 5.4.2 TONR: Memory-type turn-on delay timing instruction 73
  - 5.4.3 TOF: Turn-off delay timing instruction 74
  - 5.4.4 TMON: Non-retriggerable monostable timing instruction 75
- 5.5 Counter instructions....76

5.5.1 CTU: 16-bit increment counterinstruction 76
5.5.2 CTR: 16-bit cycliccountinginstruction 76
5.5.3 DCNT:32-bit increment and decrement counting instruction 77

5.1 Contact logic instructions

5.1.1 LD: NO contact instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LD *(S)*	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T			

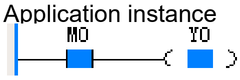
Operand description

S: Source operand

Function description

Connecting to the left bus to connect (state: ON) or disconnect (state: OFF) the energy flow.

Application instance

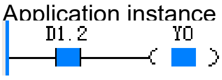


LD M0

OUT Y0

When M0 is ON, the Y0 output is ON.

Application instance

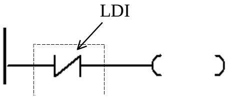


LD D1.2

OUT Y0

When the second bit of D1 is 1, the Y0 output is ON.

5.1.2 LDI: NC contact instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LDI *(S)*	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T			

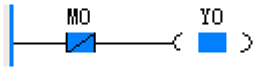
Operand description

S: Source operand

Function description

Connecting to the left bus to connect (state: ON) or disconnect (state: OFF) the energy flow.

Application instance

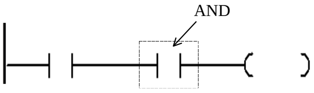


LDI M0

OUT Y0

When M0 is OFF, the Y0 output is ON.

5.1.3 AND: NO contact and instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: AND *(S)*	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T			

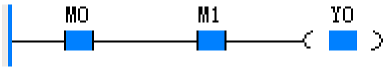
Operand description

S: Source operand

Function description

Performing the "AND" operation on the ON/OFF state of the designated contact (***S***) and the current energy flow, and assigning the obtained result to the current energy flow.

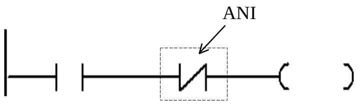
Application instance



LD M0
AND M1
OUT Y0

When M0 is ON and M1 is ON, the Y0 output is ON.

5.1.4 ANI: NC contact and instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5													
	Influenced flag bit														
IL: ANI *(S)*	**Step length**	**1**													
Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T			

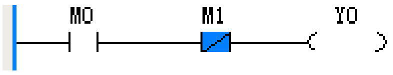
Operand description

S: Source operand

Function description

Performing the "NOT" operation on the ON/OFF state of the designated contact (S), and then performing the "AND" operation on the obtained result and the current energy flow value, and assigning the obtained result to the current energy flow.

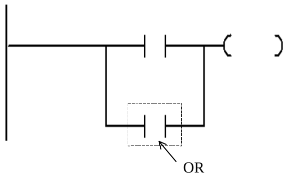
Application instance



LD M0
ANI M1
OUT Y0

When M0 is ON and M1 is OFF, the Y0 output is ON.

5.1.5 OR: NO contact or instruction

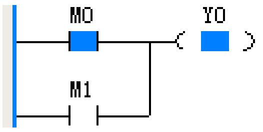
LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5													
	Influenced flag bit														
IL: OR *(S)*	**Step length**	**1**													
Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T			

Operand description

S: Source operand

Function description

Application instance

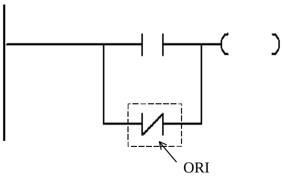


LD M0
OR M1
OUT Y0

Performing the "OR" operation on the ON/OFF state of the designated contact (S) and the current energy flow, and assigning the obtained result to thecurrent energy flow.

When M0 or M1 is ON, the Y0 output is ON.

5.1.6 ORI: NO contact or instruction

LAD:	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	**Influenced flag bit**	
IL: ORI *(S)*	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T			

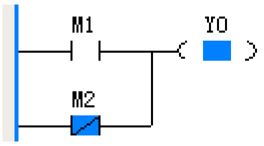
Operand description

S: Source operand

Function description

Performing the "NOT" operation on the ON/OFF state of the designated contact (***S***), and then performing the "OR" operation on the obtained result and the current energy flow value, and assigning the obtained result to the current energy flow.

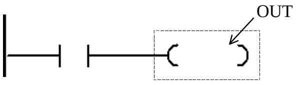
Application instance



```
LD    M1
ORI   M2
OUT   Y0
```

When M1 is ON or M2 is OFF, the Y0 output is ON.

5.1.7 OUT: Coil output instruction

LAD:	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	**Influenced flag bit**	
IL: OUT *(S)*	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T			

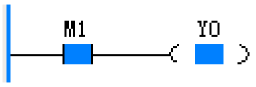
Operand description

S: Source operand

Function description

Assigning the value of the current energy flow to the designated coil (***D***).

Application instance

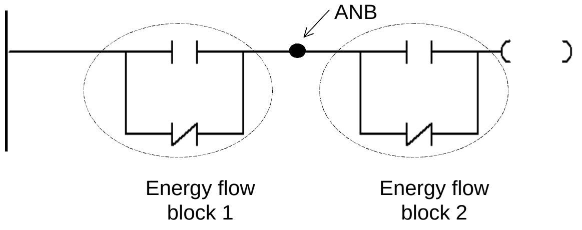


```
LD    M1
OUT   Y0
```

When M1 is ON, the Y0 output is ON.

5.1.8 ANB: Energy flow block and instruction

LAD:	**Applicable model**	VC1S VC1 VC2 VC3 VC5

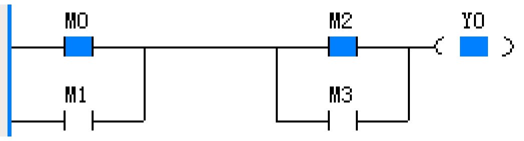
 ANB Energy flow block 1 Energy flow block 2	**Influenced flag bit**	
IL: **ANB**	**Step length**	**1**

Operand description

Function description

Conducting "AND"operation on the energy flow values of two energy flow blocks, and then assigning the obtained result tothe current energy flow.

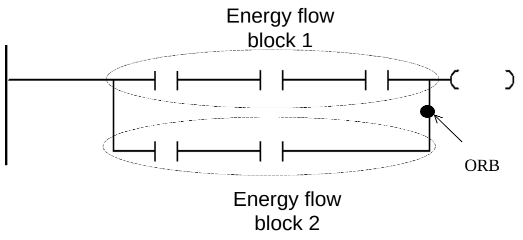
Application instance


M0
M2
Y0
M1
M3

```
LD M0
OR M1
LD   M2
OR M3
ANB
OUT   Y0
```

When either M0 or M1 is on, and either M2 or M3 is ON, the Y0 output is ON.

5.1.9 ORB: Energy flow block or instruction

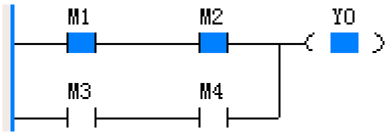
LAD:  Energy flow block 1 ORB Energy flow block 2	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **ORB**	**Step length**	**1**

Operand description

Function description

Conducting "OR" operation on the energy flow values of two energy flow blocks, and then assigning the obtained result to the current energy flow.

Application instance

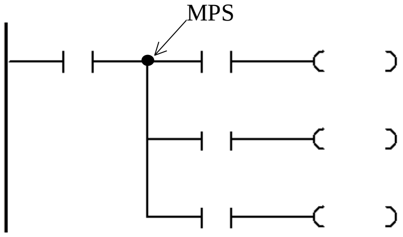

M1
M2
Y0
M3
M4

```
LD M1
AND M2
LD M3
AND M4
ORB
OUT   Y0
```

When both M1 and M2 are ON, or both M3 and M4 are ON, the Y0 output is ON.

5.1.10 MPS: Output energy flow push instruction

LAD:	**Applicable model**	VC1S VC1 VC2 VC3 VC5

	**Influenced flag bit**	
IL: **MPS**	**Step length**	**1**

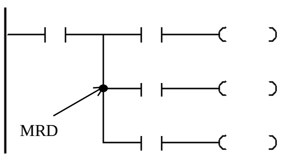
Function description

Pushing and storing the current energy values for the energy flow calculation of subsequent output branches.

Note:

It is not allowed to use MPS instructions(without MPP instructions among them) for over 8 consecutive times in only one ladder diagram network,otherwise the energy flow output stack may overflow.

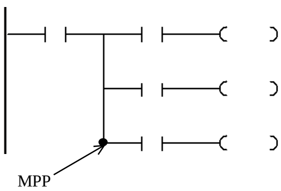
5.1.11 MRD: Instruction for reading output energy flow stack top value

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **MRD**	**Step length**	**1**

Function description

Assigning the top value of the energy flow output stack to the current energy flow.

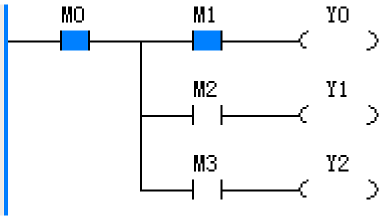
5.1.12 MPP: Output energy flow stack pop instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **MPP**	**Step length**	**1**

Function description

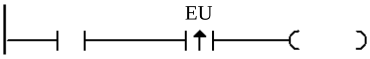
Performing the Pop operation on the energy flow output stack, andassigning the popped value to the current energy flow.

Application instance



LDM0
MPS
AND M1
OUT Y0
MRD
AND M2
OUT Y1
MPP
AND M3
OUT Y2

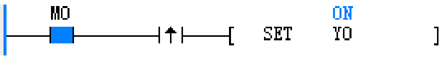
5.1.13 EU: Rising edge detection instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: EU	**Step length**	**2**

Function description

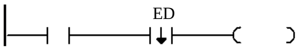
Comparing the state of the input energy flow in the current scan with that in the last scan. If the energy flow rises (OFF→ON), the output in the current scan cycle is valid.

Application instance



LD M0
EU
SET Y0

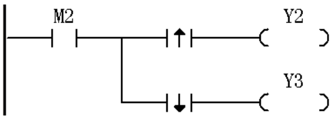
5.1.14 ED: Falling edge detection instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: ED	**Step length**	**2**

Function description

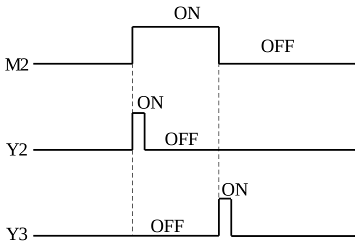
Comparing the state of the input energy flow in the current scan with that in the last scan. If the energy flow falls (OFF→ON), the output in the current scan cycleis valid.

Application instance



LD M2
MPS
EU
OUT Y2
MPP
ED
OUT Y3

Sequence chart of the application instance



Note

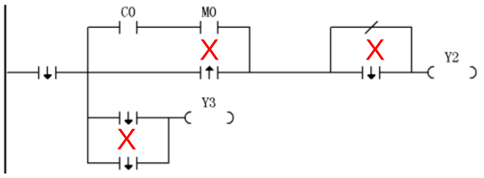
1. In two consecutive scan cycles, the states of M2 contact are OFF and ON respectively, and the EU instruction detects a rising edge change, so that Y2 outputs an ON state with a scancycle width.

2. In two consecutive scan cycles, the states of M2 contact are ON and OFF respectively, and the ED instruction detects a falling edge edge, so that Y3 outputsan ON state with a scan cycle width.

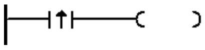
In LAD, the rising edge contact andfalling edge contact instructions need to be used in series rather than in parallel with other contact elements.

In LAD, the rising edge contact and falling edge contact instructions cannot be connected to the left energy flow bus directly.

The following figure shows an example of incorrect use of EU and ED instructions in LAD.



5.1.15 LDP: Rising edge of contact

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LD *(S)*	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM				C	T			

Operand description

S: Source operand

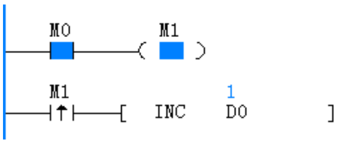
Function description

The LDP command is used to fetch the rising edge of the contact signal

The rising edge jump of the corresponding signal is detected in, then the contact is effective,

On the next scan, the contact becomes invalid

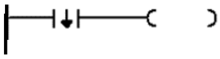
Application instance



```
LD    M0
OUT   Y0
LDP   M1
INC   D0
```

When M0 is ON, the M1 output is ON. At this time, M1 changes from OFF to ON and remains valid for 1 scanning cycle, and D0 performs increase 1. In the next scanning cycle, M1 will be invalid and D0 remains 1.

5.1.16 LDF: Falling edge of contact

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LD *(S)*	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM				C	T			

Operand description

S: Source operand

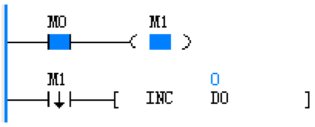
Function description

The LDF command is used to fetch the falling edge of the contact signal

The falling edge jump of the corresponding signal is detected in, then the contact is effective,

On the next scan, the contact becomes invalid

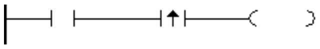
Application instance



```
LD    M0
OUT   Y0
LDF   M1
INC   D0
```

When M0 is ON, the M1 output is ON. At this time,If M1 changes from ON to OFF and remains valid for 1 scanning cycle, D0 performs increase 1. In the next scanning cycle, M1 will be invalid and D0 remains 1.

5.1.17 ANDP: Rising edge of contact

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LD *(S)*	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM				C	T			

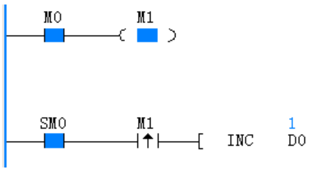
Operand description

S: Source operand

Function description

The ANDP instruction is to participate in the AND operation of the rising edge of the contact.

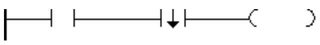
Application instance



```
LD    M0
OUT   Y0
LD    SM0
ANDP  M1
INC   D0
```

When M0 is ON, the M1 output is ON. At this time,If M1 changes from OFF to ON and remains valid for 1 scanning cycle, D0 performs increase 1. In the next scanning cycle, M1 will be invalid and D0 remains 1.

5.1.18 ANDF: Falling edge of contact

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LD *(S)*	**Step length**	**1**

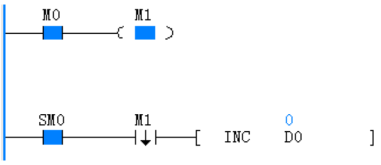
Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM				C	T			

Operand description

S: Source operand

Function description

Application instance

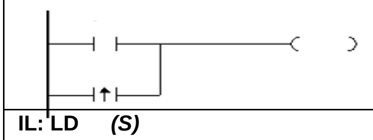


```
LD    M0
OUT   Y0
LD    SM0
ANDF  M1
INC   D0
```

The ANDF instruction is to participate in the AND operation of the falling edge of the contact.

When M0 is ON, the M1 output is ON. At this time,If M1 changes from ON to OFF and remains valid for 1 scanning cycle, D0 performs increase 1. In the next scanning cycle, M1 will be invalid and D0 remains 1.

5.1.19 ORP: Rising edge of contact

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LD *(S)*	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM				C	T			

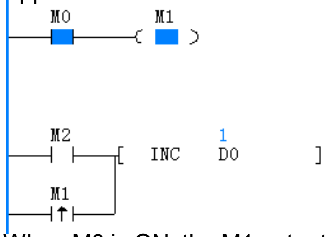
Operand description

S: Source operand

Function description

The ORP instruction is to participate in the OR operation of the rising edge of the contact.

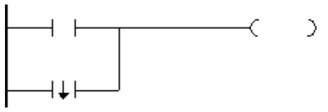
Application instance



LD M0
OUT Y0
LD M2
ORP M1
INC D0

When M0 is ON, the M1 output is ON. At this time,If M1 changes from OFF to ON and remains valid for 1 scanning cycle, D0 performs increase 1. In the next scanning cycle, M1 will be invalid and D0 remains 1.

5.1.20 ORF: Falling edge of contact

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LD *(S)*	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	BOOL	X	Y	M	S	LM	SM				C	T			

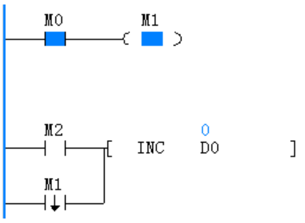
Operand description

S: Source operand

Function description

The ORF instruction is to participate in the OR operation of the falling edge of the contact.

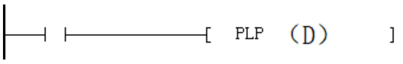
Application instance



LD M0
OUT Y0
LD M2
ORF M1
INC D0

When M0 is ON, the M1 output is ON. At this time,If M1 changes from ON to OFF and remains valid for 1 scanning cycle, D0 performs increase 1. In the next scanning cycle, M1 will be invalid and D0 remains 1.

5.1.21 PLP: Rising edge output instruction

<table>
<tr><td colspan="8" rowspan="2">LAD:
 </td><td colspan="3">Applicable model</td><td colspan="5">VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="5"></td></tr>
<tr><td colspan="8">IL: LD <i>(S)</i></td><td colspan="3">Step length</td><td colspan="5">1</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S</td><td>BOOL</td><td>X</td><td>Y</td><td>M</td><td>S</td><td>LM</td><td>SM</td><td></td><td></td><td></td><td>C</td><td>T</td><td></td><td></td><td></td></tr>
</table>

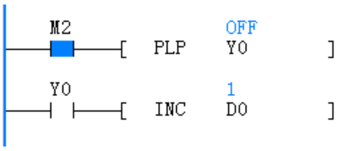
Operand description

S: Source operand

Function description

The PLP instruction is to take the rising edge of the output signal. If the rising edge of the output signal is detected in this scanning, the output contact will be ON. In the next scanning cycle, the contact of the output will become invalid. .

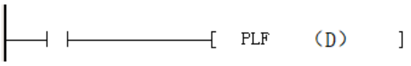
Application instance



```
LD    M2
PLP   Y0
LD    Y0
INC   D0
```

When M2 is ON, the Y0 output is ON and remains valid for 1 scanning cycle, D0 performs increase 1. In the next scanning cycle, M1 will be invalid and D0 remains 1.

5.1.22 PLF: Falling edge output instruction

<table>
<tr><td colspan="8" rowspan="2">LAD:
 </td><td colspan="3">Applicable model</td><td colspan="5">VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="5"></td></tr>
<tr><td colspan="8">IL: LD <i>(S)</i></td><td colspan="3">Step length</td><td colspan="5">1</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S</td><td>BOOL</td><td>X</td><td>Y</td><td>M</td><td>S</td><td>LM</td><td>SM</td><td></td><td></td><td></td><td>C</td><td>T</td><td></td><td></td><td></td></tr>
</table>

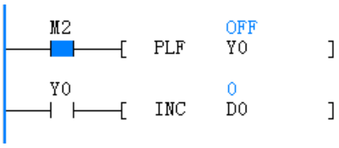
Operand description

S: Source operand

Function description

The PLP instruction is to take the Falling edge of the output signal. If the falling edge of the output signal is detected in this scanning, the output contact will be ON. In the next scanning cycle, the contact of the output will become invalid. .

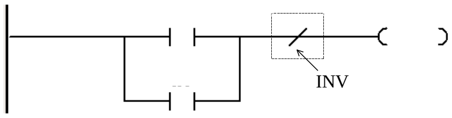
Application instance



```
LD    M2
PLF   Y0
LD    Y0
INC   D0
```

When M2 changes from ON to OFF, the Y0 output is ON and remains valid for 1 scanning cycle, D0 performs increase 1. In the next scanning cycle, M1 will be invalid and D0 remains 1.

5.1.23 INV: Energy flow negation instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: INV	**Step length**	**1**

Function description

Performing the "INV" operation on the current energy flow value and then assigning the obtained result to the current energy flow.

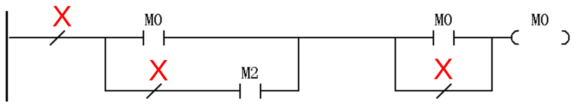
Note

In LAD, the INV instruction needs to be used in series rather than in parallel with other contact elements.

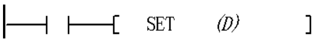
INV cannot be used as the first instruction in the parallel input branches.

In LAD, the INV instruction cannot be directly connected to the left energy flow bus.

The following figure shows an example of incorrect use of the INV instruction in LAD.



5.1.24 SET: Coil set instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: SET *(S)*	**Step length**	**1**

Operand	Type						Applicable soft element								Indexing
S	BOOL		Y	M	S	LM	SM		Dx.y		C	T			

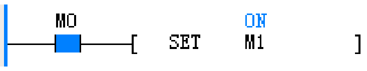
Operand description

S: Source operand

Function description

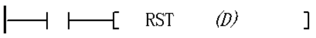
Setting the bit element assigned by ***D*** when the energy flow is valid.

Application instance



LD M0
SET M1

5.1.25 RST: Coil reset instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: RST *(S)*	**Step length**	**1**

Operand	Type						Applicable soft element								Indexing
S	BOOL		Y	M	S	LM	SM		Dx.y		C	T			

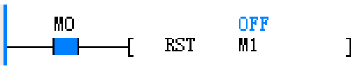
Operand description

S: Source operand

Function description

Resetting the specified bit element (***D***) to zero when the energy flow is valid.

Application instance

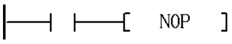


LD M0
RST M1

Note

If D is a C element, the corresponding count value is deleted. If D is a T element, the corresponding timing value is deleted.

5.1.26 NOP:No operation instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **NOP**	**Step length**	**1**

Function description

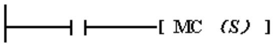
Performing no operation.

Note

In LAD, this instruction cannot be directly connected to the left energy flow bus.

5.2 Main control instructions

5.2.1 MC: Main control instruction

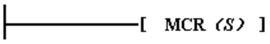
LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **MC *(S)***	**Step length**	**3**

Operand	Type	Applicable soft element													Indexing
S	INT	Constant													

Operand description

S: Source operand

5.2.2 MCR: Main control reset instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **MCR *(S)***	**Step length**	**1**

Operand	Type	Applicable soft element													Indexing
S	INT	Constant													

Operand description

S: Source operand

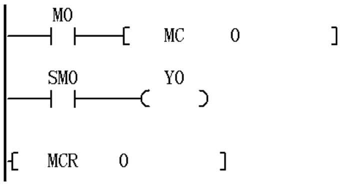
Function description

1. Matching the MC and MCR instructions to form an MC-MCR structure. The MC instruction represents the beginning of an MC-MCR structure, and its operand Sis the label of the MC-MCR structure. The value of operand S is a constantranging from 0 to 7. MCR represents the end of an MC-MCR structure.

2. Executing the instruction in the middle of the MC-MCR structure when the energy flow before the MC instruction is valid,

3. Skipping over the instruction in middle of the MC-MCR structure when the energy flow before the MC instruction is invalid, and executing it and deleting the destination operands corresponding to OUT, TON, TOF, PWM,HCNT, PLSY,PLSR, DHSCS, SPD, DHSCI, DHSCR, DHSZ,DHST, DHSP and BOUT in the structure after the program directly jumps to the structure

Application instance



LD M0
MC 0
LD SM0
OUT Y0
MCR 0

When M0 = ON, instructions in the MC 0–MCR 0 structure are executed, and Y0 = ON. When M0 = OFF,instructions in the MC 0–MCR 0 structure are not be executed,and the bit element Y0 designated by the destination operandof the OUT instruction in the structure is deleted, Y0 = OFF.

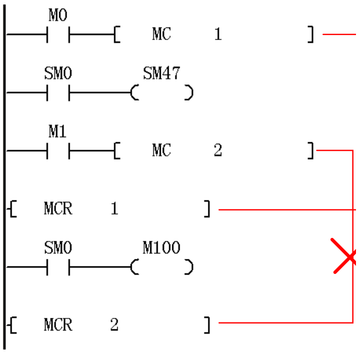
Note

1. In LAD, the MCR instruction must be directly connected to the left energy flow bus.

2. In LAD, the MCR instruction cannot connect in parallel or in series with other instructions.

3. Multiple different numbered MC-MCR structures can be used in the nested mode, but thenumber of nest levels cannot exceed 8. MC-MCR structures with the same number cannotbe used in the nested mode.

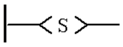
4. Two MC-MCR structures cannot be used in the cross manner. The use method shown in the following figure is illegal.



Note: It cannot be used in SFC programming.

5.3 SFC instructions

5.3.1 STL: SFC state loadinginstruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **STL *(S)***	**Step length**	**3**

Operand	Type	Applicable soft element													Indexing
S	BOOL						S								

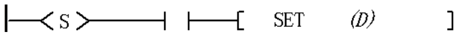
Operand description

S: Source operand

Function description

1. Indicating the beginning of the process of a step state (***S***).
2. Executing thebuilt-in instructions of a step when the state of the step is valid (ON).
3. If the state of a step changes from ON to OFF (falling edge change), not executing the built-in instructions of the step, and deleting the destination operands corresponding to OUT, TON, TOF, PWM, HCNT,PLSY, PLSR, DHSCS, SPD, DHSCI, DHSCR,DHSZ, DHST, DHSP and BOUT.
4. Not executing the built-in instructions of a step when the state of the step is invalid (OFF).
5. Defining a parallel merging structure by consecutive STL instructions (serial connectionof STL elements), where the STL instructions can be used in a maximum of 16 consecutive times, that is, the maximum number of branches in the parallel merging structure is 16.

5.3.2 SET Sxx: SFC state transition instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **SET *(D)***	**Step length**	**3**

Operand	Type	Applicable soft element													Indexing
D	BOOL						S								

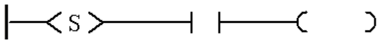
Operand description

D:Destination operand

Function description

Setting the state of the specified step (***D***) to be valid when the energy flow is valid, and setting the current valid step to be invalid to complete the state transition.

5.3.3 OUT Sxx: SFC state jumpinstruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **OUT *(D)***	**Step length**	**3**

Operand	Type	Applicable soft element													Indexing
D	BOOL						S								

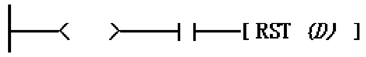
Operand description

D:Destination operand

Function description

Setting the state of the specified step (***D***) to be valid when the energy flow is valid, and setting the current valid step to be invalid to complete the state jump.

5.3.4 RST Sxx: SFC state reset instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **RST *(D)***	**Step length**	**3**

Operand	Type	Applicable soft element													Indexing
D	BOOL						S								

Operand description

D: Destination operand

Function description

Setting the state of the specified step (***D***) to be invalid when the energy flow is valid.

5.3.5 RET: SFC program segment end instruction

LAD: [RET]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **RET**	**Step length**	**1**

Function description

Indicating the end of a segment of the SFC program.

Note

It can be used only in the main program.

5.4 Timer instructions

5.4.1 TON: Turn-on delay timing instruction

LAD: [TON *(D)* *(S)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **TON** ***(D)*** ***(S)***	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
D	INT											T				
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

D: Destination operand

S: Source operand

Function description

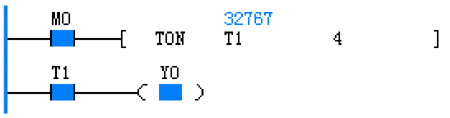
1. Timing the specified T element (D) when the energy flow is valid and the timing value is less than 32,767 (the timing value increases with the time), and remaining in 32,767 after it is reached.

2. Setting the timing coil output of the designated Telement to ON when the timing value is more than or equal to the preset value (S).

3. Stopping timing, resetting the timing value to zero, and setting the timing coil output to OFF when the energy flow is OFF.

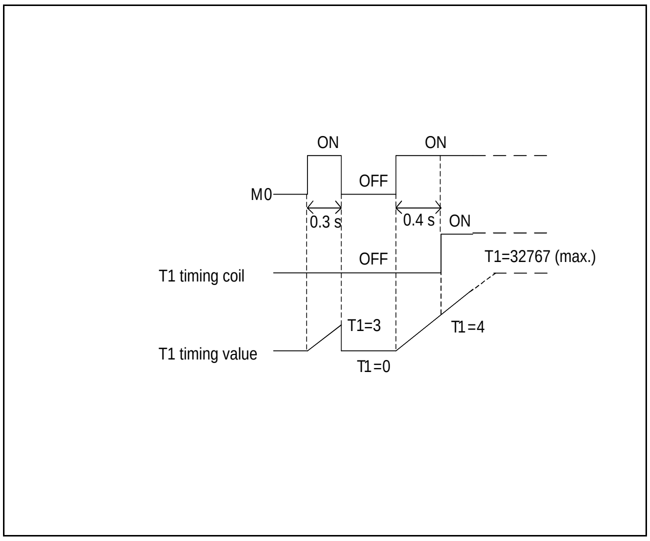
4. Setting the timing coil value of the designated T element to OFF and resetting the timing value to zero when the system executes the instruction for the first time.

Application instance



```
LD   M0
TON  T1  4
LD   T1
OUT  Y0
```

Sequence chart of the application instance



5.4.2 TONR: Memory-type turn-on delay timing instruction

LAD: [TONR *(D)* *(S)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **TONR** ***(D)*** ***(S)***	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
D	INT											T				

S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

D: Destination operand

S: Source operand

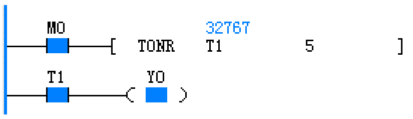
Function description

1. Timing the specified T element (***D***) when the energy flow is valid and the timing value is less than 32,767 (the timing value increases with the time), and remaining in 32,767 after it is reached.

2. Setting the timing coil output of the designated T element to ON when the timing value is more than or equal to the preset value (***S***).

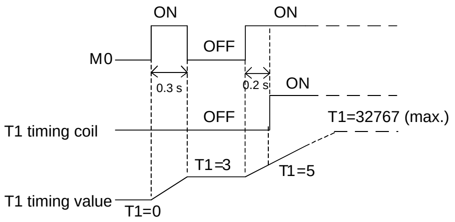
3. Stopping timing, and remaining the current timing coil and timing value when the energy flow is OFF.

Application instance



```
LD    M0
TONR        T1
5
LD    T1
OUT   Y0
```

Sequence chart of the application instance



5.4.3 TOF: Turn-off delay timing instruction

LAD: |—| |—[TOF *(D)* *(S)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: TOF *(D)* *(S)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
D	INT											T				
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

D: Destination operand

S: Source operand

Function description

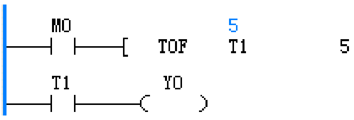
1. Starting the timing of the specified timer T (***D***) when the energy flow changes from ON to OFF (falling edge change).

2. Keeping timing when the energy flow is OFF but the specified timer T has been started, and setting the timing coil output of the T element to OFF when the timing value reaches the preset value (***S***), and then remaining in the preset value.

3. Not timing when the timing has not started but the energy flow input is OFF.

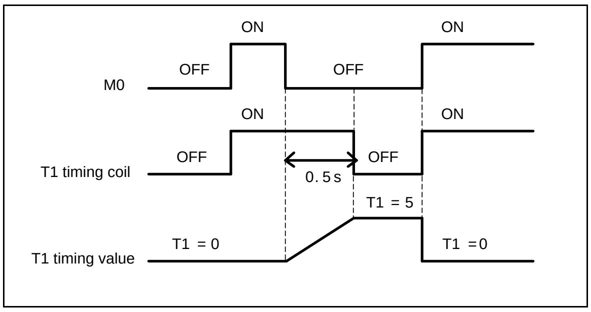
4. Stopping timing, resetting the timing value to zero, and setting the timing coil output to ON when the energy flow is ON.

Application instance

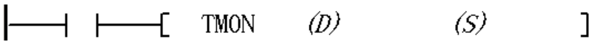


```
LD   M0
TOF       T1
5
LD   T1
OUT   Y0
```

Sequence chart of the application instance



5.4.4 TMON: Non-retriggerable monostable timing instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **TMON** ***(D) (S)***	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
D	INT											T				
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

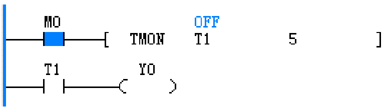
Operand description

D: Destination operand

S: Source operand

Function description

Application instance



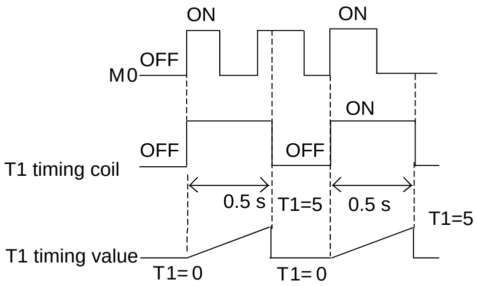
```
LD   M0
TMON   T1   5
LD   T1
OUT   Y0
```

Sequence chart of the application instance

1. Starting the timing of the designated timer T (***D***) (based on the current value) when the input energy flow changes from OFF to ON (rising edge change), and the timing has not started, and keeping the timing coil output ON when in the timing state (the timing length is determined by ***S***).

2. Keeping timing and keeping the timing coil output ON no matter how the energy flow changes when it is in the timing state (the timing length is determined by ***S***).

3. Stopping timing, resetting the timing value to zero, and setting the timing coil output to OFF when the timing value is reaches the preset point.



5.5 Counter instructions

5.5.1 CTU: 16-bit increment counterinstruction

LAD: ├─┤ ├─[CTU (D) (S)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: CTU *(D) (S)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
D	INT											C				
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

D: Destination operand

S: Source operand

Function description

1. Increasing the count value of the specified16-bit counter C (D) by 1 when the input energy flow changes from OFF to ON (rising edge change).

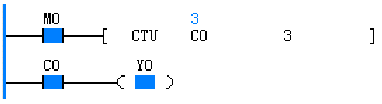
2. Remaining in the count value when it reaches 32,767.

3. Setting the counting coil to ON when the count value is larger than or equal to the preset point (***S***).

Note

Setting the address of the 16-bit counter C (***D***) to a value ranging from C0 to C199.

Application instance



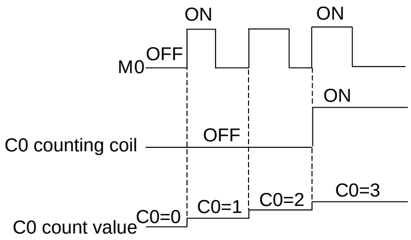
LD M0

CTU C0 3

LD C0

OUT Y0

Time sequence diagram instance



5.5.2 CTR: 16-bit cycliccountinginstruction

LAD: ├─┤ ├─[CTR (D) (S)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: CTR *(D) (S)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
D	INT											C				

S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

D: Destination operand

S: Source operand

Function description

1. Increasing the count value of the specified 16-bit counter C (***D***) by 1 when the input energy flow changes from OFF to ON (rising edge change).

2. Setting the counting coil to ON, when the count value is equal to the preset point (***S***).

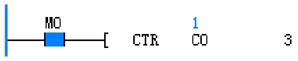
3. Setting the count value to 1, and the counting coil to OFF when the count value reaches the preset point (***S***),and the input energy flow changes from OFF to ON again (rising edge change).

Note

1. Performing no counting action when the preset count value (***S***) is less than or equal to 0.

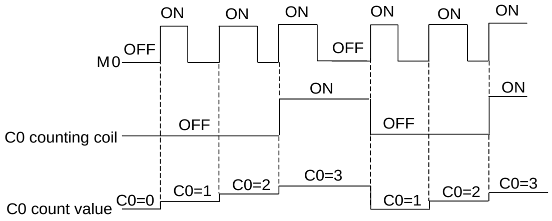
2. Setting the address of the 16-bit counter C (***D***) to a value ranging from C0 to C199.

Application instance



LD M0
CTR C0 3

Sequence diagram of the Application instance



5.5.3 DCNT:32-bit increment and decrement counting instruction

<table>
<tr><td colspan="9" rowspan="2">LAD:
|—| |—[DCNT <i>(D)</i> <i>(S)</i>]</td><td colspan="3">Applicable model</td><td colspan="5">VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="5"></td></tr>
<tr><td colspan="9">IL: DCNT <i>(D)</i> <i>(S)</i></td><td colspan="3">Step length</td><td colspan="5">7</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>D</td><td>DINT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>C</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>S</td><td>DINT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
</table>

Operand description

D: Destination operand

S: Source operand

Function description

1. Increasing or decreasing the count value of the specified 32-bit counter C (***D)*** by 1 when the input energy flow changes from OFF to ON (rising edge change) (the increase and decrease depend on the corresponding SM flag bit).

2. For increment counters, setting the counting coil to ON when the count value is larger than or equal to the preset point (***S***).

3. For decrement counters, setting the counting coil to OFF when the count value is less than or equal to the preset point (***S***).

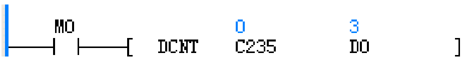
4. Changing the count value to –2147483648 when the count value is 2147483647 and is increased by 1.

5. Changing the count value to 2147483647 when the count value is –2147483648 and is decreased by 1.

Note

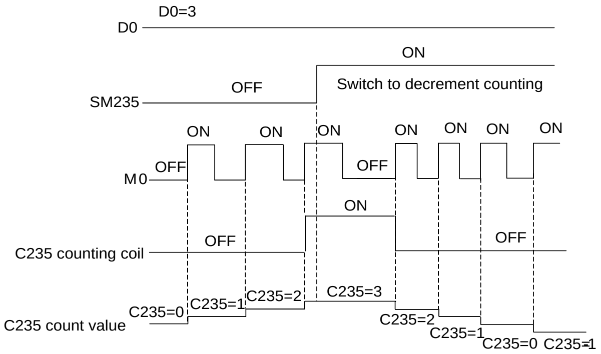
The address of C element (***D***) must be within the range of C200 to C235.

Application instance


M0
DCNT
0
C235
3
D0

```
LD    M0
DCNT    C235
D0
```

Sequence diagram of the application instance


D0=3
D0
ON
OFF
Switch to decrement counting
SM235
ON
ON
ON
ON
ON
ON
ON
OFF
OFF
M0
ON
OFF
OFF
C235 counting coil
C235=0
C235=1
C235=2
C235=3
C235=2
C235=1
C235=0
C235=1
C235 count value

Chapter 6 Application instructions

This chapter detailedly describes the application instructions of VC series micro-PLCs, including the instruction format (form), operand, influenced flag bit, function, application instance, and sequence diagram.

Chapter 6 Application instructions 79

- 6.1 Program flow control instructions..85
  - 6.1.1 FOR: Cycle instruction 85
  - 6.1.2 NEXT: Cycle return instruction 85
  - 6.1.3 LBL: Jump label definition instruction..86
  - 6.1.4 CJ: Conditional jump instruction..88
  - 6.1.5 CFEND: Instruction for conditional return of user main program..88
  - 6.1.6 WDT: Instruction for watchdog reset of user program..89
  - 6.1.7 EI: Enable interrupt instruction 89
  - 6.1.8 DI: Disable interrupt instruction 89
  - 6.1.9 CIRET: Instruction for conditional return of user interrupt program..89
  - 6.1.10 STOP: Instruction for stopping the user program 89
  - 6.1.11 CALL: Instruction for calling the user subprogram 90
  - 6.1.12 CSRET: Instruction for conditional return of user subprogram..90
- 6.2 Data transmission instructions..91
  - 6.2.1 MOV: Word data transmissioninstruction 91
  - 6.2.2 DMOV: Double word data transmission instruction 91
  - 6.2.3 RMOV: Floating-point data transmission instruction 92
  - 6.2.4 BMOV: Block data transmission instruction..92
  - 6.2.5 FMOV: Data block fill instruction 92
  - 6.2.6 DFMOV: Data block double word fill instruction 93
  - 6.2.7 SWAP: MSB/LSB swop instruction 93
  - 6.2.8 XCH: Word swop instruction 94
  - 6.2.9 DXCH: Double word swop instruction 94
  - 6.2.10 PUSH: Data push instruction..94
  - 6.2.11 FIFO: First-in-first-out instruction..95
  - 6.2.12 LIFO: Last-in-first-out instruction 96
  - 6.2.13 WSFR: Word string shift right instruction..96
  - 6.2.14 WSFL: Word string shift left instruction 97
- 6.3 Integer arithmetic operation instructions..98
  - 6.3.1 ADD: Integer addition instruction..98
  - 6.3.2 SUB: Integer subtraction instruction..99
  - 6.3.3 MUL: Integer multiplication instruction 99
  - 6.3.4 DIV: Integer division instruction..99
  - 6.3.5 SQT: Instruction for extracting the square root of an integer..100
  - 6.3.6 INC: Integer plus one instruction..100
  - 6.3.7 DEC: Integer minus one instruction..101
  - 6.3.8 VABS: Instruction for obtaining the absolute value of an integer..101
  - 6.3.9 NEG: Integer negation instruction 102
  - 6.3.10 DADD: Long integer addition instruction 102
  - 6.3.11 DSUB: Long integer subtraction instruction..102
  - 6.3.12 DMUL: Long integer multiplication instruction 103
  - 6.3.13 DDIV: Long integer division instruction 103
  - 6.3.14 DSQT: Instruction for extracting the square root of a long integer 104

6.3.15 DINC: Long integer plus one instruction........104
6.3.16 DDEC: Long integer minus one instruction........105
6.3.17 DVABS: Instruction for obtaining the absolute value of a long integer........105
6.3.18 DNEG: Long integer negation instruction........106
6.3.19 SUM: Integeraccumulation instruction........106
6.3.20 DSUM: Long integer accumulation instruction........108
6.4 Floating-point arithmetic operation instructions........108
6.4.1 RADD: Floating-point numberaddition instruction........108
6.4.2 RSUB: Floating-point numbersubtraction instruction........109
6.4.3 RMUL: Floating-point numbermultiplication instruction........109
6.4.4 RDIV: Floating-point numberdivision instruction........110
6.4.5 RSQT: Instruction for extracting the square root of a floating-point number........110
6.4.6 RVABS: Instruction for obtaining the absolute value of a floating-point number........111
6.4.7 RNEG: Floating-point number negation instruction........111
6.4.8 SIN: Instruction for obtaining SIN of a floating-point number........111
6.4.9 COS: Instruction for obtaining COS of a floating-point number........112
6.4.10 TAN: Instruction for obtaining TAN of a floating-point number........112
6.4.11 POWER: Instruction for exponentiation of a floating-point number........113
6.4.12 LN: Instruction for obtaining the natural logarithm of a floating-point number........113
6.4.13 EXP: Instruction for obtaining the natural number power of a floating-point number........114
6.4.14 RSUM: Floating-point number accumulation instruction........114
6.4.15 ASIN: Instruction for obtaining ASIN of a floating-point number........115
6.4.16 ACOS: Instruction for obtaining ACOS of a floating-point number........115
6.4.17 ATAN: Instruction for obtaining ATAN of a floating-point number........115
6.4.18 LOG: Instruction for obtaining the common logarithm of a floating-point number........116
6.4.19 RAD: Instruction for floating-point number angle-radian conversion........116
6.4.20 DEG: Instruction for floating-point number radian-angle conversion........116
6.5 Value conversion instructions........117
6.5.1 DTI: Instruction for converting a long integer to an integer........117
6.5.2 ITD: Instruction for converting an integer to a long integer........117
6.5.3 FLT: Instruction for converting an integer to a floating-point number........118
6.5.4 DFLT: Instruction for converting a long integer to a floating-point number........118
6.5.5 INT: Instruction for converting a floating-point number to an integer........118
6.5.6 DINT: Instruction for convert a floating-point number to a long integer........119
6.5.7 BCD: Instruction for converting a word to a 16-bit BCD code........119
6.5.8 DBCD: Instruction for converting a double word to a 32-bit BCD code........120
6.5.9 BIN: Instruction for converting a 16-bit BCD code to a word........120
6.5.10 DBIN: Instruction for converting a 32-bit BCD code to a double word........121
6.5.11 GRY: Instruction for converting a word to a 16-bit gray code........121
6.5.12 DGRY: Instruction for converting a double word to a 32-bit gray code........121
6.5.13 GBIN: Instruction for converting a 16-bit gray code to a word........122
6.5.14 DGBIN: Instruction for converting a 32-bit gray code to a double word........122
6.5.15 SEG: Instruction for converting a word to a 7-segment code........123
6.5.16 ASC: ASCII code conversion instruction........123
6.5.17 ITA: Instruction for convertinga 16-bit hex data to an ASCII code........124
6.5.18 ATI: Instruction for converting an ASCII code to a 16-bit hex data........124
6.5.19 LCNV: Project conversion instruction........125
6.5.20 RLCNV: Floating-point project conversion instruction........126
6.6 Word logic operation instructions........127
6.6.1 WAND: Word AND instruction........127

6.6.2 WOR: INT OR instruction 129
6.6.3 WXOR: Word XOR instruction 129
6.6.4 WINV: Word INV instruction 130
6.6.5 DWAND: Double word AND instruction 130
6.6.6 DWOR: Double word OR instruction 130
6.6.7 DWXOR: Double word XOR instruction 131
6.6.8 DWINV: Double word negation instruction 131
6.7 Bit shift and rotate instructions 132
6.7.1 ROR: 16-bit rotate right instruction 132
6.7.2 ROL: 16-bit rotate left instruction 132
6.7.3 RCR: Instruction for 16-bit rotate right with carry flag bit 133
6.7.4 RCL: Instruction for 16-bit rotate left with carry flag bit 134
6.7.5 DROR: 32-bit rotate right instruction 134
6.7.6 DROL: 32-bit rotate left instruction 135
6.7.7 DRCR: Instruction for 32-bit rotate right with carry flag bit 135
6.7.8 DRCL: Instruction for 32-bit rotate left with carry flag bit 136
6.7.9 SHR: 16-bit shift right instruction 136
6.7.10 SHL:16-bit shift left instruction 137
6.7.11 DSHR: 32-bit shift right instruction 138
6.7.12 DSHL: 32-bit shift left instruction 138
6.7.13 SFTR: Bit string shift right instruction 139
6.7.14 SFTL: Bit string shift left instruction 139
6.8 Peripheral instructions 141
6.8.1 FROM: Instruction for reading words from a special module buffer register 141
6.8.2 DFROM: Instruction for reading double words from a special module buffer register 141
6.8.3 TO: Instruction for writing words from a special module buffer register 142
6.8.4 DTO: Instruction for writing double words from a special module buffer register 143
6.8.5 VRRD: Instruction for readingthe value of an analog potentiometer 143
6.8.6 REFF: Instruction for setting input filtering constant 144
6.8.7 REF: Instruction for immediately refreshing I/O 144
6.8.8 EROMWR: EEPROM write instruction 145
6.8.9 PR: Printing instruction 145
6.8.10 TKY: Numeric key input instruction 146
6.9 Real-time clock instructions 148
6.9.1 TRD: Real-time clock read instruction 148
6.9.2 TWR: Real-time clock write instruction 149
6.9.3 TADD: Clock addition instruction 150
6.9.4 TSUB: Clock subtraction instruction 151
6.9.5 HOUR: Chronograph instruction 152
6.9.6 DCMP: (=、<、>、<>、>=、<=)Date comparison instruction 153
6.9.7 TCMP: (=、<、>、<>、>=、<=)Time comparison instruction 154
6.9.8 HTOS: Instruction for converting hour-minute-second data to seconds 155
6.9.9 STOH: Instruction for converting seconds to hour-minute-second data 155
6.10 High-speed I/O instructions 156
6.10.1 HCNT: Instruction for driving the high-speed counter 156
6.10.2 DHSCS: Instruction of setting the high-speed count comparison 156
6.10.3 DHSCI: Instruction for triggering interrupt based on comparison of high-speed count 157
6.10.4 DHSPI:Triggering interrupt Instruction based on comparison of absolute high-speed output positions 159
6.10.5 DHSCR: Instruction for resetting the high-speed count comparison 160
6.10.6 DHSZ: High-speed count range comparison instruction 161

6.10.7 DHST: High-speed count table comparison instruction........162
6.10.8 DHSP: Instruction for pulse output based on high-speed count table comparison........164
6.10.9 SPD: Frequency measuring instruction........166
6.10.10 PLSY: High-speed pulse output instruction........167
6.10.11 PLSR: Instruction for count pulse output with acceleration/deceleration........168
6.10.12 PLS: Envelopepulse output instruction........171
6.10.13 PWM: Pulse output instruction........172
6.11 Control calculation instructions........174
6.11.1 PID: Function instruction........174
6.11.2 RAMP: Ramp signal output instruction........178
6.11.3 HACKLE: Sawtooth wave signal output instruction........179
6.11.4 TRIANGLE: Triangle wave signal output instruction........180
6.11.5 ABSD: Cam absolute control instruction........181
6.11.6 DABSD: Double word cam absolute control instruction........183
6.11.7 ALT: Alternate output instruction........184
6.12 Communication instructions........184
6.12.1 MODBUS: Master station communication instruction........184
6.12.2 XMT: Free-port sending instruction........185
6.12.3 RCV: Free-port receiving instruction........186
6.12.4 MODRW: Modbus read/write instruction........187
6.13 Check instructions........191
6.13.1 CCITT: Check instruction........191
6.13.2 CRC16: Check instruction........191
6.13.3 LRC: Check instruction........192
6.14 Enhanced bit processing instructions........193
6.14.1 ZRST: Instruction for resetting bits to 0 in batch........193
6.14.2 ZSET: Instruction for resetting bits in batch........193
6.14.3 DECO: Decoding instruction........193
6.14.4 ENCO: Encoding instruction........194
6.14.5 BITS: Instruction for counting on bit in word........194
6.14.6 DBITS: Instruction for counting on bit in double word........195
6.14.7 BON: Instruction for judging on bit in word........195
6.15 Word contact instructions........195
6.15.1 BLD: Word bit contact LD instruction........195
6.15.2 BLDI: Word bit contact LDI instruction........196
6.15.3 BAND: Word bit contact AND instruction........196
6.15.4 BANI: Word bit contact ANI instruction........197
6.15.5 BOR: Word bit contact OR instruction........197
6.15.6 BORI: Word bit contact ORI instruction........198
6.15.7 BOUT: Word bit coil output instruction........198
6.15.8 BSET: Word bit coil setinstruction........198
6.15.9 BRST: Word bit coil reset instruction........199
6.16 Comparison contact instructions........199
6.16.1 LD (=, <, >, <>, >=, <=): Integer comparison LD※instruction........199
6.16.2 AND (=, <, >, <>, >=, <=): Integer comparison AND ※instruction........200
6.16.3 OR (=, <, >, <>, >=, <=): Integer comparison OR※instruction........201
6.16.4 LDD (=, <, >, <>, >=, <=): Long integer comparison LDD※instruction........202
6.16.5 ANDD (=, <, >, <>, >=, <=): Long integer comparison ANDD※instruction........203

6.16.6 ORD (=, <, >, <>, >=, <=): Long integer comparison ORD※instruction....204
6.16.7 LDR: Floating-point number comparison instruction....205
6.16.8 ANDR: Floating-point number comparison instruction....206
6.16.9 ORR: Floating-point number comparison instruction....207
6.16.10 CMP: Instruction for setting integer comparison to ON....209
6.16.11 LCMP: Instruction for setting long integer comparison to ON....209
6.16.12 RCMP: Instruction for setting floating-point number comparison to ON....210
6.17 Bulk data processing instructions....210
6.17.1 BKADD: Bulk data addition operation instruction....210
6.17.2 BKSUB: Bulk data subtraction operation instruction....211
6.17.3 BKCMP=,>,<,<>,<=,>=: Bulk data comparison instruction....211
6.18 Datasheet instructions....212
6.18.1 LIMIT: Upper/lower limit control instruction....212
6.18.2 DBAND: Deadband control instruction....213
6.18.3 ZONE: Zone control instruction....213
6.18.4 SCL: Coordinatesetting instruction....214
6.18.5 SER: Data search instruction....215
6.19 Character string instructions....216
6.19.1 STRADD: String combination instruction....216
6.19.2 STRLEN: Instruction for detecting the string length....217
6.19.3 STRRIGHT: Instruction for reading a string from right....217
6.19.4 STRLEFT: Instruction for reading a string from left....218
6.19.5 STRMIDR: Instruction for reading any characters of a string....219
6.19.6 STRMIDW: Instruction for replacing any characters of a string....219
6.19.7 STRINSTR: String search instruction....220
6.19.8 STRMOV: String transmission instruction....221
6.20 Extension file register instructions....221
6.20.1 LOADR: Instruction for reading data from an extension file register....221
6.20.2 SAVER: Instruction for writing data to an extension file register....222
6.20.3 INITR: Instruction for initializing an extension register....223
6.20.4 LOGR: Instruction for logging on an extension register....224
6.20.5 INITER: Instruction for initializing an extension file register....225
6.21 Positioning instructions....227
6.21.1 ZRN: Zero return instruction....227
6.21.2 PLSV:Variable speed pulse output instruction....228
6.21.3 DRVI: Relative position control instruction....229
6.21.4 DRVA: Absolute position control instruction....230
6.21.5 DSZR: Instruction for zero return with DOG....231
6.21.6 DVIT: Interrupt positioning instruction....232
6.21.7 STOPDV: Pulse output stop instruction....233
6.21.8 LIN: Linear trajectory interpolation instruction....234
6.21.9 CW: Clockwise arc trajectory interpolation....237
6.21.10 CCW: Counterclockwise arc trajectory interpolation instruction....238
6.21.11 MOVELINK: Synchronous control instruction....241
6.21.12 GEARBOX: Electronic gear instruction....243
6.22 Data processing instructions....244
6.22.1 MEAN: Mean instruction....244
6.22.2 WTOB: Byte-unit data separation instruction....244
6.22.3 BTOW: Byte-unit data combination instruction....245
6.22.4 UNI: Instruction for combining 4bits of 16-bit data....246

6.22.5 DIS: Instruction for separating 4bitsof 16-bit data 247
6.22.6 ANS: Signal alarm set instruction..248
6.22.7 ANR: Signal alarm reset instruction 249
6.23 Other instructions 249
6.23.1 RND: Instruction for generating random numbers..249
6.23.2 DUTY: Instruction for generating timed pulses...250

6.1 Program flow control instructions

6.1.1 FOR: Cycle instruction

LAD: ├─┤ ├─[FOR *(S)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: FOR *(S)*	**Step length**	**3**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S: Source operand

6.1.2 NEXT: Cycle return instruction

LAD: ├[NEXT]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: NEXT	**Step length**	**1**

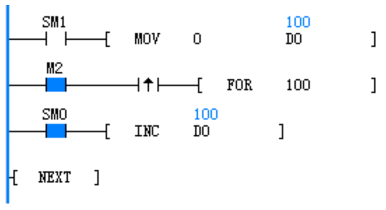
- Function description

  1. FOR and NEXT instructionsform a FOR-NEXT structure.

  2. Cyclically executing the instructions in the middle of the FOR-NEXT structure by S times continuously when the energy flow before FOR is valid and the cycle times (***S***) is larger than 0. Continuing to execute the instructions after the FOR-NEXT structure when the cyclic execution is done by S times.

  3. Not executing the instructions in the middle of the FOR-NEXT structure, directly jumping to the FOR-NEXT structure to continue the execution when the energy flow before FOR is invalid, or the cycle times (***S***) is less than or equal to zero.

- Application instance

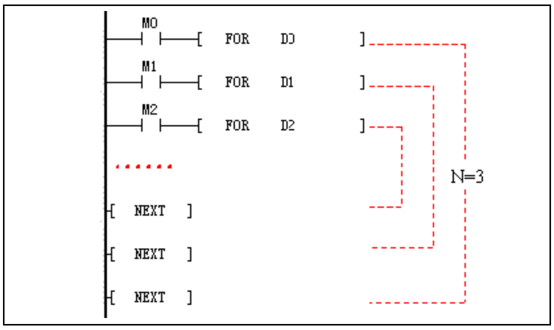
 SM1 MOV 0 100 D0 M2 FOR 100 SM0 100 INC D0 NEXT	LD SM1 MOV 0 D0 LD M2 EU FOR 100 LD SM0 INC D10 NEXT

The initial conditions of the operation are: D0=0, and M2=OFF. When M2 changes from OFF to ON, the instructions within the FOR-NEXT structure are consecutively executed for 100 times. D0 is increased by one for 100 times. When the cycle is over, D0=100.

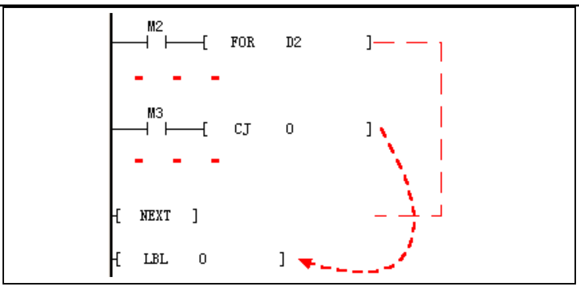
- Note

  1. The FOR-NEXT instruction must be used in pairs in a program body (POU), otherwise the user program cannot be compiled correctly.

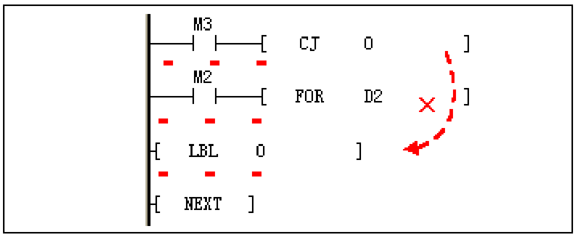
  2. Supporting the nesting of multiple FOR-NEXT structures. The CPU unit of the VC2 series supports the nesting of a maximum of 8 FOR-NEXT structures. (The following figure shows the nesting of 3 FOR-NEXT structures.)



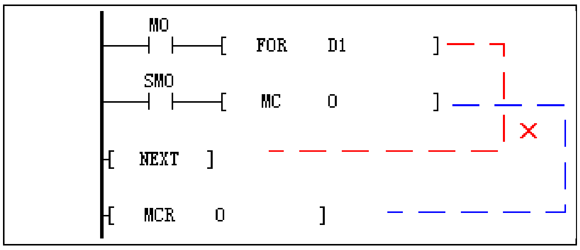
  3. You can use the conditional jump (CJ) instruction to jump out of the cycle body, so as to terminate the cycle body execution in advance, as shown in the LAD below:



  4. It is prohibited to use the CJ statement to jump into the next cycle body, and the LAD below cannot be compiled correctly:



  5. The intersection of the MC-MCR structure body and the FOR-NEXT structure body is prohibited, and the LAD below cannot be compiled correctly:



📖 Note

The execution of the FOR-NEXT cycle is time-consuming. The more the cycle times are, or the more the instructions are contained in the cycle body, the longer time it takes. To prevent the operation timeout error, you need to use the WDT instruction within a time-consuming cycle body.

6.1.3 LBL: Jump label definition instruction

LAD: |[LBL *(S)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **LBL** ***(S)***	**Step length**	**3**

Operand	Type	Applicable soft element													Indexing
S	INT	Constant													

- Operand description

  S:Labeling value

- Function description

  1. Defining a label whose labeling value is ***S***.

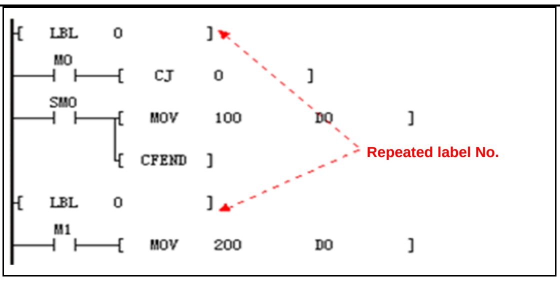
  2. Not used for actual operation. It only indicates the specific jumping position for the CJ instruction.

- Note

  1. Range of labeling value S: 0≤**S**≤127。

  2. Two labels with a repetitive definition are not allowed in one POU, otherwise the user program cannot be compiled. However, repetitive label definitionsare allowed in different POUs (for example, different subprograms).

- Application instance of wrong program


LBL 0
M0
CJ 0
SM0
MOV 100 D0
CFEND
LBL 0
M1
MOV 200 D0
Repeated label No.

6.1.4 CJ: Conditional jump instruction

LAD: ├─┤ ├─[CJ (S)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **CJ *(S)***	**Step length**	**3**

Operand	Type	Applicable soft element													Indexing
S	INT	Constant													

- Operand description

  S: Labeling value

- Function description

  1. When the energy flow is valid, the user program jumps to execute the instruction numbered ***S,*** a legal label.

  2. When the energy flow is invalid, there is no jump operation, and the system executes the instruction after CY in order.

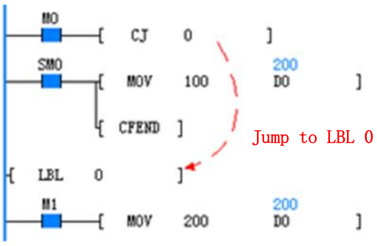
- Note

  1. The label ***S*** (0 ≤ S ≤ 127) to be jumped by the CJ instruction shall be a valid, and defined label, otherwise the user program cannot be compiled correctly.

  2. It is not allowed to use the CJ instruction to jump to a FOR-NEXT structure.

  3. It is allowable to use the CJ instruction to jump out of or jump into the MC-MCR structure or SFC state. However, such operation damages the logic of the MC-MCR structure and SFC state and makes the program complex. Thus, such operation is not recommended.

- Application instance



```
LD   M0
CJ   0
LD   SM0
MOV  100
D0
CFEND
LBL   0
LD   M1
MOV
200   D0
```

1. When the initial condition is M0=OFF and M1=ON, the CJ 0 instruction is not jumped, and D0=100. After executing the CFEND instruction, the program exits the main program in advance, instructions of LD M1 and MOV 200 D0 are not executed.

2. When M0=ON, M1=ON, the program executes the CJ 0 instruction, skips over the MOV 100 D0 and CFEND instructions. After jumping to the LBL 0 instruction, the program executes the MOV 200 D0 instruction, and now D0=200.

6.1.5 CFEND: Instruction for conditional return of user main program

LAD: ├─┤ ├─[CFEND]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **CFEND**	**Step length**	**1**

- Function description

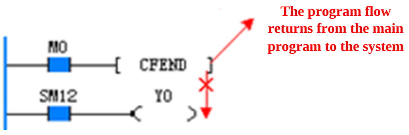
  1. When the energy flow of the instruction is valid, the main program returns to the system from the current scan cycle (the user main program is executed by the system repeatedly according to the scan cycle), and the instructionsin the subsequent main program are not executed.

  2. When the energy flow of the instruction is invalid, this instruction does not generate any action, and the subsequent instructions are executed in order.

- Note

  The CFEND instruction must be used in the user main program, otherwise the program cannot be compiled.

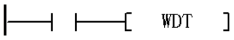
- Application instance


The program flow returns from the main program to the system
M0
CFEND
SM12
Y0

```
LD   M0
CFEND
LD
SM12
OUT   Y0
```

When the program is running, if M0=OFF, the CFEND instruction does not generate any action, the subsequent LD SM12 and OUT Y0 instructions are executed, and you can see the Y0 blinks periodically.If M0=ON, CFEND instruction generates an action, the program flow returns to the system from the main program in advance, the subsequent LD SM12 and OUT Y0 instructions are not be executed, and the Y0 no longer blinks periodically.

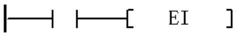
6.1.6 WDT: Instruction for watchdog reset of user program

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **WDT**	**Step length**	**1**

- Function description

  When the energy flow is valid, this instruction clears the timing value of the watchdog in the user program, and the watchdog of the system user program restarts the timing.

6.1.7 EI: Enable interrupt instruction

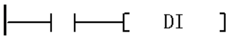
LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **EI**	**Step length**	**1**

- Function description

  1. When the energy flow is valid, the interrupts in the current scan cycle are enabled.

  2. When the EI instruction is valid, the interrupt request is allowed to be added into the interrupt request queue, waiting for the system response.

6.1.8 DI: Disable interrupt instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **DI**	**Step length**	**1**

- Function description

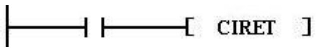
  1. When the energy flow is valid, the global interrupt enable flag is invalid,that is, the global interrupt is disabled.

  2. When the global interrupt enable flag is invalid, the interrupt events cannot generate any interrupt request.

- Note

  When the interrupt disabling request instruction is valid, the system still responds to the unprocessed interrupt requests in the interrupt request queue, but the new interrupt events cannot generate the interrupt requests.

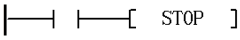
6.1.9 CIRET: Instruction for conditional return of user interrupt program

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **CIRET**	**Step length**	**1**

- Function description

  When the energy flow is valid, the system exits the current interrupt program in advance.

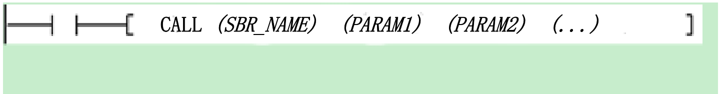
6.1.10 STOP: Instruction for stopping the user program

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **STOP**	**Step length**	**1**

- Function description

  When the energy flow is valid, the system stops the execution of the user program immediately.

6.1.11 CALL: Instruction for calling the user subprogram

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **CALL (subprogram name) (subprogram parameter 1) (subprogram parameter 2) …**	**Step length**	**Determined by the parameters carried by the subprogram**

- Function description
  When the energy flow is valid, the system calls the designated subprogram for execution, and after execution, returns to the instructions following the CALL instruction to continue the execution.

- Note
  1. The subprogram called by the CALL instruction needs to be defined in the user program in advance. If an undefined subprogram occurs in the CALL instruction, the program cannot be compiled.

  2. The element type of the operands in the CALL instruction needs to match with the data type defined in the local variable table of the subprogram, otherwise the program cannot be compiled.

  The following application instance demonstrates some illegal matches.

  Application instance 1: In the local variable table of the SBR1 subprogram, the data type of operand 1 is DINT/DWORD.

  The following usages are illegal:

  - CALL SBR1 Z0 (The data type of Z element cannot be DINT/DWORD)
  - CALL SBR1 C199 (The data type of C0 to C199 element cannot be DINT/DOWRD)
  - CALL SBR1 K2X0 (Kn addressing 1≤n≤3, the data type cannot be DINT/DWORD)

  Application instance 2: In the local variable table of the SBR1 subprogram, the data type of operand 1 is INT/WORD.

  The following usages are illegal:

  - CALL SBR1 C200 (The data type of C200 to C255 elements cannot be INT/WORD)
  - CALL SBR1 K2X0 (Kn addressing 4≤n≤8, the data type cannot be INT/WORD)

  3. The element type of the operands in the CALL instruction needs to match with the data type defined in the local variable table of the subprogram, otherwise the program cannot be compiled.

  The following application instance demonstrates some illegal matches.

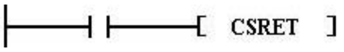
  Application instance: In the local variable table of the SBR1 subprogram, the data type of operand 1 is OUT or IN_OUT.

  The following usages are illegal:

  - CALL SBR1 321 (the constant cannot be changed, so it does not match with the OUT or IN_OUT type of the operand)
  - CALL SBR1 K4X0 (K4X0 is read-only, so it does not match with the OUT or IN_OUT type of the operand)
  - CALL SBR1 SD0 (SD0 is read only, so it does not match with the OUT or IN_OUT type of the operand)

  4. The number of the operands in the CALL instruction need to match with the local variable table of the subprogram, otherwise the program cannot be compiled.

6.1.12 CSRET: Instruction for conditional return of user subprogram

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **CSRET**	**Step length**	**1**

- Function description
  When the energy flow is valid, the system exits the current subprogram and return to the previous-level subprogram.

6.2 Data transmission instructions

6.2.1 MOV: Word data transmissioninstruction

LAD: ├─┤ ├─[MOV *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **MOV** ***(S)*** ***(D)***	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	Z	R	√

- Operand description

  S: Source operand

  D: Destination operand

- Function description

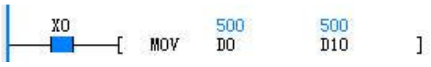
  Assigning the content of ***S*** to ***D*** and keeping the value of ***S*** when the energy flow is valid.

- Note

  1. MOV instruction supports two kinds of integers: one with symbol and the other without symbol. If the two operands of the instruction are both soft elements, then the data type is integer with symbol. If the source operand of the instruction is long integer with symbol, such as (－10, ＋100), then the destination operand is integer with symbol. If the source operand is long integer without symbol, such as (100, 45535), then the destination operand is also integer without symbol.

  2. The corresponding soft element C supports C0 to C199 only.

- Application instance



```
LD   X0
MOV   D0
D10
```

Assigning the content of D0 to D10 when X0=ON.In this case, D10=500.

6.2.2 DMOV: Double word data transmission instruction

LAD: ├─┤ ├─[DMOV *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **DMOV** ***(S)*** ***(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V	Z	R	√
D	DINT			KnY	KnM	KnS	KnLM		D	SD	C		V		R	√

- Operand description

  S: Source operand

  D: Destination operand

- Function description

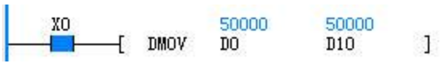
  Assigning the content of ***S*** to ***D*** and keeping the value of ***S*** when the energy flow is valid.

- Note

  1. DMOV instruction supports two kinds of long integers: one with symbol and the other without symbol. If the two operands of the instruction are both soft elements, then the data type is integer with symbol. If the source operand of the instruction is long integer with symbol, such as (－10, ＋100), then the destination operand is also integer with symbol. If the source operand is long integer without symbol, such as (100, 45535), then the destination operand is also integer without symbol.

  2. The corresponding soft element C supports 32-bit C element only.

- Application instance



LD X0
DMOV D0 D10

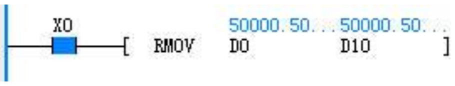
Assigning the content of (D0, D1) to (D10, D11) when X0=ON. In this case, (D10, D11)=50000.

6.2.3 RMOV: Floating-point data transmission instruction

LAD: ├─┤ ├─[RMOV *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: RMOV *(S)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description
  S: Source operand
  D: Destination operand
- Function description
  Assigning the content of ***S*** to ***D*** and keeping the value of ***S*** when the energy flow is valid.
- Application instance



LD X0
RMOV D0 D10

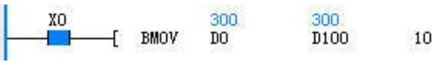
Assigning the content of (D0, D1) to (D10, D11) when X0=ON.In this case, (D10, D11)=50000.5.

6.2.4 BMOV: Block data transmission instruction

LAD: ├─┤ ├─[BMOV *(S1)* *(D)* *(S2)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: BMOV *(S1)* *(D)* *(S2)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V		R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description
  S1: Source operand, start unit of data block
  D: Destination operand, start unit of data block
  S2: Size of the data block
- Function description
  Assigning the content of ***S2*** unitsstarting from ***S1*** to ***S2*** units starting from ***D*** and keeping the content of ***S2*** units starting from ***S1*** when the energy flow is valid.
- Application instance



LD X0
BMOV D0 D100 10

Assigning the content of 10 units starting from D0 to 10 units starting from D100 when X0=ON. In this case, D100=D0, D101=D1, ……, D109=D9

6.2.5 FMOV: Data block fill instruction

LAD: ├─┤ ├─[FMOV *(S1)* *(D)* *(S2)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: FMOV *(S1)* *(D)* *(S2)*	**Step length**	**7**

Operand	Type	Applicable soft element	Indexing

S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S: Source operand, start unit of data block

  D: Destination operand, start unit of data block

  S2: Size of the data block

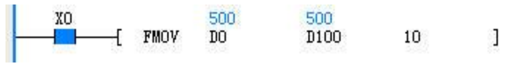
- Function description

  Fillingthe content of S1 in S2 units starting from D and keeping the content of S1 when the energy flow is valid.

- Note

  1. When ***S1, D***, and ***S2*** use C elements, the legal range is C0 to C199.
  2. ***S2*** is greater than 0.
  3. When ***S1*** and ***D*** are both Kn addressing, Kn needs to be equal.

- Application instance



LD X0

FMOV D0 D100 10

Filling the content of D0 in 10 units starting from D100 when X0=ON. In this case, D100=D101=……=D109=D0=500.

6.2.6 DFMOV: Data block double word fill instruction

LAD: ├─┤ ├─[DFMOV *(S1)* *(D)* *(S2)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DFMOV *(S1)* *(D)* *(S2)*	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Start of source operand

  D: Destination operand, start unit of data block

  S2: Size of the data block

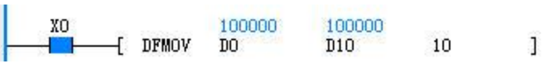
- Function description

  Fillingthe content of ***S1*** in ***S2*** units starting from ***D*** and keeping the content of ***S1*** when the energy flow is valid.

- Note

  1. When ***S1, D***, and ***S2*** use C elements, only 32-bit C elements are supported.
  2. ***S2*** is greater than 0.
  3. When ***S1*** and ***D*** are both Kn addressing, Kn needs to be equal.

- Application instance



LD X0

DFMOV D0 D10 10

Filling the content of (D0,D1) in10×2 units starting from D10 when X0=ON. In this case, (D10,D11)= (D12,D13)=……= (D28,D29)= (D0,D1)=100000.

6.2.7 SWAP: MSB/LSB swop instruction

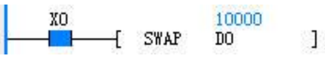
LAD: ├─┤ ├─[SWAP *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: SWAP *(D)*	**Step length**	**3**

Operand	Type	Applicable soft element														Indexing
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description

D: Destination operand, the word element whose most significant byte (MSB) and least significant byte (LSB) are swopped

- Function description
  Storing the value obtained after the MSB and LSB of the content of D are swopped into D when the energy flow is valid.
- Application instance



LD X0
SWAP D0

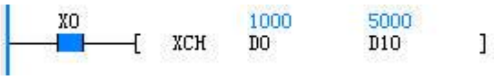
Storing the value obtained after the MSB and LSB of the content of D0=0x1027 (4135) are swapped into D0 when X0=ON. In this case, D0=0x2710 (10000).

6.2.8 XCH: Word swop instruction

LAD: XCH (D1) (D2)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **XCH *(D1) (D2)***	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
D1	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
D2	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description
  D1: Destination operand 1
  D2: Destination operand 2
- Function description
  Exchanging the content of ***D1*** with the content of ***D2*** and then storing the obtained value into ***D1*** and ***D2*** units when the energy flow is valid.
- Note
  When using the Kn addressing mode, the Kn in *D1* and *D2* should be the same.
- Application instance



LD X0
XCH D0 D10

Exchanging the content of D0 with D10 when X0=ON. Before execution: D0=5000, D10=1000.

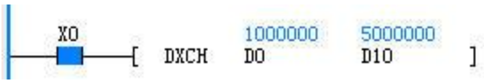
After execution: D0=1000, D10=5000.

6.2.9 DXCH: Double word swop instruction

LAD: DXCH (D1) (D2)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **DXCH *(D1) (D2)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
D1	DINT			KnY	KnM	KnS	KnLM		D		C	T	V		R	√
D2	DINT			KnY	KnM	KnS	KnLM		D		C	T	V		R	√

- Operand description
  D1: Destination operand 1;
  D2: Destination operand 2
- Function description
  Exchanging the content of ***D1*** with the content of ***D2*** and then storing the obtained value into ***D1*** and ***D2*** units when the energy flow is valid.
- Note
  When using the Kn addressing mode, the Kn in *D1* and *D2* should be the same.
- Application instance



LD X0
DXCH D0 D10

Exchanging the content of (D0,D1)with(D10,D11) when X0=ON. Before execution: (D0,D1) =5000000, (D10,D11) =1000000. After execution: (D0,D1) =1000000, (D10,D11) =5000000.

6.2.10 PUSH: Data push instruction

LAD: PUSH (S1) (D) (S2)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	

IL: **PUSH** ***(S1) (D) (S2)***								**Step length**			**7**					
Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT								D				V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Push value

  D: Number of elements in the storage stack. An element No. indicates the position of a stack bottom.

  S2: Size of the stack

- Function description

  1. Pushing the value of ***S1*** into the stack top whose stack bottom is***D*** and increasing the value of ***D*** by 1 when the energy flow is valid. In this case, the No. of the stack top unitis No. of ***D***＋the value of ***D***.

  2. Setting the operation carry flag bit (SM81) to 1, and not executing the push operation when the value of ***D*** is equal to the value of ***S2*** and there are still push instructions being executed.

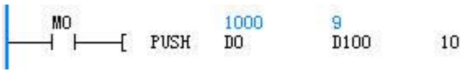
- Note

  1. When the stack is defined to be illegal (for example, when the size of the stack is no more than 0, the number of elements in the stack is less than 0 or the size of the stack is beyond the limit), the system reportsan error, indicating the definition of the stack operated is invalid.

  2. The size of the stack does not include the stack bottom element(the element designated by ***D***).

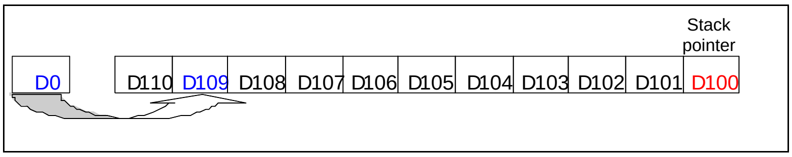
  3. ***S2*** indicates the size of the stack and the range is greater than 0.

- Application instance



LD M0

PUSH D0 D100 10



1. Pushing the content of D0 into the stack whose stack bottom is D100 when M0=ON.

2. Before execution: D0=1000, D100=8, D109=0.

3. After execution: D0=1000, D100=9, D109=1000.

6.2.11 FIFO: First-in-first-out instruction

LAD: ├┤ ├─[FIFO *(D1)* *(D2)* *(S)*]								**Applicable model**			VC1S VC1 VC2 VC3 VC5					
								Influenced flag bit								
IL: **FIFO** ***(D1) (D2) (S)***								**Step length**			**7**					
Operand	Type	Applicable soft element														Indexing
D1	INT								D				V		R	√
D2	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  D1: Number of elements in the stack, and the initial element of the stack is its element number plus 1.

  D2: Storage register for popped value

  S: Size of the queue

- Function description

  1. Assigning the initial value of the word stack starting from ***D1***(the content of the unit after ***D1***) to ***D2***, subtracting the value of ***D1*** by 1, moving the content of ***S*** units after ***D1*** forward and filling 0 in the last unit when the energy flow is valid.

  2. Setting the zero flag bit (SM80) to 1 when D1 is 0, that is, the stack is empty.

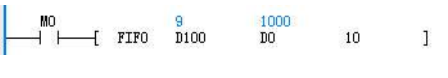
- Note

  1. When the stack is defined to be illegal (for example, when the size of the stack is no more than 0, the number of elements in the stack is less than 0 or the size of the stack is beyond the limit), the system reports an error, indicating the definition of the stack operated is invalid.

  2. The size of the stack does not include the stack bottom element(the element designated by ***D1***).

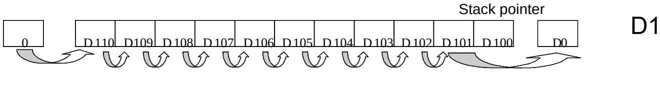
  3. ***S*** indicates the size of the stack and the range is greater than 0.

- Application instance



LD M0

FIFO D100 D0 10



1. Filling the content of D101in D0,movingthe content of D101 to D110forward, and filling 0 in D110 when M0=ON.

2. Before execution: D0=0, D100=10, D101=1000, D102=2000, ……, D109=9000, D110=10000.

3. After execution: D0=1000, D100=9, D101=2000, D102=3000, ……, D109=10000, D110=0.

6.2.12 LIFO: Last-in-first-out instruction

LAD: LIFO *(D1)* *(D2)* *(S)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **LIFO** ***(D1)*** ***(D2)*** ***(S)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
D1	INT								D				V		R	√
D2	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  D1: Number of elements in the queue, and the initial element of the stack is its element number plus 1

  D2: Storage register for popped value

  S: Size of the queue

- Function description

  1. Assigning the content of the stack top with ***D1*** as the stack bottom to ***D2*** and subtracting the value of ***D1*** by 1 when the energy flow is valid.

  2. Setting the zero flag bit (SM80) to 1 when ***D1*** is 0, that is, the stack is empty.

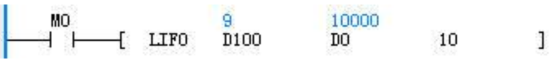
- Note

  1. When the stack is defined to be illegal (for example, when the size of the stack is no more than 0, the number of elements in the stack is less than 0 or the size of the stack is beyond the limit), the system reportsan error, indicating the definition of the stack operated is invalid

  2. The size of the stack does not include the stack bottom element(the element designated by ***D1***).

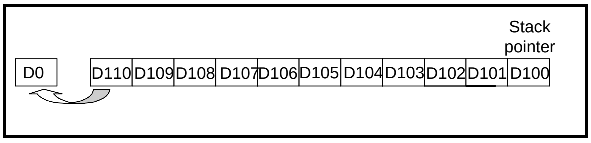
  3. ***S*** indicates the size of the stack and the range is greater than 0.

- Application instance



LD M0

LIFO D100 D0 10



1. Assigning the content of D110 to D0 and keeping the content of D101 to D110 when M0=ON.

2. Before execution: D0=0, D100=10, D101=1000, D102=2000, ……, D109=9000, D110=10000.

3. After execution: D0=10000, D100=9,D101=1000, D102=2000, ……, D109=9000, D110=10000.

6.2.13 WSFR: Word string shift right instruction

LAD: WSFR *(S1)* *(D)* *(S2)* *(S3)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Carry flag, borrow flag**
IL: **WSFR** ***(S1)*** ***(D)*** ***(S2)*** ***(S3)***	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V		R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand

  D: Destination operand, start element of a word string

  S2: Size of destination word queue

  S3: Number of words filled when shifting a word string rightward

- Function description

  Shifting the content of ***S2*** units starting from *D*rightward by ***S3*** units in the unit of word, discarding the ***S3*** data on the rightmost side, and shifting the content of ***S3*** unitsstarting from ***S1*** to the left end of the word string when the energy flow is valid.

- Note

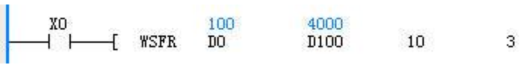
  1. Left and right order. The elements with small numbers are at the right, and the elements with large numbers are at the left.

  2.Both of ***S2*** and ***S3*** is no less than 0.

  3. ***S2*** is no less than ***S3***.

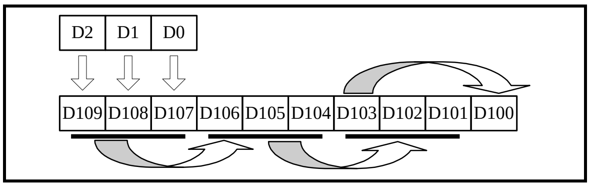
  4. When ***S1*** and ***D*** both use the Kn addressing, Kn needs to be the same.

- Application instance



LD X0

WSFR D0 D100 10 3

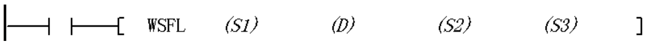


1. Shifting the content of 10 units starting from D100 rightward by three units in the unit of word, discarding the data of D102 to D100 units on the rightmost side, and shifting the content of the three units starting from D0 to the left end of the word string when M0=ON.

2. Before execution: D2=300, D1=200, D0=100.D109=10000, D108=9000, D107=8000, D106=7000, D105=6000, D104=5000, D103=4000, D102=3000, D101=2000, D100=1000.

3. After execution: keeping the content of D0 to D2. D2=300, D1=200, D0=100. D109=300, D108=200, D107=100, D106=10000, D105=9000, D104=8000, D103=7000, D102=6000, D101=5000, D100=4000.

6.2.14 WSFL: Word string shift left instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **WSFL *(S1) (D) (S2) (S3)***	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V		R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand

  D: Destination operand, start element of a word string

  S2: Size of destination word queue

  S3: Number of words filled when shifting a word string rightward

- Function description

  Shifting the content of ***S2*** units starting from *D* leftward by ***S3*** units in the unit of word, discarding the ***S3*** data on the leftmost side, and shifting the content of ***S3*** units starting from ***S1*** to the right end of the word string when the energy flow is valid.

- Note

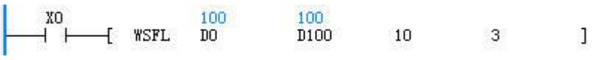
  1. Left and right order. The elements with small numbers are at the right, and the elements with large numbers are at the left.

  2.Both of ***S2*** and ***S3*** is no less than 0.

  3. ***S2*** is no less than ***S3***.

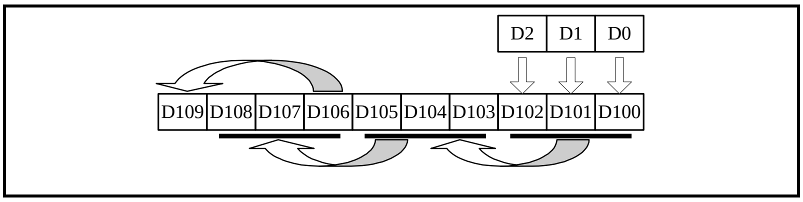
  4. When ***S1*** and ***D*** both use the Kn addressing, Kn needs to be the same.

- Application instance



LD X0

WSFL D0 D100 10 3



1. Shifting the content of 10 units starting from D100 leftward by three units in the unit of word, discarding the data of D109 to D107 units on the leftmost side, and shifting the content of the three units starting from D0 to the right end of the word string when X0=ON.

2. Before execution: D0=100, D1=200, D2=300. D109=10000, D108=9000, D107=8000, D106=7000, D105=6000, D104=5000, D103=4000, D102=3000, D101=2000, D100=1000.

3. After execution: Keeping the content of D0 to D2. D2=300, D1=200, D0=100. D109=7000, D108=6000, D107=5000, D106=4000, D105=3000, D104=2000, D103=1000, D102=300, D101=200, D100=100.

6.3 Integer arithmetic operation instructions

6.3.1 ADD: Integer addition instruction

LAD: ADD *(S1)* *(S2)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **ADD** ***(S1)*** ***(S2)*** ***(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

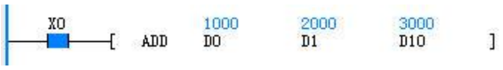
  D: Destination operand

- Function description

  1. Increasing ***S1*** by ***S2*** and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the carry flag bit (SM81) when the operation result (***D***) is larger than 32767. Setting the zero flag bit (SM80) when the operation result is 0. Setting the borrow flag bit (SM82) when the operation result is less than －32768.

- Application instance

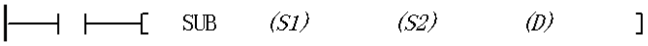


LD X0

ADD D0 D1 D10

Increasing D0 (1000)by D1 (2000), and assigning the operation result toD10 when X0=ON. In this case, D10=3000.

6.3.2 SUB: Integer subtraction instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **SUB *(S1)* *(S2)* *(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

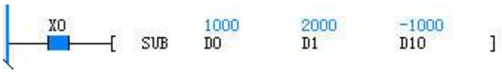
  D: Destination operand

- Function description

  1. Subtracting ***S1*** by ***S2*** and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the carry flag bit (SM81) when the operation result (***D***) is larger than 32767. Setting the zero flag bit (SM80) when the operation result is 0. Setting the borrow flag bit (SM82) when the operation result is less than －32768.

- Application instance

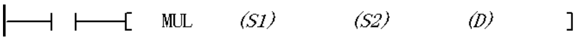


LD X0

SUB D0 D1 D10

SubtractingD0 (1000)by D1 (2000), and assigning the operation result toD10 when X0=ON. In this case, D10=－1000.

6.3.3 MUL: Integer multiplication instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **MUL *(S1)* *(S2)* *(D)***	**Step length**	**8**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

  D: Destination operand

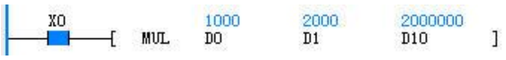
- Function description

  Multiplying ***S1*** by ***S2*** and assigning the operation result to ***D*** when the energy flow is valid.

- Note

  The operation result of MUL instruction is a 32-bit data.

- Application instance



LD X0

MUL D0 D1 D10

Multiplying D0 (1000) by D1 (2000), and assigning the operation result to (D10,D11) when X0=ON. In this case, (D10,D11)=2000000.

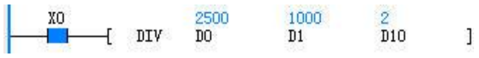
6.3.4 DIV: Integer division instruction

LAD:	**Applicable model**	VC1S VC1 VC2 VC3 VC5

├─┤ ├─[DIV (S1) (S2) (D)]	**Influenced flag bit**	
IL: DIV *(S1)* *(S2)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description
  S1: Source operand 1
  S2: Source operand 2
  D: Destination operand
- Function description
  Dividing ***S1*** by ***S2*** and assigning the operation result to ***D*** (D includes two units, one stores the quotient value and the other stores the residue value) when the energy flow is valid.
- Note
  If ***S2***≠0, the system reportsan error, indicating that 0 cannot be used as a divisor and does not execute the division operation.
- Application instance



LD X0
DIV D0 D1 D10

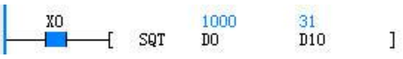
Dividing D0 (2500) by D1 (1000), and assigning the operation result to (D10,D11) when X0=ON. In this case, D10=2, D11=500.

6.3.5 SQT: Instruction for extracting the square root of an integer

LAD: ├─┤ ├─[SQT (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: SQT *(S)* *(D)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description
  S: Source operand
  D: Destination operand
- Function description
  1. Extracting the square root of ***S*** and assigning the operation result to ***D*** when the energy flow is valid.
  2. Setting the zero flag bit (SM80) When (***D***) is 0. Setting the carry flag bit (SM82) when the operation result is rounded off.
- Note
  If ***S*** ≥ 0, the system reports an operand error and does not execute the square root extraction.
- Application instance



LD X0
SQT D0 D10

Assigning the square root result of D0 (1000) to D10 when X0=ON. In this case, D10=31.

6.3.6 INC: Integer plus one instruction

LAD: ├─┤ ├─[INC (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: INC *(D)*	**Step length**	**3**

Operand	Type	Applicable soft element	Indexing

D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description

  D: Destination operand

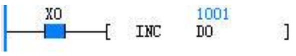
- Function description

  Increasing ***D*** by 1 automatically when the energy flow is valid.

- Note

  This instruction is a cyclic addition instruction, and the range is from -32768 to 32767. The supported range of C element is from C0 to C199.

- Application instance



LD X0

INC D0

Increasing D0 (1000) by 1 automatically when X0=ON. After execution, D0=1001.

6.3.7 DEC: Integer minus one instruction

<table>
<tr><td colspan="9">LAD:
├─┤ ├─[DEC (D)]</td><td colspan="3">Applicable model</td><td colspan="5">VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="9"></td><td colspan="3">Influenced flag bit</td><td colspan="5"></td></tr>
<tr><td colspan="9">IL: DEC <i>(D)</i></td><td colspan="3">Step length</td><td colspan="5">3</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>D</td><td>INT</td><td></td><td></td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td></td><td>D</td><td></td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
</table>

- Operand description

  D: Destination operand

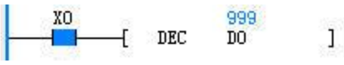
- Function description

  Decreasing D by 1 automaticallyWhen the energy flow is valid.

- Note

  This instruction is cyclic minus instruction, and the range is from -32768 to 32767.

- Application instance



LD X0

DEC D0

DecreasingD0 (1000) by 1 automatically when X0=ON. After execution, D0=999.

6.3.8 VABS: Instruction for obtaining the absolute value of an integer

<table>
<tr><td colspan="9">LAD:
├─┤ ├─[VABS (S) (D)]</td><td colspan="3">Applicable model</td><td colspan="5">VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="9"></td><td colspan="3">Influenced flag bit</td><td colspan="5"></td></tr>
<tr><td colspan="9">IL: VABS <i>(S)</i> <i>(D)</i></td><td colspan="3">Step length</td><td colspan="5">5</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S</td><td>INT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>INT</td><td></td><td></td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td></td><td>D</td><td></td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
</table>

- Operand description

  S: Source operand

  D: Destination operand

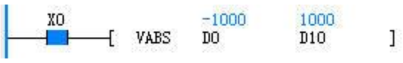
- Function description

  Taking an absolute value of S and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  The range of ***S*** needs to be from -32767 to 32767. When the value of ***S*** is -32768, the system reportsan error, indicating the value of the instruction operand is invalid, and the instruction does not generate any action.

- Application instance



LD X0

VABS D0 D10

Taking an absolute value of D0 (-1000), assigning the obtained result to D10 when X0=ON. In this case, D10=1000.

6.3.9 NEG: Integer negation instruction

LAD: ├─┤ ├──[NEG (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: NEG (S) (D)	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description

  S: Source operand

  D: Destination operand

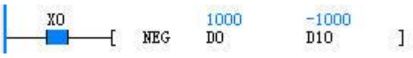
- Function description

  Taking the negative of ***S*** and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  The range of ***S*** needs to be from -32767 to 32767. When the value of ***S*** is -32768, the system reports an error, indicating the value of the instruction operand is invalid, and the instruction does not generate any action.

- Application instance



LD X0

NEG D0 D10

Taking the negative of D0 (1000), and assigning the obtained result to D10 when X0=ON. In this case, D10=-1000.

6.3.10 DADD: Long integer addition instruction

LAD: ├─┤ ├──[DADD (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: DADD (S1) (S2) (D)	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

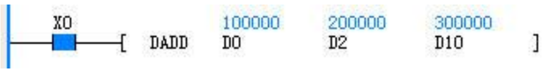
  D: Destination operand

- Function description

  1. Increasing ***S1*** by ***S2*** and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the carry flag bit (SM81) when the operation result (***D***) is larger than 2147483647. Setting the zero flag bit (SM80) when the operation result is 0. Setting the borrow flag bit (SM82) when the operation result is less than －2147483648.

- Application instance



LD X0

DADD D0 D2 D10

Increasingthe value (100000) of (D0,D1) bythe value (200000) of (D2,D3),and assigning the operation result to (D10,D11)when X0=ON. In this case, (D10,D11)=300000.

6.3.11 DSUB: Long integer subtraction instruction

LAD: ├─┤ ├──[DSUB (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: DSUB (S1) (S2) (D)	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

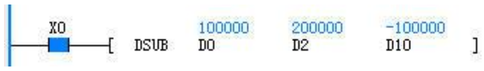
  D: Destination operand

- Function description

  1. Subtracting ***S1*** by ***S2*** and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the carry flag bit (SM81) when the operation result (***D***) is larger than 2147483647. Setting the zero flag bit (SM80) when the operation result is 0. Setting the borrow flag bit (SM82) when the operation result is less than −2147483648.

- Application instance



LD X0

DSUB D0 D2 D10

Subtractingthe value (100000) of (D0,D1) bythe value (200000) of (D2,D3), and assigning the operation result to (D10,D11)when X0=ON. In this case, (D10,D11)=−100000.

6.3.12 DMUL: Long integer multiplication instruction

LAD: |—| |—[DMUL (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DMUL *(S1)* *(S2)* *(D)*	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

  D: Destination operand

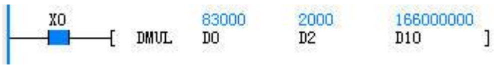
- Function description

  Multiplying ***S1*** by ***S2*** and assigning the operation result to ***D*** when the energy flow is valid.

- Note

  Please note that the operation result of DMUL instruction is a 32-bit data, which may cause an outflow.

- Application instance



LD X0

DMUL D0 D2 D10

Multiplying the value (83000) of (D0,D1) by the value (2000) of (D2,D3), and assigning the operation result to (D10,D11)when X0=ON. In this case, (D10,D11)=1660000000.

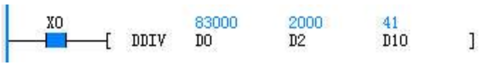
6.3.13 DDIV: Long integer division instruction

LAD: |—| |—[DDIV (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DDIV *(S1)* *(S2)* *(D)*	**Step length**	**10**

Operand	Type	Applicable soft element	Indexing

S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description
  S1: Source operand 1
  S2: Source operand 2
  D: Destination operand
- Function description
  Dividing***S1*** by ***S2*** and assigning the operation result to ***D*** (***D*** includes four units, the first two store the quotient value and the last two store the residue value) when the energy flow is valid.
- Note
  If *S2*≠0, the system reports an error, indicating that 0 cannot be used as a divisor and does not execute the division operation.
- Application instance



LD X0
DDIV D0 D2 D10

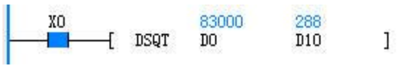
Dividing the value (83000) of (D0,D1) by the value (2000) of (D2,D3), and assigning the operation result to (D10,D11) and (D12,D13)when X0=ON. In this case, (D10,D11)=41 and (D12,D13)=1000.

6.3.14 DSQT: Instruction for extracting the square root of a long integer

LAD: ├─┤ ├─[DSQT *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: DSQT *(S)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description
  S: Source operand
  D: Destination operand
- Function description
  1. Extracting the square root of ***S*** and assigning the operation result to ***D*** when the energy flow is valid.
  2. Setting the zero flag bit (SM80) When (***D***) is 0. Setting the carry flag bit (SM82) when the operation result is rounded off.
- Note
  If ***S*** ≥ 0, the system reports an operand error and does not execute the square root extraction.
- Application instance



LD X0
DSQT D0
D10

Extracting the square root of the value (83000) of (D0,D1), and assigning the square root result to (D10,D11) when X0=ON. In this case, (D10,D11)=288.

6.3.15 DINC: Long integer plus one instruction

LAD: ├─┤ ├─[DINC *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DINC *(D)*	**Step length**	**4**

Operand	Type	Applicable soft element														Indexing
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  D: Destination operand

- Function description

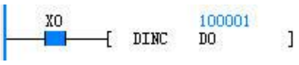
  Increasing ***D*** by 1 automatically when the energy flow is valid.

- Note

  1. This instruction is a cyclic addition instruction, and the range is from -2147483648 to 2147483647.

  2. Supporting 32-bit C element only among C elements.

- Application instance



LD X0

DINC D0

Increasing the value (100000) of (D0,D1)automatically when X0=ON. After execution,(D0,D1)=100001.

6.3.16 DDEC: Long integer minus one instruction

LAD: ├─┤ ├─[DDEC *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DDEC *(D)*	**Step length**	**4**

Operand	Type	Applicable soft element														Indexing
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  D: Destination operand

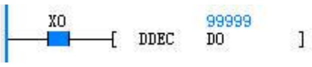
- Function description

  Decreasing ***D*** by 1 automatically When the energy flow is valid.

- Note

  This instruction is cyclic minus instruction, and the range is from -2147483648 to 2147483647.

- Application instance



LD X0

DDEC D0

Decreasing the value (100000) of (D0,D1) automatically when X0=ON. After execution, (D0,D1)=99999.

6.3.17 DVABS: Instruction for obtaining the absolute value of a long integer

LAD: ├─┤ ├─[DVABS *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: DVABS *(S)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S: Source operand

  D: Destination operand

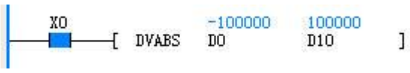
- Function description

  Taking an absolute value of S and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  The range of ***S*** needs to be from -2147483647 to 2147483647. When the value of ***S*** is -2147483648, the system an error, indicating the value of the instruction operand is invalid, andthe instruction does not generate any action.

- Application instance



LD X0

DVABS D0 D10

Taking an absolute value (－100000) of (D0,D1), and assigning the obtained result to (D10,D11) when X0=ON. In this case, (D10,D11)=100000.

6.3.18 DNEG: Long integer negation instruction

LAD: ├─┤ ├─[DNEG *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **DNEG** ***(S)*** ***(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S: Source operand

  D: Destination operand

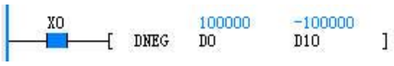
- Function description

  Taking the negative of ***S*** and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  The range of ***S*** needs to be from −2147483647 to 2147483647. When the value of ***S*** is −2147483648, the system an error, indicating the value of the instruction operand is invalid, and the instruction does not generate any action.

- Application instance



LD X0
DNEG D0 D10

Taking the negative of the value (100000) of (D0,D1), and assigning the obtained result to (D10,D11) when X0=ON. In this case, (D10,D11)=−100000.

6.3.19 SUM: Integeraccumulation instruction

LAD: ├─┤ ├─[SUM *(S1)* *(S2)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **SUM** ***(S1)*** ***(S2)*** ***(D)***	**Step length**	**8**

Operand	Type	Applicable soft element														Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S1: Source operand, start unit for accumulation

  S2: Source operand, number of pieces of data to be accumulated

  D: Destination operand, accumulation result

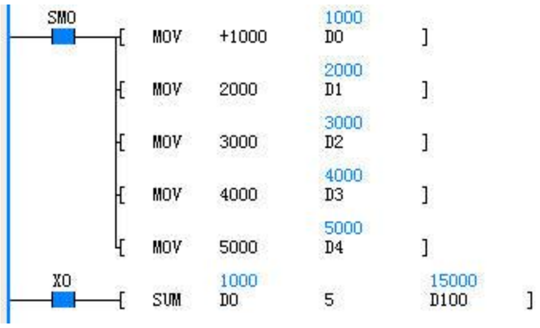
- Function description

  Accumulating the content of ***S2*** units starting from the start unit (***S1***), and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  1. The operation result of the SUM instructionis 32-bit data.
  2. If 0≤***S2***≤255, the system reports an operand error.
  3. Since ***D*** is a 32-bit data, the carry and borrow flags are always 0, while the zero flag bit is determined by the final accumulation result.

- Application instance



LD SM0
MOV 1000 D0
MOV 2000 D1
MOV 3000 D2
MOV 4000 D3
MOV 5000 D4
LD X0
SUM D0 5 D100

Accumulating the data of 5 units starting from D0, and assigning the obtained result to (D100,D101) when X0=ON.In this case, (D100,D101)=D0＋……＋D4=15000.

6.3.20 DSUM: Long integer accumulation instruction

LAD: ├─┤ ├──[DSUM (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **DSUM** ***(S1)*** ***(S2)*** ***(D)***	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	DINT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S1: Source operand, start unit for accumulation

  S2: Source operand, number of pieces of data to be accumulated

  D:Destination operand, accumulation result

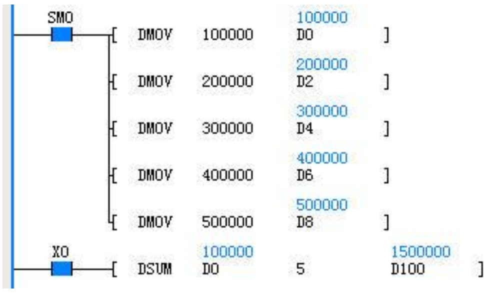
- Function description

  Accumulating the content of ***S2***×2 units starting from the start unit (***S1***), performing the DSUM instruction, and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  If 0≤***S2***≤255, the system reports an operand error.

- Application instance



LD SM0
DMOV 100000 D0
DMOV 200000 D2
DMOV 300000 D4
DMOV 400000 D6
DMOV 500000 D8
LD X0
DSUM D0 5 D100

Accumulating the data of 5×2 units starting from D0, and assigning the obtained result to (D100,D101) when X0=ON.

In this case, (D100,D101)= (D0,D1)＋......＋ (D8,D9)=1500000.

6.4 Floating-point arithmetic operation instructions

6.4.1 RADD: Floating-point numberaddition instruction

LAD: ├─┤ ├──[RADD (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **RADD** ***(S1)*** ***(S2)*** ***(D)***	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

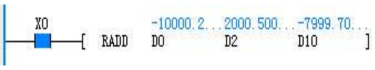
  D: Destination operand

- Function description

  1. Increasing ***S1*** by ***S2*** and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the carry flag bit (SM81) when the operation result (***D***) is larger than 1.701412e＋038 or less than －1.701412e＋038. Setting the zero flag bit (SM80) when the operation result is 0.

- Application instance



LD X0
RADD D0 D2 D10

Increasingthe value (-10000.2) of (D0,D1) bythe value (2000.5) of (D2,D3),and assigning the operation result to (D10,D11)when X0=ON. In this case, (D10,D11)=-7999.7.

6.4.2 RSUB: Floating-point numbersubtraction instruction

LAD: ├─┤ ├─[RSUB *(S1)* *(S2)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: RSUB *(S1)* *(S2)* *(D)*	**Step length**	**10**

Operand	Type	Applicable soft element													Indexing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

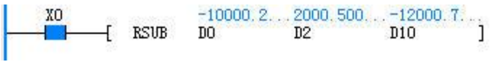
  D: Destination operand

- Function description

  1. Subtracting ***S1*** by ***S2*** and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the carry flag bit (SM81) when the operation result (***D***) is larger than 1.701412e＋038 or less than －1.701412e＋038. Setting the zero flag bit (SM80) when the operation result is 0.

- Application instance



LD X0

RSUB D0 D2 D10

Subtractingthe value (－10000.2) of (D0,D1) bythe value (2000.5) of (D2,D3), and assigning the operation result to (D10,D11)when X0=ON. In this case, (D10,D11)=－12000.7.

6.4.3 RMUL: Floating-point numbermultiplication instruction

LAD: ├─┤ ├─[RMUL *(S1)* *(S2)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: RMUL *(S1)* *(S2)* *(D)*	**Step length**	**10**

Operand	Type	Applicable soft element													Indexing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

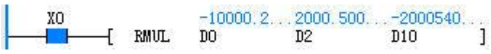
  D: Destination operand

- Function description

  1. Multiplying ***S1*** by ***S2*** and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the carry flag bit (SM81) when the operation result (D) is larger than 1.701412e＋038 or less than －1.701412e＋038. Setting the zero flag bit (SM80) when the operation result is 0.

- Application instance



LD X0

RMUL D0 D2 D10

Multiplying the value (－10000.2) of (D0,D1) by the value (2000.5)of (D2,D3), and assigning the operation result to (D10,D11)when X0=ON. In this case, (D10,D11)=-20005400.0 (actually the result is -20005400.1, but is rounded off according to the calculation precision).

6.4.4 RDIV: Floating-point numberdivision instruction

LAD: RDIV *(S1)* *(S2)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **RDIV *(S1)* *(S2)* *(D)***	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

  D: Destination operand

- Function description

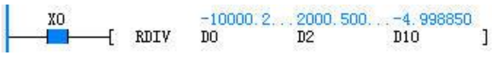
  1. Dividing ***S1*** by ***S2*** and assigning the operation result to ***D***.

  2. Setting the carry flag bit (SM81) when the operation result (D) is larger than 1.701412e＋038 or less than －1.701412e＋038. Setting the zero flag bit (SM80) when the operation result is 0.

- Note

  If *S2*≠0, the system reportsan error, indicating that 0 cannot be used as a divisor and does not execute the division operation.

- Application instance



LD X0

RDIV D0 D2 D10

Dividing the value (-10000.2) of (D0,D1) by the value (2000.5) of (D2,D3), and assigning the operation result to (D10,D11)when X0=ON.In this case, (D10,D11)=-4.998850.

6.4.5 RSQT: Instruction for extracting the square root of a floating-point number

LAD: RSQT *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **RSQT *(S)* *(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description

  S: Source operand

  D: Destination operand

- Function description

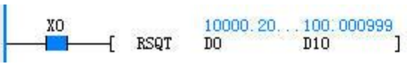
  1. Extracting the square root of ***S*** and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the zero flag bit (SM80) When (***D***) is 0.

- Note

  If ***S***≥0, the system reports an operand error and does not execute square root extraction.

- Application instance



LD X0

RSQT D0 D10

Extracting the square root of the value (10000.2)of (D0,D1), and assigning the square root result to (D10,D11)when X0=ON. In this case, (D10,D11)=100.000999.

6.4.6 RVABS: Instruction for obtaining the absolute value of a floating-point number

LAD: ├──┤ ├──[RVABS (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **RVABS** ***(S)*** ***(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description

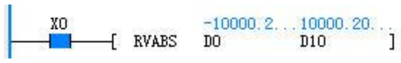
  S: Source operand

  D: Destination operand

- Function description

  Taking an absolute value of ***S*** and assigning the obtained result to ***D*** when the energy flow is valid.

- Application instance



LD X0

RVABS D0 D10

Taking an absolute value (-10000.2) of (D0,D1), and assigning the obtained result to (D10,D11) when X0=ON. In this case, (D10,D11)=10000.2.

6.4.7 RNEG: Floating-point number negation instruction

LAD: ├──┤ ├──[RNEG (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **RNEG** ***(S)*** ***(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description

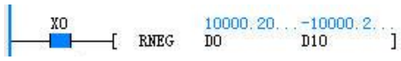
  S: Source operand

  D: Destination operand

- Function description

  Taking the negative of ***S*** and assigning the obtained result to ***D*** when the energy flow is valid.

- Application instance



LD X0

RNEG D0 D10

Taking the negative of the value (10000.2) of (D0,D1), and assigning the obtained result to (D10,D11) when X0=ON. In this case, (D10,D11)=-10000.2.

6.4.8 SIN: Instruction for obtaining SIN of a floating-point number

LAD: ├──┤ ├──[SIN (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **SIN** ***(S)*** ***(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description

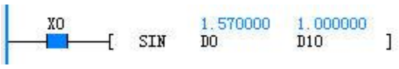
  S: Source operand

  D: Destination operand

- Function description

1. Obtaining the SIN value of ***S*** (unit: radian), and assigning the obtained result to ***D*** when the energy flow is valid.
2. Setting the zero flag bit (SM80) when the operation result (***D***) is 0.

- Application instance



LD X0

SIN D0 D10

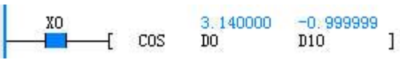
Taking the SIN value (1.57) of (D0,D1), and assigning the obtained result to (D10,D11) when X0=ON. In this case, (D10,D11) =1.

6.4.9 COS: Instruction for obtaining COS of a floating-point number

LAD: ├─┤ ├─[COS *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **COS *(S)* *(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description
  S: Source operand
  D: Destination operand
- Function description
  1. Obtaining the COS value of ***S*** (unit: radian), and assigning the obtained result to ***D*** when the energy flow is valid.
  2. Setting the zero flag bit (SM80) when the operation result (***D***) is 0.
- Application instance



LD X0

COS D0 D10

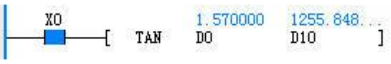
Taking the COS value (3.14) of (D0,D1), and assigning the obtained result to (D10,D11) when X0=ON. In this case, (D10,D11) =-0.999999.

6.4.10 TAN: Instruction for obtaining TAN of a floating-point number

LAD: ├─┤ ├─[TAN *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **TAN *(S)* *(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description
  S: Source operand
  D: Destination operand
- Function description
  1. Obtaining the TAN value of ***S*** (unit: radian), and assigning the obtained result to ***D*** when the energy flow is valid.
  2. Setting the carry flag bit (SM80) when the operation result (***D***) is larger than 1.701412e+038 or less than -1.701412e+038. Setting the zero flag bit (SM81) when the operation result is 0.
- Application instance



LD X0

TAN D0 D10

Taking the TAN value (1.57) of (D0,D1), and assigning the obtained result to (D10,D11) when X0=ON. In this case, (D10,D11) =1255.848398.

6.4.11 POWER: Instruction for exponentiation of a floating-point number

LAD: ├─┤ ├─[POWER (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: POWER *(S1) (S2) (D)*	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description

  S1: Source operand 1

  S2: Source operand 2

  D: Destination operand

- Function description

  1. Obtaining the ***S2*** power of ***S1***, and then assigning the operation result to ***D*** when the energy flow is valid.

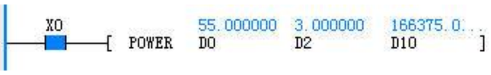
  2. Setting the carry flag bit (SM81) when the operation result (***D***) is larger than 1.701412e＋038 or less than -1.701412e＋038. Setting the zero flag bit (SM81) when the operation result is 0.

- Note

  1. If ***S1***=0 and ***S2***≤0, the system reportsan operand value error and does not execute the operation.

  2. If ***S1***＜0 and the mantissa of ***S2*** is not 0, the system reports an operand value error and does not executes the operation.

- Application instance



LD X0

POWER D0 D2 D10

Obtaining the (D2,D3) power of (D0,D1), namely (the 3.0 power of 55.0), and assigning the result to (D10,D11) when X0=ON. In this case, (D10,D11) =166375.0.

6.4.12 LN: Instruction for obtaining the natural logarithm of a floating-point number

LAD: ├─┤ ├─[LN (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: LN *(S) (D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- Operand description

  S: Source operand

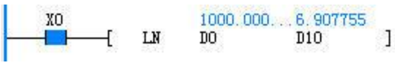
  D: Destination operand

- Function description

  1. Obtaining the LN value of ***S***, and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the carry flag bit (SM81) when the operation result (***D***) is larger than 1.701412e＋038 or less than -1.701412e＋038. Setting the zero flag bit (SM80) when the operation result is 0.

- Application instance



LD X0

LN D0 D10

Taking the LN value (1000.0) of (D0,D1),and assigning the obtained result to (D10,D11) when X0=ON. In this case, (D10,D11) =6.907755.

6.4.13 EXP: Instruction for obtaining the natural number power of a floating-point number

<table>
<tr><td colspan="9" rowspan="2">LAD:
├─┤ ├─[EXP <i>(S)</i> <i>(D)</i>]</td><td colspan="3">Applicable model</td><td colspan="4">VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="4">Zero flag, carry flag, and borrow flag</td></tr>
<tr><td colspan="9">IL: EXP <i>(S)</i> <i>(D)</i></td><td colspan="3">Step length</td><td colspan="4">7</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S</td><td>REAL</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>REAL</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
</table>

- Operand description

  S: Source operand

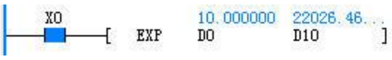
  D: Destination operand

- Function description

  1. Obtaining the EXP value of ***S***, and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the carry flag bit (SM81) when the operation result (***D***) is larger than 1.701412e+038 or less than -1.701412e+038. Setting the zero flag bit (SM80) when the operation result is 0.

- Application instance



```
LD   X0
EXP  D0  D10
```

Taking the EXP value (10.0) of (D0,D1), and assigning the obtained result to (D10,D11) when X0=ON. In this case, (D10,D11) =22026.464844.

6.4.14 RSUM: Floating-point number accumulation instruction

<table>
<tr><td colspan="9" rowspan="2">LAD:
├─┤ ├─[RSUM <i>(S1)</i> <i>(S2)</i> <i>(D)</i>]</td><td colspan="3">Applicable model</td><td colspan="4">VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="4">Zero flag, carry flag, and borrow flag</td></tr>
<tr><td colspan="9">IL: RSUM <i>(S1)</i> <i>(S2)</i> <i>(D)</i></td><td colspan="3">Step length</td><td colspan="4">9</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S1</td><td>REAL</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>S2</td><td>INT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td></td><td></td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>REAL</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
</table>

- Operand description

  S1: Source operand, start unit for accumulation

  S2:Source operand, number of pieces of data to be accumulated

  D:Destination operand,accumulation result

- Function description

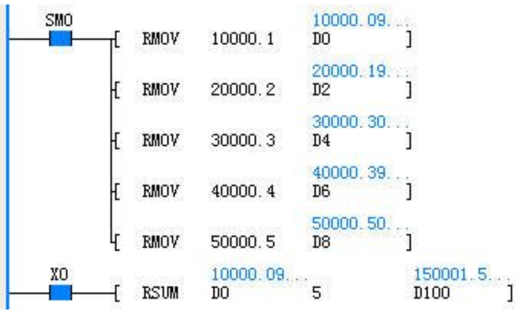
  Accumulating the content of ***S2***×2 units starting from the start unit (***S1***), performing the RSUM instruction, and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  1. If 0≤***S2***≤255, the system reports an operand error.

  2. The system does not perform the accumulation operation once an overflow occurs.

- Application instance



```
LD    SM0
RMOV  10000.1  D0
RMOV  20000.2  D2
RMOV  30000.3  D4
RMOV  40000.4  D6
RMOV  50000.5  D8
LD    X0
RSUM  D0  5  D100
```

Accumulating the data of 5×2 units starting from D0, and assigning the obtained result to (D100,D101) when X0=ON.

In this case, (D100,D101)= (D0,D1)+……+ (D8,D9)=150001.5.

6.4.15 ASIN: Instruction for obtaining ASIN of a floating-point number

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: ASIN (*S*) (*D*)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

- Operand description

  S: Source operand

  D: Destination operand

- Function description

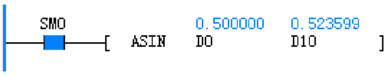
  1. Obtaining the SIN^{-1} value of ***S***, and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the zero flag bit (SM80) when the operation result (***D***) is 0.

- Note

  If S＞1 or S＜-1, the system reports an operand error, does not perform the conversion operation and keeps the content of ***D***.

- Application instance



LD SM0
ASIN D0 D10

Taking the SIN^{-1} value (0.500000) of (D0,D1),and assigning the obtained result to (D10,D11) when SM0=ON. In this case, (D10,D11) =0.523599.

6.4.16 ACOS: Instruction for obtaining ACOS of a floating-point number

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: ACOS (*S*) (*D*)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

- Operand description

  S: Source operand

  D: Destination operand

- Function description

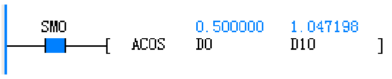
  1. Obtaining the COS^{-1} value of ***S***, and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the zero flag bit (SM80) when the operation result (***D***) is 0.

- Note

  If S＞1 or S＜-1, the system reports an operand error, does not perform the conversion operation and keeps the content of ***D***.

- Application instance



LD SM0
ACOS D0 D10

Taking the COS^{-1} value (0.500000) of (D0,D1), and assigning the obtained result to (D10,D11) when SM0=ON. In this case, (D10,D11) =1.047198.

6.4.17 ATAN: Instruction for obtaining ATAN of a floating-point number

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: ATAN (*S*) (*D*)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S	REAL	Constant							D				V	R	√

D	REAL								D				V	R	√

- Operand description

  S: Source operand

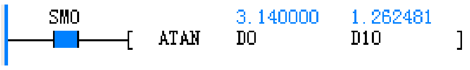
  D: Destination operand

- Function description

  1. Obtaining the TAN^{-1} value of ***S***, and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the zero flag bit (SM80) when the operation result (***D***) is 0.

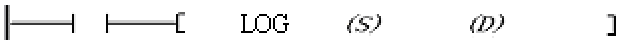
- Application instance



LD SM0
ATAN D0 D10

Taking the TAN^{-1} value (3.14)of (D0,D1), and assigning the obtained result to (D10,D11) when SM0=ON. In this case, (D10,D11) =1.262481.

6.4.18 LOG: Instruction for obtaining the common logarithm of a floating-point number

LAD: 	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: LOG (*S*) (*D*)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

- Operand description

  S: Source operand

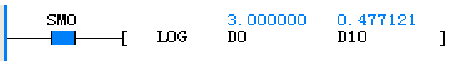
  D: Destination operand

- Function description

  1. Obtaining the LOG value of S, and assigning the operation result to D when the energy flow is valid. LOG is a common logarithm operation based on 10.

  2. Setting the carry (overflow) flag bit (SM81) when the operation result (D) overflows. Setting the zero flag bit (SM80) when the operation result is 0.

- Application instance



LD SM0
LOG D0 D10

Taking the value (3.0)of D0(D1), and assigning the obtained result to D10(D11)when SM0=ON. In this case, D10(D11) =0.477121.

6.4.19 RAD: Instruction for floating-point number angle-radian conversion

LAD:	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: RAD (*S*) (*D*)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

- Operand description

  S: Source operand

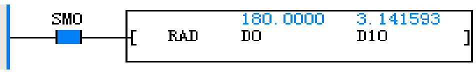
  D: Destination operand

- Function description

  1. Converting the angle value of ***S*** unit floating-point number into a radian value, and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the zero flag bit (SM80) when the operation result is 0.

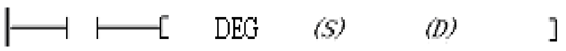
- Application instance



LD SM0
RAD D0 D10

Taking the value (180.0)of D0(D1), and assigning the obtained result to D10(D11)when SM0=ON. In this case, D10(D11) =3.141593.

6.4.20 DEG: Instruction for floating-point number radian-angle conversion

LAD: 	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**

IL: DEG (*S*) (*D*)														**Step length**	**7**
Operand	Type	Applicable soft element													Indexing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

- Operand description

  S: Source operand

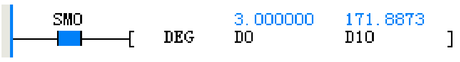
  D: Destination operand

- Function description

  1. Converting the radian value of ***S*** unit floating-point number into a angle value, and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the zero flag bit (SM80) when the operation result is 0.Setting the carry (overflow) flag bit (SM81) when the operation result (D) overflows.

- Application instance



LD SM0
DEG D0 D10

Taking the value (3.0)of D0(D1), and assigning the obtained result to D10(D11)when SM0=ON. In this case, D10(D11) =171.8873.

6.5 Value conversion instructions

6.5.1 DTI: Instruction for converting a long integer to an integer

LAD: ├──┤ ├──[DTI (*S*) (*D*)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **DTI** ***(S)*** ***(D)***	**Step length**	**6**

Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description

  S: Source operand

  D: Destination operand

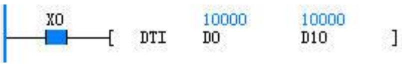
- Function description

  Converting ***S*** from a long integer to an integer, and assigning the operation result to ***D*** when the energy flow is valid.

- Note

  If ***S***>32767 or ***S***<-32768, the system reports an operand error, does not execute the conversion operation and keeps the content of ***D***.

- Application instance



LD X0
DTI D0 D10

Converting (D0, D1) =10000 from a long integer to an integer, and assigning the operation result to D10 when X0=ON. In this case, D10=10000.

6.5.2 ITD: Instruction for converting an integer to a long integer

LAD: ├──┤ ├──[ITD (*S*) (*D*)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **ITD** ***(S)*** ***(D)***	**Step length**	**6**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

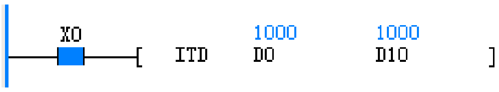
  S: Source operand

  D: Destination operand

- Function description

  Converting ***S*** from an integer to a long integer, and assigning the operation result to ***D*** when the energy flow is valid.

- Application instance



Converting D0=1000 from an integer to a long integer, and assigning the operation result to D10 when X0=ON. In this case, (D10,D11)=1000.

LD X0
ITD D0 D10

6.5.3 FLT: Instruction for converting an integer to a floating-point number

LAD: ├─┤ ├──[FLT *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **FLT** ***(S)*** ***(D)***	**Step length**	**6**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	REAL								D				V		R	√

- Operand description

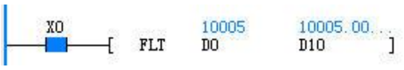
S: Source operand

D: Destination operand

- Function description

Converting ***S*** from an integer to a floating-point number, and assigning the operation result to ***D*** when the energy flow is valid.

- Application instance



LD X0
FLT D0 D10

Converting D0=10005 from an integer to a floating-point number, and assigning the operation result to (D10,D11) when X0=ON. In this case, (D10,D11)=10005.0.

6.5.4 DFLT: Instruction for converting a long integer to a floating-point number

LAD: ├─┤ ├──[DFLT *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **DFLT** ***(S)*** ***(D)***	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	REAL								D				V		R	√

- Operand description

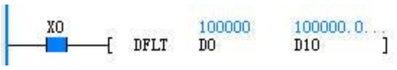
S: Source operand

D: Destination operand

- Function description

Converting ***S*** from a long integer to a floating-point number, and assigning the operation result to ***D*** when the energy flow is valid.

- Application instance



LD X0
DFLT D0 D10

Converting (D0,D1)=100000 from a long integer to a floating-point number, and assigning the operation result to (D10,D11) when X0=ON. In this case, (D10,D11)=100000.0.

6.5.5 INT: Instruction for converting a floating-point number to an integer

LAD: ├─┤ ├──[INT *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **INT** ***(S)*** ***(D)***	**Step length**	**6**

Operand	Type	Applicable soft element	Indexing

S	REAL	Constant							D				V		R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description

  S: Source operand

  D: Destination operand

- Function description

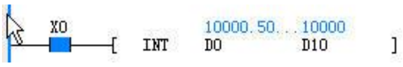
  1. Converting ***S*** from a floating-point number to an integer, and assigningthe operation result to ***D*** when the energy flow is valid.

  2. This instruction affects zero flagand borrow flag. Setting the zero flag bit(SM80) when the conversion result is 0. Setting the borrow flag when the decimals of the result are rounded off. Setting the carry flag bit (SM81) when the result exceeds the data range of the long integer data.

- Note

  When ***S***>32767, ***D***=32767. When ***S***<-32768, ***D***=-32768, and set the carry (overflow) flag bit(SM81).

- Application instance



LD X0
INT D0 D10

Converting (D0,D1)=10000.5from a floating-point number to an integer, and assigning the operation resultto D10 when X0=ON. In this case, (D0,D1) =10000.5.

6.5.6 DINT: Instruction for convert a floating-point number to a long integer

LAD: ├─┤ ├──[DINT *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: DINT *(S)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	REAL	Constant							D				V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S: Source operand

  D: Destination operand

- Function description

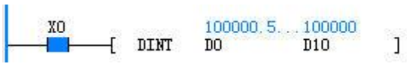
  1. Converting ***S*** from a floating-point number to a long integer, and assigning the operation result to ***D*** when the energy flow is valid.

  2. Setting the zero flag bit (SM80) when the conversion result is 0. Setting the borrow flag when the decimals of the result are rounded off. Setting the carry (overflow) flag bit (SM81) when the result exceeds the data range of the long integer data.

- Note

  When ***S***>2147483647, ***D***=2147483647. When ***S***<-2147483648, ***D***=-2147483648, and set the carry (overflow) flag (SM81) bitsimultaneously.

- Application instance



LD X0
DINT D0 D10

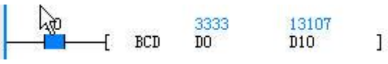
Converting (D0,D1)=100000.5from a floating-point number to a long integer, and assigning the operation resultto (D10,D11) when X0=ON. In this case, (D10,D11)=100000.

6.5.7 BCD: Instruction for converting a word to a 16-bit BCD code

LAD: ├─┤ ├──[BCD *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: BCD *(S)* *(D)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description
  S: Source operand, ≤9999
  D: Destination operand
- Function description
  Converting ***S*** from an integer to a 16-bit BCD code, and assigning the obtained result to ***D*** when the energy flow is valid.
- Note
  If ***S***＞9999, the system reports an operand error, does not execute the conversion operation, and keeps the content of ***D***.
- Application instance



LD X0
BCD D0 D10

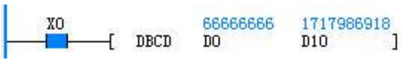
Converting D0=0x0D05 (3333) from an integer to a 16-bit BCD code, and assigning the operation resultto D10 when X0=ON. In this case, D10=0x3333 (13107).

6.5.8 DBCD: Instruction for converting a double word to a 32-bit BCD code

LAD: |[DBCD *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: DBCD *(S)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description
  S: Source operand, ≤99999999
  D: Destination operand
- Function description
  Converting ***S*** from a long integer to a 32-bit BCD code, and assigning the obtained result to ***D*** when the energy flow is valid.
- Note
  If ***S***＞99999999, the system reports an operand error, does not execute the conversion operation, and keeps the content of ***D***.
- Application instance



LD X0
DBCD D0 D10

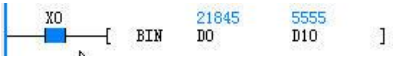
Converting (D0,D1)=0x3F940AA (66666666) from a long integer to a 32-bit BCD code, and assigning the operation resultto (D10,D11) when X0=ON. In this case, (D10,D11)=0x66666666 (1717986918).

6.5.9 BIN: Instruction for converting a 16-bit BCD code to a word

LAD: |[BIN *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: BIN *(S)* *(D)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description
  S: Source operand, its data format must comply with BCD code format
  D: Destination operand.
- Function description
  Converting ***S*** from a 16-bit BCD codeto an integer, and assigning the obtained result to ***D*** when the energy flow is valid.
- Note
  If the data format of ***S*** does not comply with BCD code format, the system reports an operand error, does not execute the conversion operation, and keeps the content of ***D***.
- Application instance



LD X0

BIN D0 D10

Converting D0=0x5555 (21845) from a 16-bit BCD code to an integer, and assigning the operation resultto D10 when X0=ON. In this case, D10=0x15B3 (5555).

6.5.10 DBIN: Instruction for converting a 32-bit BCD code to a double word

LAD: ├──┤ ├──[DBIN (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5	
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**	
IL: DBIN (S) (D)	**Step length**	**7**	

Operand	Type	Applicable soft element													Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S: Source operand

  D: Destination operand

- Function description

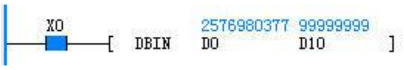
  1. Converting ***S*** from a 32-bit BCD codeto a long integer, and assigning the obtained result to ***D*** when the energy flow is valid.

  2. The data format of ***S*** must comply with BCD code format.

- Note

  If the data format of ***S*** does not comply with BCD code format, the system reports an operand error, does not execute the conversion operation, and keeps the content of ***D***.

- Application instance



LD X0

DBIN D0 D10

Converting (D0,D1)=0x99999999 (2576980377) from a 32-bit BCD code to a long integer, and assigning the operation resultto (D10,D11) when X0=ON. In this case, (D10,D11)=0x5F5E0FF (99999999).

6.5.11 GRY: Instruction for converting a word to a 16-bit gray code

LAD: ├[GRY (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5	
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**	
IL: GRY (S) (D)	**Step length**	**5**	

Operand	Type	Applicable soft element													Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description

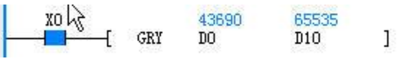
  S: Source operand

  D: Destination operand

- Function description

  Converting ***S*** from an integer to a 16-bit gray code, and assigning the obtained result to ***D*** when the energy flow is valid.

- Application instance



LD X0

GRY D0 D10

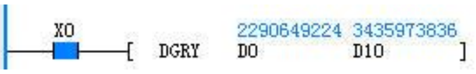
Converting D0=0xAAAA (43690) from an integer to a 16-bit gray code, and assigning the operation resultto D10 when X0=ON. In this case, D10=0xFFFF (65535).

6.5.12 DGRY: Instruction for converting a double word to a 32-bit gray code

LAD: ├──┤ ├──[DGRY (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**

IL: DGRY (S) (D)										**Step length**			**7**			
Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description
  S: Source operand
  D: Destination operand
- Function description
  Converting ***S*** from a long integer to a 32-bit gray code, and assigning the obtained result to ***D*** when the energy flow is valid.
- Application instance



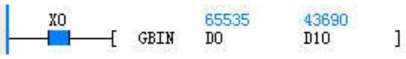
LD X0
DGRY D0 D10

Converting (D0,D1)=0x88888888 (2290649224) from a long integer to a 32-bit gray code, and assigning the operation resultto (D10,D11) when X0=ON. In this case, (D10,D11)=0xCCCCCCCC (3435973836).

6.5.13 GBIN: Instruction for converting a 16-bit gray code to a word

LAD: ├─┤ ├──[GBIN (S) (D)]										**Applicable model** **Influenced flag bit**			VC1S VC1 VC2 VC3 VC5 **Zero flag, carry flag, and borrow flag**			
IL: GBIN (S) (D)										**Step length**			**5**			
Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description
  S: Source operand
  D: Destination operand
- Function description
  Converting ***S*** from a 16-bit gray code to an integer, and assigning the obtained result to ***D*** when the energy flow is valid.
- Application instance



LD X0
GBIN D0 D10

Converting D0=0xFFFF (65535)from a 16-bit gray code to an integer, and assigning the operation resultto D10 when X0=ON. In this case, D10=0xAAAA (43690).

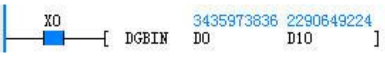
6.5.14 DGBIN: Instruction for converting a 32-bit gray code to a double word

LAD: ├─┤ ├──[DGBIN (S) (D)]										**Applicable model** **Influenced flag bit**			VC1S VC1 VC2 VC3 VC5 **Zero flag, carry flag, and borrow flag**			
IL: DGBIN (S) (D)										**Step length**			**7**			
Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description
  S: Source operand
  D: Destination operand
- Function description

Converting S from a 32-bit gray code to a long integer, and assigning the obtained result to ***D*** when the energy flow is valid.

- Application instance



LD X0
DGBIN D0 D10

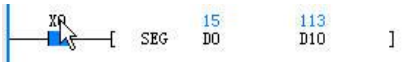
Converting (D0,D1)=0xCCCCCCCC (3435973836) from a 32-bit gray code to a long integer, and assigning the operation result to (D10,D11) when X0=ON. In this case, (D10,D11)=0x88888888 (2290649224).

6.5.15 SEG: Instruction for converting a word to a 7-segment code

LAD: —[SEG (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **SEG *(S) (D)***	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description
  S: Source operand, ***S***≤15
  D: Destination operand
- Function description
  Converting ***S*** from aninteger to a7-segment code, and assigning the obtained result to ***D*** when the energy flow is valid.
- Note
  If***S*** ＞ 15, the system reports an operand error, does not execute the conversion operation, and keeps the content of ***D***.
- Application instance



LD X0
SEG D0 D10

Converting D0=0x0F (15)from an integer to a 7-segment code, and assigning the operation result to D10 when X0=ON. In this case, D10=0x71 (113).

6.5.16 ASC: ASCII code conversion instruction

LAD: —[ASC (S1˜S8) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **ASC *(S1–S8) (D)***	**Step length**	**19**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant														
S2	INT	Constant														
S3	INT	Constant														
S4	INT	Constant														
S5	INT	Constant														
S6	INT	Constant														
S7	INT	Constant														
S8	INT	Constant														
D	INT								D		C	T	V	Z	R	√

- Operand description
  S1－***S8***: Source operand (If the number of characters is less than 8, the remaining characters are filled with 0)
  Only supporting the characters whose ASCII code is 0x21 to 0x7E (input through keyboard, if the number of characters is less than 8, the remaining characters are filled with 0)
  D: Destination operand
- Function description

Converting the character string ***S1*** – ***S8*** to ASCII code, and assigning the obtained result to the elements starting from ***D*** when the energy flow is valid. Storing two ASCII code data in the high/low byte of each ***D*** element when SM85=OFF. Storing one ASCII code data in low byte of each ***D*** element when SM85=ON.

- Application instance

M0
ASC 12345678 d0]
12849

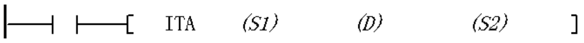
LD M0

ASC 12345678 D0

Executing the ASCII conversion when M0=ON and data is stored in two modes:

- If SM85=OFF, the execution result is: D0=0x3231, D1=0x3433, D2=0x3635, D3=0x3837.
- If SM85=ON, the execution result is: D0=0x31, D1=0x32, D2=0x33, D3=0x34, D4=0x35, D5=0x36, D6=0x37, D7=0x38.

6.5.17 ITA: Instruction for convertinga 16-bit hex data to an ASCII code

LAD:  ITA *(S1)* *(D)* *(S2)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: ITA *(S1)* *(D) (S2)*	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

S1: Source hex data to be converted

D: Destination operand

S2: Number of ASCII codes (1≤***S2***≤256)

- Function description

Converting the hex data starting from ***S1*** element to S2 ASCII codes, and assigning the obtained result to the elements starting from ***D*** when the energy flow is valid. Storing two ASCII code data in the high/low byte of each ***D*** element when SM85=OFF. Storing one ASCII code data in low byte of each ***D*** element when SM85=ON.

- Note

1. If ***S1*** and ***D*** use Kn addressing, Kn=4.

2. If ***S2*** is not within the range of 1 to 256, the system reports an operand error, does not execute the conversion operation, and keeps the content of ***D***.

3. If ***S1*** is a constant, when ***S2***≥4, the system default is ***S2***=4 and the system does not report an operand error once the default is adopted.

- Application instance

M0
ITA 16#9876 D20 8]
14393

Source data: 0x9876.

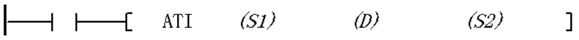
LD M0

ITA 16#9876 D20 8

Executing ITA conversion operation when M0=ONand data is stored in two modes:

- If SM85=OFF, the execution result is: D20=0x3839, D21=0x3637.
- If SM85=ON, the execution result is: D20=0x39, D21=0x38, D22=0x37, D23=0x36.

6.5.18 ATI: Instruction for converting an ASCII code to a 16-bit hex data

LAD:  ATI *(S1)* *(D)* *(S2)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: ATI *(S1)* *(D) (S2)*	**Step length**	**7**

Operand	Type	Applicable soft element	Indexing

S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source ASCII code data to be converted

  0x30≤***S1***≤0x39 or 0x41≤***S1***≤0x46 (when SM85=OFF, it is required thathigh/low bytes of ***S1*** be within this range)

  D: Destination operand

  S2: Number of ASCII codes (1≤***S2***≤256)

- Function description

  When energy flow is valid, convert the ASCII code data of the first ***S2*** of ***S1*** element to hex data, and store the result per 4 bits into the starting element of ***D***. When SM85=OFF, high/low byte of each ***D*** element stores two ASCII code data, when SM85=ON, low byte of each ***D*** element stores one ASCII code data.

  Converting S2 ASCII codes starting from ***S1*** element tothe hex data, and storing the obtained result every 4 bits in the elements starting from ***D*** when the energy flow is valid. Storing two ASCII code data in the high/low byte of each ***D*** element when SM85=OFF. Storing one ASCII code data in low byte of each ***D*** element when SM85=ON.

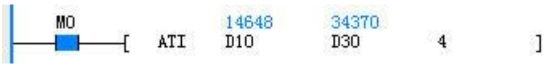
- Note

  1. If ***S1*** and ***D*** use Kn addressing, Kn=4.

  2. If ***S1*** is not within the range of 0x30 to 0x39 or 0x41 to 0x46, or ***S2*** is not within the range of 1 to 256, the system reports an operand error, does not execute the conversion operation, and keeps the content of ***D***.

  3. If ***S1*** is a constant, when SM85=OFF and ***S2***≥2, the system default is ***S2***=2. When SM85=ON and S2≥1, the system default is ***S2***=1. The system does not report an operand error once the default is adopted.

- Application instance



LD M0

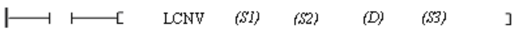
ATI D10 D30 4

Source data: D10=0x3938, D11=0x3736, D12=0x3534, D13=0x3332.

Executing ITA conversion operation when M0=ONand the results generated according to the data storage mode are listed below:

- If SM85=OFF, execution result is: D30=0x8967.
- If SM85=ON, execution result is: D30=0x8642.

6.5.19 LCNV: Project conversion instruction

LAD: 	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: LCNV *(S1) (S2) (D) (S3)*	**Step length**	**9**

Operand	Type	Applicable soft element													Indexing
S1	INT								D				V	R	
S2	INT								D				V	R	
D	INT								D				V	R	
S3	INT	Constant							D				V	R	

- Operand description

  S1: The start address of the source operand to be converted

  S2:The start address of conversion table

  D: The start address that stores the conversion result

  S3: Number of data to be converted (1≤***S3***≤64)

- Function description

  When adopting the analog input module to read the external analog signal, you can use this instruction to convert the original analog read value into the corresponding project read value.

  When temperature or analog modulesare used for temperature or analog measurement applications, if the temperature or project read value measured by the PLC deviates from the result measured by the standard thermometer or relevant standard instruments, this instruction can be used to make linear modification to calibrate the actual measurement.

  Filling the conversion table with four parameters, namely low point measurement value V_{ML} , high point measurement value V_{MH} and corresponding low point standard value V_{SL} and high point standard value V_{SH} . When the linear conversion is executed, the target standard value is generated by applying below formulato source data, in which S_n is the original input data, and D_n is conversion result data.

$$A = (V_{SL} - V_{SH})/(V_{ML} - V_{MH})*10000$$

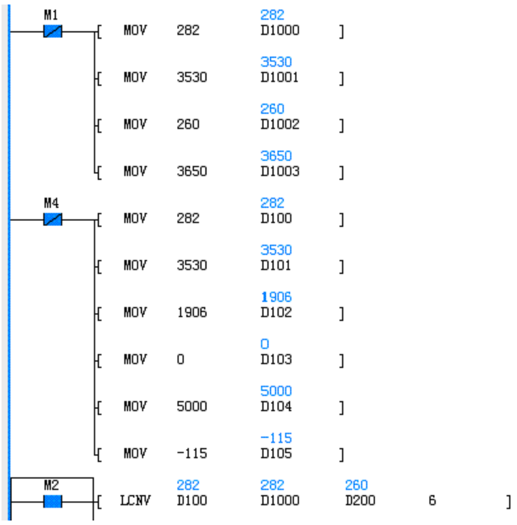
$$B = V_{SL} - (V_{ML} * A/10000)$$

$$D_n = (S_n * A/10000) + B$$

- Note

These four data in the conversion table has their actual meanings,for instance, the low point measurement value should be less than the high point measurement value. The result is inaccurate if the conversion result exceeds the integer range. If D_n is larger than 32767, it is 32767; if it is smaller than -32768, it is -32768.

- Application instance


M1
MOV 282 D1000 282
MOV 3530 D1001 3530
MOV 260 D1002 260
MOV 3650 D1003 3650
M4
MOV 282 D100 282
MOV 3530 D101 3530
MOV 1906 D102 1906
MOV 0 D103 0
MOV 5000 D104 5000
MOV -115 D105 -115
M2
LCNV D100 D1000 D200 6
282 282 260

```
LDI   M1
MOV    282   D1000
MOV    3530   D1001
MOV    260    D1002
MOV    3650   D1003
LDI     M4
MOV    282    D100
MOV    3530   D101
MOV    1906   D102
MOV    0        D103
MOV    5000   D104
MOV    -115    D105
LD       M2
LCNV   D100   D100   D1000   D200   6
```

Executing the LGNV instruction when M2=ON. The results generated according to data storage mode are listed below:

D200 = 260

D201 = 3650

D202 = 1955

D203 = -34

D204 = 5184

D205 = -154

6.5.20 RLCNV: Floating-point project conversion instruction

LAD: RLCNV (S1) (S2) (D) (S3)]	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: RLCNV (S1) (S2) (D) (S3)	**Step length**	**12**

Operand	Type	Applicable soft element													Indexing
S1	REAL								D				V	R	
S2	REAL								D				V	R	
D	REAL								D				V	R	
S3	INT	Constant							D				V	R	

- Operand description

S1: The start address of the source operand to be converted

S2:The start address of conversion table

D: The start address that stores the conversion result

S3: Number ofdata to be converted (1≤***S3***≤64)

- Function description

When adopting the analog input module to read the external analog signal, you can use this instruction to convert the original analog read value into the corresponding project read value.

When temperature or analog modulesare used for temperature or analog measurement applications, if the temperature or project read value measured by the PLC deviates from the result measured by the standard thermometer or relevant standard instruments, this instruction can be used to make linear modification to calibrate the actual measurement.

Filling the conversion table with four parameters, namely low point measurement value V_{ML} , high point measurement value V_{MH} and corresponding low point standard value V_{SL} and high point standard value V_{SH} . When the linear conversion is executed, the target standard value is generated by applying below formulato source data, in which S_n is the original input data, and D_n is conversion result data.

$$A = (V_{SL} - V_{SH})/(V_{ML} - V_{MH})*10000$$

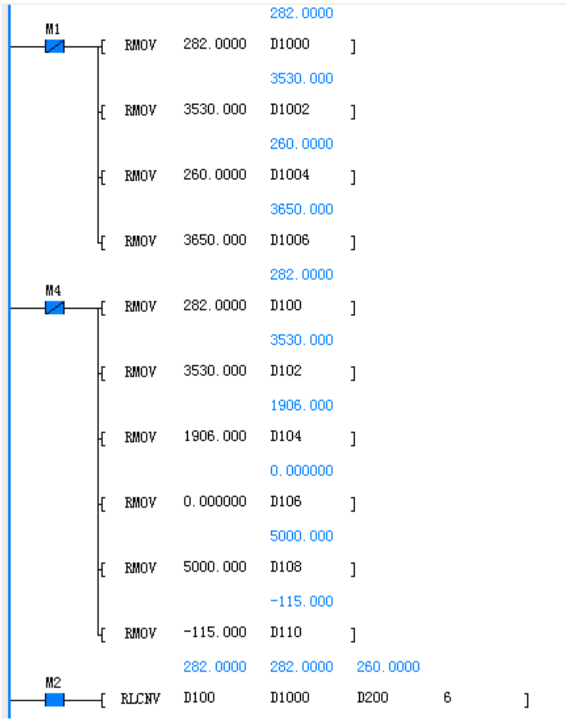
$$B = V_{SL} - (V_{ML} * A/10000)$$

$$D_n = (S_n * A/10000) + B$$

- Note

These four data in the conversion table has their actual meanings,for instance, the low point measurement value should be less than the high point measurement value. The result is inaccurate if the conversion result exceeds the integer range. If D_n is larger than 32767, it is 32767; if it is smaller than -32768, it is -32768.

- Application instance


M1
RMOV 282.0000 D1000 282.0000
RMOV 3530.000 D1002 3530.000
RMOV 260.0000 D1004 260.0000
RMOV 3650.000 D1006 3650.000
M4
RMOV 282.0000 D100 282.0000
RMOV 3530.000 D102 3530.000
RMOV 1906.000 D104 1906.000
RMOV 0.000000 D106 0.000000
RMOV 5000.000 D108 5000.000
RMOV -115.000 D110 -115.000
M2
RLCNV D100 D1000 D200 6
282.0000 282.0000 260.0000

```
LDI    M1
RMOV    282    D1000
RMOV    3530   D1002
RMOV    260    D1004
RMOV    3650   D1006
LDI     M4
RMOV    282    D100
RMOV    3530   D102
RMOV    1906   D104
RMOV    0      D106
RMOV    5000   D108
RMOV    -115   D110
LD      M2
RLCNV   D100    D1000   D200   6
```

Executing the RLCNV instruction when M2=ON. The results generated according to data storage mode are listed below:

D200(D201) = 260

D202(D203) = 3650

D204(D205) = 1955

D206(D207) = -34.3288

D208(D209) = 5184.267

D210(D211) = -154.357

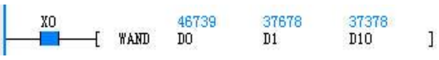
6.6 Word logic operation instructions

6.6.1 WAND: Word AND instruction

LAD: |—| |—[WAND (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	

IL: WAND *(S1)* *(S2)* *(D)*									**Step length**		**7**					
Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description
  S1: Source operand 1
  S2: Source operand 2
  D: Destination operand
- Function description
  Performingthe AND operationby bit on ***S1*** and ***S2***, and assigningthe obtained result to **D** when the energy flow is valid.
- Application instance



LD X0
WAND
D0 D1 D10

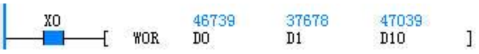
Performing the AND operationon D0=2#1011011010010011 (46739) and D1=2#1001001100101110 (37678) by bit, and assigning the obtained result to D10 when X0=ON. In this case, D10=2#1001001000000010 (37378).

6.6.2 WOR: INT OR instruction

LAD: WOR (S1) (S2) (D)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: WOR *(S1) (S2) (D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description
  S1: Source operand 1
  S2: Source operand 2
  D: Destination operand
- Function description
  Performingthe OR operationby bit on ***S1*** and ***S2***, and assigning the obtained result to ***D*** when the energy flow is valid.
- Application instance



LD X0
WOR D0 D1 D10

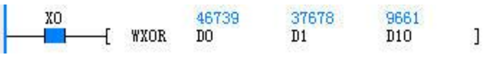
Performing the OR operation by bit on D0=2#1011011010010011 (46739) and D1=2#1001001100101110 (37678), and assigning the obtained result to D10 when X0=ON. In this case, D10=2#1011011110111111 (47039).

6.6.3 WXOR: Word XOR instruction

LAD: WXOR (S1) (S2) (D)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: WXOR *(S1) (S2) (D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description
  S1: Source operand 1
  S2: Source operand 2
  D: Destination operand
- Function description
  Performingthe XOR operationby bit on ***S1*** and ***S2***, and assigning the obtained result to ***D*** when the energy flow is valid.
- Application instance



LD X0
WXOR D0 D1 D10

Performing the XOR operation by bit on D0=2#1011011010010011 (46739) and D1=2#1001001100101110 (37678), and assigning the obtained result to D10 when X0=ON. In this case, D10=2#0010010110111101 (9661).

6.6.4 WINV: Word INV instruction

LAD: ├──┤ ├──[WINV (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: WINV *(S)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- Operand description

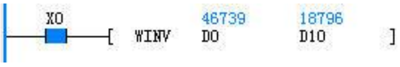
  S: Source operand

  D: Destination operand

- Function description

  Performingthe INV operationby bit on ***S***, and assigning the obtained result to ***D*** when the energy flow is valid.

- Application instance



LD X0
WINV D0 D10

Performing the INV operation by bit on D0= (46739), and assigning the obtained result to D10 when X0=ON. In this case, D10= (18796).

6.6.5 DWAND: Double word AND instruction

LAD: ├──┤ ├──[DWAND (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DWAND *(S1)* *(S1)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

  S1: Source operand 1

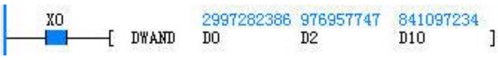
  S2: Source operand 2

  D: Destination operand

- Function description

  Performingthe AND operationby bit on ***S1*** and ***S2***, and assigning the obtained result to **D** when the energy flow is valid.

- Application instance



LD X0
DWAND D0 D2
D10

Performing the AND operationon (D0,D1)=2#10110010101001101110011001010010 (2997282386) and (D2, D3)=2#00111010001110110011000100110011 (976957747) by bit, and assigning the obtained result to (D10, D11) when X0=ON. In this case, (D10, D11)=2#00110010001000100010000000010010 (841097234).

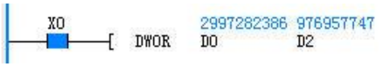
6.6.6 DWOR: Double word OR instruction

LAD: ├──┤ ├──[DWOR (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DWOR *(S1)* *(S2)* *(D)*	**Step length**	**10**

Operand	Type	Applicable soft element	Indexing

S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description
  S1: Source operand 1
  S2: Source operand 2
  D: Destination operand
- Function description
  Performingthe OR operationby bit on ***S1*** and ***S2***, and assigning the obtained result to ***D*** when the energy flow is valid.
- Application instance



LD X0
DWOR D0 D2 D10

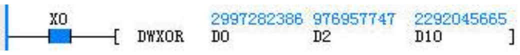
Performing the OR operation by bit on (D0, D1)=2#10110010101001101110011001010010 (2997282386) and (D2, D3)=2#00111010001110110011000100110011 (976957747), and assigning the obtained result to (D10, D11) when X0=ON. In this case, (D10, D11)=2#10111010101111111111011101110011 (3133142899).

6.6.7 DWXOR: Double word XOR instruction

LAD: ├─┤ ├─[DWXOR *(S1)* *(S2)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DWXOR *(S1)* *(S2)* *(D)*	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description
  S1: Source operand 1
  S2: Source operand 2
  D: Destination operand
- Function description
  Performingthe XOR operationby bit on ***S1*** and ***S2***, and assigning the obtained result to ***D*** when the energy flow is valid.
- Application instance



LD X0
DWXOR D0 D2
D10

Performing the XOR operation by bit on (D0,D1)=2#10110010101001101110011001010010(2997282386) and (D2,D3)=2#00111010001110110011000100110011 (976957747), and assigning the obtained result to (D10, D11) when X0=ON. In this case, (D10, D11)=2#10001000100111011101011101100001 (2292045665).

6.6.8 DWINV: Double word negation instruction

LAD: ├─┤ ├─[DWINV *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DWINV *(S)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element	Indexing

S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- Operand description

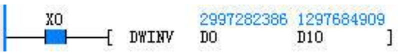
  S: Source operand

  D: Destination operand

- Function description

  Performing the INV operation by bit on ***S***, and assigning the obtained result to ***D*** when the energy flow is valid.

- Application instance


X0
2997282386 1297684909
DWINV
D0
D10

LD X0
DWINV D0 D10

Performing the INV operation by bit on (D0,D1)=2#10110010101001101110011001010010 (2997282386), and assigning the obtained result to (D10,D11) when X0=ON. In this case, (D10,D11)=2#01001101010110010001100110101101 (1297684909).

6.7 Bit shift and rotate instructions

6.7.1 ROR: 16-bit rotate right instruction

LAD: ├─┤ ├─[ROR (S1) (D) (S2)]		**Applicable model**	VC1S VC1 VC2 VC3 VC5
		Influenced flag bit	**Carry flag SM81**
IL: ROR *(S1)* *(D)* *(S2)*		**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  D: Destination operand

  S2: Source operand 2

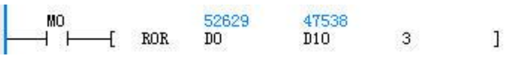
- Function description

  Cyclically shifting the data of ***S1*** rightward by ***S2*** bits, assigning the obtained result to ***D,*** and storing the destination bit in the carry flag bit (SM81) when the energy flow is valid.

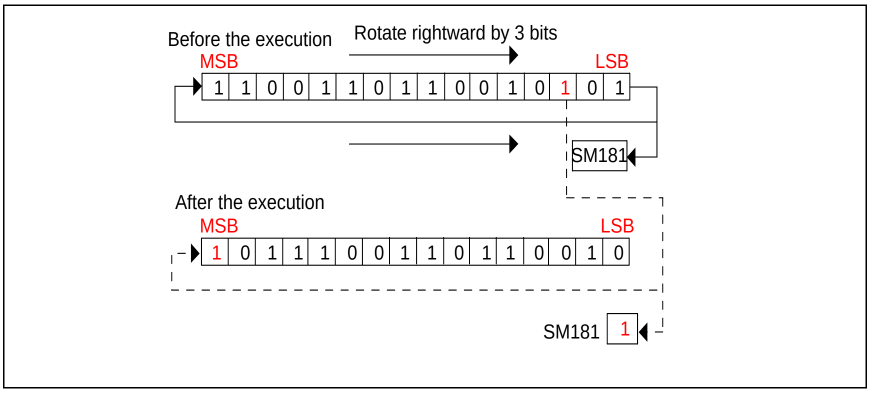
- Note

  S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 4.

- Application instance


M0
ROR
52629
D0
47538
D10
3

LD M0
ROR D0 D10 3


Before the execution
Rotate rightward by 3 bits
MSB
LSB
1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1
SM181
After the execution
MSB
LSB
1 0 1 1 1 0 0 1 1 0 1 1 0 0 1 0
SM181
1

Cyclically shifting D0=2#1100110110010101 (52629) rightward by 3 bits, assigning the obtained result to D10, and storing the destination bit in the carry flag bit when M0=ON. In this case, D10=2#1011100110110010 (47538) and SM81=ON.

6.7.2 ROL: 16-bit rotate left instruction

LAD: ├─┤ ├─[ROL (S1) (D) (S2)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Carry flag SM81**

IL: ROL (S1) (D) (S2)								**Step length**			**7**					
Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

S1: Source operand 1

D: Destination operand

S2: Source operand 2

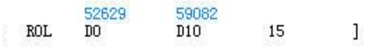
- Function description

Cyclically shifting the data of ***S1*** leftward by ***S2*** bits, assigning the obtained result to ***D,*** and storingthe destination bit in the carry flag bit (SM81)when the energy flow is valid.

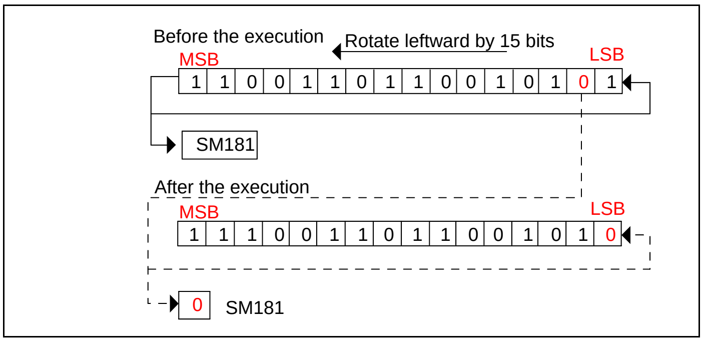
- Note

S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 4.

- Application instance



LD M0
ROL D0 D10 15



Cyclically shiftingD0=2#1100110110010101 (52629) leftward by 15 bits, assigning the obtained result to D10, and storing the destination bit in the carry flag bit when M0=ON. In this case, D10=2#1110011011001010 (59082) and SM81=OFF.

6.7.3 RCR: Instruction for 16-bit rotate right with carry flag bit

LAD: [RCR (S1) (D) (S2)]								**Applicable model**			VC1S VC1 VC2 VC3 VC5					
								Influenced flag bit			**Carry flag SM81**					
IL: RCR (S1) (D) (S2)								**Step length**			**7**					
Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

S1: Source operand 1

D: Destination operand

S2: Source operand 2

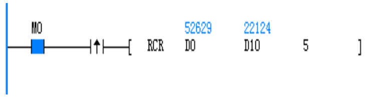
- Function description

Cyclically shifting the data of ***S1*** with the carry flag bit (SM81) rightward by ***S2*** bits, and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

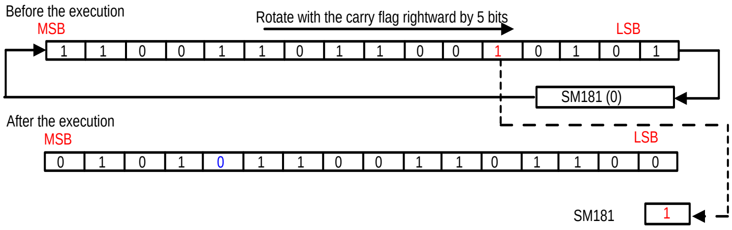
S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 4.

- Application instance


M0
52629
22124
RCR
D0
D10
5
]

LD M0
RCR D0 D10 5

Cyclically shifting D0=2#1100110110010101 (52629) with the carry flag bit (SM81=OFF) rightward by 5 bits, and assigning the obtained result to D10 when M0=ON.In this case, D10=2#0101011001101100 (22124) and SM81=ON.


Before the execution
Rotate with the carry flag rightward by 5 bits
MSB
LSB
1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1
SM181 (0)
After the execution
MSB
LSB
0 1 0 1 0 1 1 0 0 1 1 0 1 1 0 0
SM181
1

6.7.4 RCL: Instruction for 16-bit rotate left with carry flag bit

LAD: RCL (S1) (D) (S2)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Carry flag SM81**
IL: RCL (S1) (D) (S2)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  D: Destination operand

  S2: Source operand 2

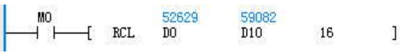
- Function description

  Cyclically shifting the data of ***S1*** with the carry flag bit (SM81) leftward by ***S2*** bits, and assigning the obtained result to ***D*** when the energy flow is valid.

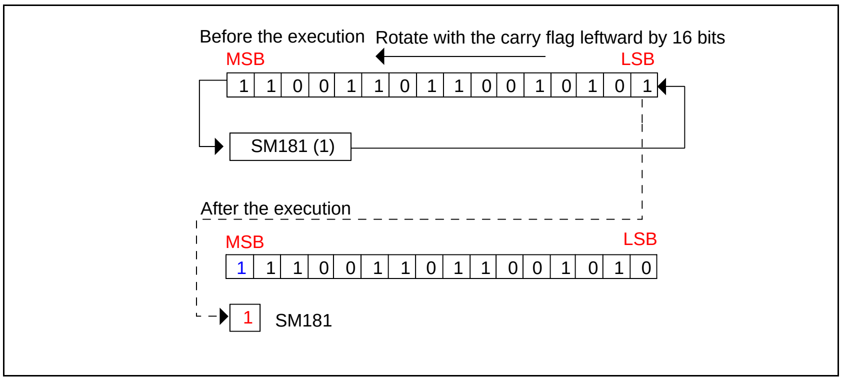
- Note

  S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 4.

- Application instance


M0
52629
59082
RCL
D0
D10
16
]

LD M0
RCL D0 D10 16


Before the execution
Rotate with the carry flag leftward by 16 bits
MSB
LSB
1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1
SM181 (1)
After the execution
MSB
LSB
1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0
1
SM181

Cyclically shifting D0=2#1100110110010101 (52629) with the carry flag bit (SM81=ON) rightward by16 bits, and assigning the obtained result to D10 when M0=ON. In this case, D10=2#1110011011001010 (59082) and SM81=ON.

6.7.5 DROR: 32-bit rotate right instruction

LAD: DROR (S1) (D) (S2)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Carry flag SM81**
IL: DROR (S1) (D) (S2)	**Step length**	**9**

Operand	Type	Applicable soft element													Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  D: Destination operand

  S2: Source operand 2

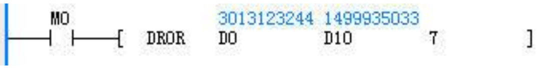
- Function description

  Cyclically shifting the data of ***S1*** rightward by ***S2*** bits, assigning the obtained result to ***D,*** and storingthe destination bit in the carry flag bit (SM81)when the energy flow is valid.

- Note

  S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 8.

- Application instance



LD M0
DROR D0 D10 7

Cyclically shifting D0 (D1)=2#10110011100110001001110010101100 (3013123244) rightward by 7 bits, assigning the obtained result to (D10, D11), and storing the destination bit in the carry flag bit when M0=ON. In this case, (D10,D11)=2#01011001011001110011000100111001 (1499935033) and SM81=OFF.

For details, refer to the diagram of ROR instruction.

6.7.6 DROL: 32-bit rotate left instruction

LAD: [DROL (S1) (D) (S2)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Carry flag SM81**
IL: DROL *(S1) (D) (S2)*	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  D: Destination operand

  S2: Source operand 2

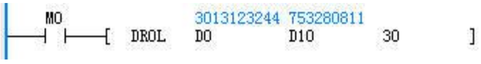
- Function description

  Cyclically shifting the data of ***S1*** leftward by ***S2*** bits, assigning the obtained result to ***D,*** and storingthe destination bit in the carry flag bit (SM81)when the energy flow is valid.

- Note

  S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 8.

- Application instance



LD M0
DROL D0 D10 30

Cyclically shifting (D0,D1)=2#10110011100110001001110010101100 (3013123244) rightward by 30 bits, assigning the obtained result to (D10, D11), and storing the destination bit in the carry flag bit when M0=ON. In this case, (D10,D11)=2#00101100111001100010011100101011 (753280811) and SM81=ON.

For detals, refer to the diagram of ROL instruction.

6.7.7 DRCR: Instruction for 32-bit rotate right with carry flag bit

LAD: [DRCR (S1) (D) (S2)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	**Carry flag SM81**
IL: DRCR *(S1) (D) (S2)*	**Step length**	**9**

Operand	Type	Applicable soft element	Indexing

S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  D: Destination operand

  S2: Source operand 2

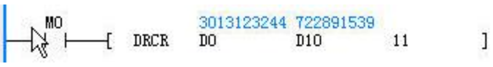
- Function description

  Cyclically shifting the data of ***S1*** with the carry flag bit (SM81) rightward by ***S2*** bits, and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 8.

- Application instance



LD M0
DRCR D0 D10 11

1. Cyclically shifting (D0,D1)=2#10110011100110001001110010101100 (3013123244)with the carry flag bit (SM81=OFF) rightward by 11 bits, and assigning the obtained result to (D10,D11) when M0=ON.In this case, (D10,D11)=2#00101011000101100111001100010011 (722891539) and SM81=ON.

2. For details, refer to the diagram of RCR instruction.

6.7.8 DRCL: Instruction for 32-bit rotate left with carry flag bit

LAD: DRCL (S1) (D) (S2)										**Applicable model**		VC1S VC1 VC2 VC3 VC5				
										Influenced flag bit		**Carry flag SM81**				
IL: DRCL *(S1) (D) (S2)*										**Step length**		**9**				
Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  D: Destination operand

  S2: Source operand 2

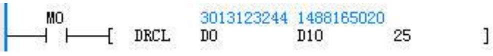
- Function description

  Cyclically shifting the data of ***S1*** with the carry flag bit (SM81) leftward by ***S2*** bits, and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 8.

- Application instance



LD M0
DRCL D0 D10 25

1. Cyclically shifting (D0,D1)=2#10110011100110001001110010101100 (3013123244)with the carry flag bit (SM81=ON) rightward by25 bits, and assigning the obtained result to (D10,D11) when M0=ON. In this case, (D10,D11)=2#00101100010110011100110001001100 (1488165020) and SM81=ON.

2. For details, refer to the diagram of RCL instruction.

6.7.9 SHR: 16-bit shift right instruction

LAD: SHR (S1) (D) (S2)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	

IL: SHR *(S1)* *(D)* *(S2)*										**Step length**		**7**				
Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  D: Destination operand

  S2: Source operand 2

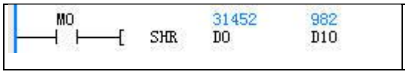
- Function description

  Shifting the data of ***S1*** rightward by ***S2*** bits, and assigning the obtained result to ***D*** when the energy flow is valid.

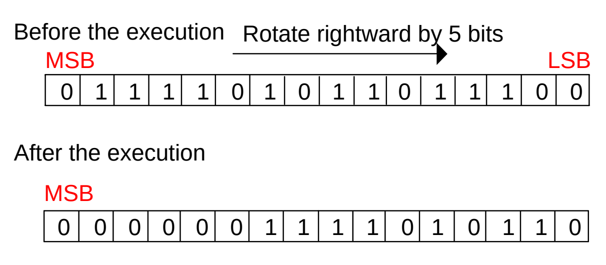
- Note

  S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 4.

- Application instance

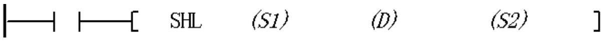

M0
SHR
31452
D0
982
D10

LD M0
SHR D0 D10 5


Before the execution
Rotate rightward by 5 bits
MSB
LSB
0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 0
After the execution
MSB
0 0 0 0 0 0 1 1 1 1 0 1 0 1 1 0

Shifting D0=2#0111101011011100 (31452) rightward by 5 bits, and assigning the obtained result to D10 when M0=ON. In this case, D10=2#0000001111010110 (982).

6.7.10 SHL:16-bit shift left instruction

LAD: 										**Applicable model**		VC1S VC1 VC2 VC3 VC5				
										Influenced flag bit						
IL: SHL *(S1)* *(D)* *(S2)*										**Step length**		**7**				
Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  D: Destination operand

  S2: Source operand 2

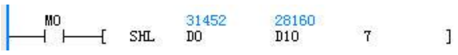
- Function description

  Shifting the data of ***S1*** leftward by ***S2*** bits, and assigning the obtained result to ***D*** when the energy flow is valid.

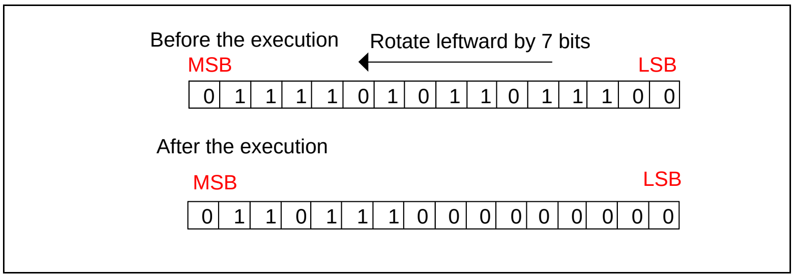
- Note

  S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 4.

- Application instance


M0
SHL
31452
D0
28160
D10
7

LD M0
SHL D0 D10 7


Before the execution
Rotate leftward by 7 bits
MSB
LSB
0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 0
After the execution
MSB
LSB
0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0

Shifting D0=2#0111101011011100 (31452) leftward by 7 bits, and assigning the obtained result to D10 when M0=ON. In this case, D10=2#0110111000000000 (28160).

6.7.11 DSHR: 32-bit shift right instruction

LAD: ├─┤ ├─[DSHR *(S1)* *(D)* *(S2)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DSHR *(S1)* *(D)* *(S2)*	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  D: Destination operand

  S2: Source operand 2

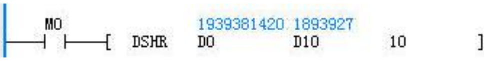
- Function description

  Shifting the data of ***S1*** rightward by ***S2*** bits, and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 8.

- Application instance



LD M0
DSHR D0 D10 10

1. Shifting (D0,D1)=2#01110011100110001001110010101100 (1939381420) rightward by 10 bits, and assigning the obtained result to (D10,D11) when M0=ON. In this case, (D10,D11)=2#00000000000111001110011000100111 (1893927).

2. For details, refer to the diagram of SHR instruction.

6.7.12 DSHL: 32-bit shift left instruction

LAD: ├─┤ ├─[DSHL *(S1)* *(D)* *(S2)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DSHL *(S1)* *(D)* *(S2)*	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Source operand 1

  D: Destination operand

  S2: Source operand 2

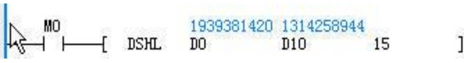
- Function description

  Shifting the data of ***S1*** leftward by ***S2*** bits, and assigning the obtained result to ***D*** when the energy flow is valid.

- Note

  S2 is greater than or equal to 0. When ***S1*** uses Kn addressing, Kn needs to be equal to 8.

- Application instance



LD M0
DSHL D0 D10 15

1. Shifting (D0,D1)=2#01110011100110001001110010101100 (1939381420) leftward by 15 bits, and assigning the obtained result to (D10,D11) when M0=ON. In this case, (D10,D11)=2#01001110010101100000000000000000 (1314258944).

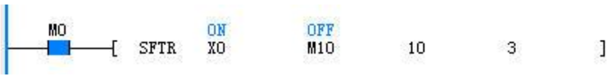
2. For details, refer to the diagram of SHR instruction.

6.7.13 SFTR: Bit string shift right instruction

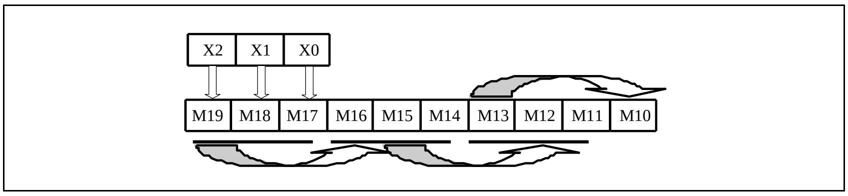
LAD: SFTR (S1) (D) (S2) (S3)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: SFTR *(S1) (D) (S2) (S3)*	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	BOOL		X	Y	M	S	LM	SM			C	T				√
D	BOOL			Y	M	S	LM				C	T				√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description
  S1: Source operand 1
  D: Destination operand
  S2: Source operand 2
  S3: Source operand 3
- Function description
  Shifting the content of the ***S2*** units starting from ***D*** rightward by ***S3*** units, discarding the ***S3*** data on the rightmost end, and shifting the content of ***S3*** units starting from ***S1*** to the left end of the word string when the energy flow is valid.
- Note
  Left right order, elements with large element No. areon the left, and elements with small element No. areon the right. Both ***S2*** and ***S3*** aregreater than or equal to zero.
- Application instance



LD M0
SFTR X0 M10 10 3



1. Shifting the content of 10 units starting from M10 rightward by 3 units, discarding M10 to M12 on the rightmost end, and shifting the content of 3 units starting from X0 to the left end of the bit string when M0=ON.

2. Before execution: X0=1, X1=0, X2=1. M10=0, M11=1, M12=1, M13=0, M14=0, M15=1, M16=0, M17=0, M18=0, and M19=1.

3. After execution: The contents of X0 to X2 are kept. M10=0, M11=0, M12=1, M13=0, M14=0, M15=0, M16=1, M17=1, M18=0, and M19=1.

6.7.14 SFTL: Bit string shift left instruction

LAD: SFTL (S1) (D) (S2) (S3)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: SFTL *(S1) (D) (S2) (S3)*	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	BOOL		X	Y	M	S	LM	SM			C	T				√
D	BOOL			Y	M	S	LM				C	T				√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description
  S1: Source operand 1
  D: Destination operand
  S2: Source operand 2

S3: Source operand 3

- Note

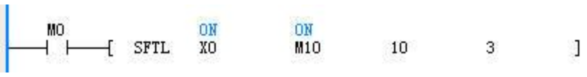
  1. Left right order, elements with large element No. areon the left, and elements with small element No. areon the right.

  2. Both ***S2*** and ***S3*** are greater than or equal to zero.

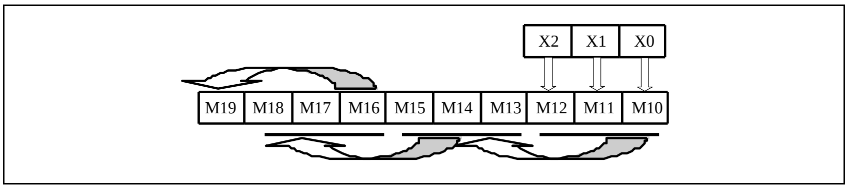
- Function description

  Shifting the content of the ***S2*** units starting from ***D*** rightward by ***S3*** units, discarding the ***S3*** data on the leftmost end, and shifting the content of ***S3*** units starting from ***S1*** to the right end of the word string when the energy flow is valid.

- Application instance



```
LD     M0
SFTL X0 M10 10 3
```



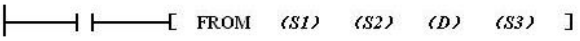
1. Shifting the content of 10 units starting from M10 leftward by 3 units, discarding M17 to M19 on the leftmost end, and shifting the content of 3 units starting from X0 to the right end of the bit string when M0=ON.

2. Before execution: X0=1, X1=0, X2=1。M10=0, M11=1, M12=1, M13=0, M14=0, M15=1, M16=0, M17=0, M18=0, and M19=1。

3. After execution: The contents of X0 to X2 are kept. M10=1, M11=0, M12=1, M13=0, M14=1, M15=1, M16=0, M17=0, M18=1, and M19=0.

6.8 Peripheral instructions

6.8.1 FROM: Instruction for reading words from a special module buffer register

LAD: 	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	
IL: FROM *(S1)* *(S2)* *(D)* *(S3)*	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant														
S2	INT	Constant														
D	INT								D				V		R	√
S3	INT	Constant														

- Operand description

  S1: SN of the special modules visited

  Settable range: 0 to 7. When accessing a special module that does not exist in the system, the system reports that the special module address is invalid.

  S2: The start address of the BFM buffer of the special module to be read

  Settable range: 0 to 32767. When accessing an invalid BFM address, the system reports that BFM unit of the assessed special module exceeds the range.

  D: The element where the data read from the special module is stored.

  S3: Number of the consecutive BFM buffers (single word) to be read

  Data range: 1 to 32767. When accessing an invalid address of the BFM buffer, the system reports that the BFM buffer of the assessed special module exceeds the range.

- Function description

  Reading consecutively ***S3*** word data starting from ***S2*** buffer in the designated BFM of designated special module (SN: ***S1***), and consecutively storing these data in ***S3*** word elements starting from ***D***.

- Note

  The execution time of the FROM instruction is relatively long, and related to the value of ***S3***.

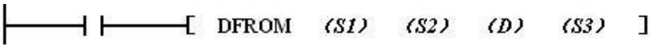
- Application instance

M0
FROM
0
3
997
D100
2

LD M0
FROM 0 3 D100 2

Reading consecutively 2 data starting from the buffer 3 in the BFM of No. 0 special module, and storing these data in D100 and D101 respectively when M0 is ON.

6.8.2 DFROM: Instruction for reading double words from a special module buffer register

LAD: 	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	
IL: DFROM *(S1)* *(S2)* *(D)* *(S3)*	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant														
S2	INT	Constant														
D	DINT								D				V		R	√
S3	INT	Constant														

- Operand description

  S1: SN of the special modules visited

  Settable range: 0 to 7. When accessing a special module that does not exist in the system, the system reports that the special module address is invalid.

  S2: The start address of the BFM buffer of the special module to be read

  Settable range: 0 to 32767. When accessing an invalid BFM address, the system reports that BFM unit of the assessed special module exceeds the range.

  D: The element where the data read from the special module is stored.

  S3: Number of the consecutive BFM buffers (double word) to be read

  Data range: 1 to 32767. When accessing an invalid address of the BFM buffer, the system reports that the BFM buffer of the assessed special module exceeds the range.

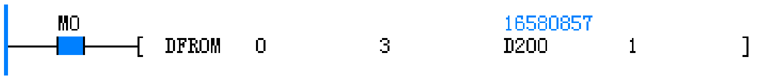
- Function description

  Reading consecutively ***S3*** word data starting from ***S2*** buffer in the designated BFM of designated special module (SN: ***S1***), and consecutively storing these data in ***S3*** double word elements starting from ***D***.

- Note

  The execution time of the DFROM instruction is relatively long, and related to the value of ***S3***.

- Application instance

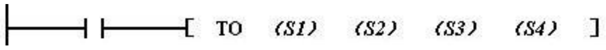


LD M0

DFROM 0 3 D200 1

Reading1 double data starting from the buffer 3 in the BFM of No. 0 special module, and storing these double data in D200 and D201 respectively when M0 is ON.

6.8.3 TO: Instruction for writing words from a special module buffer register

<table>
<tr><td colspan="16">LAD:
 </td><td colspan="2">Applicable model
Influenced flag bit</td><td colspan="2">VC2 VC3 VC5</td></tr>
<tr><td colspan="16">IL: TO <i>(S1)</i> <i>(S2)</i> <i>(S3)</i> <i>(S4)</i></td><td colspan="2">Step length</td><td colspan="2">9</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="17">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S1</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>S2</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>S3</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td></td><td>V</td><td></td><td>R</td><td></td><td></td><td></td><td>√</td></tr>
<tr><td>S4</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

- Operand description

  S1: SN of the special modules visited

  Settable range: 0 to 7. When accessing a special module that does not exist in the system, the system reports that the special module address is invalid.

  S2: The start address of the BFM buffer of the special module to be written

  Settable range: 0 to 32767. When accessing an invalid BFM address, the system reports that BFM unit of the assessed special module exceeds the range.

  S3: Data to be written into the special modules

  S4: Number of the consecutive BFM buffers (single word) to be written.

  Data range: 1 to 32767. When accessing an invalid address of the BFM buffer, the system reports that the BFM buffer of the assessed special module exceeds the range.

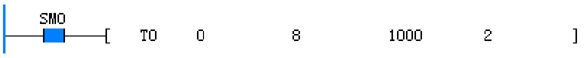
- Function description

  Writing the data from consecutive ***S4*** word units starting from ***S3*** into ***S4*** word elements starting fromthe designated BFM buffer (address is ***S2***)of the designated special module (SN: ***S1***).

- Note

  The execution time of the TO instruction is relatively long, and related to the value of ***S4***.

- Application instance



LD SM0

TO 0 8 1000 2

Writing 1000 respectively into BFM #8 and BFM #9 buffers of No. 0 special module when the PLC is running.

6.8.4 DTO: Instruction for writing double words from a special module buffer register

LAD: ——| |——[DTO (S1) (S2) (S3) (S4)]	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	
IL: DTO (S1) (S2) (S3) (S4)	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant														
S2	INT	Constant														
S3	DINT	Constant							D				V		R	√
S4	INT	Constant														

- Operand description

S1: SN of the special modules visited

Settable range: 0 to 7. When accessing a special module that does not exist in the system, the system reports that the special module address is invalid.

S2: The start address of the BFM buffer of the special module to be written

Settable range: 0 to 32767. When accessing an invalid BFM address, the system reports that BFM unit of the assessed special module exceeds the range.

S3: Data to be written into the special modules

S4: Number of the consecutive BFM buffers (double word) to be written.

Data range: 1 to 32767. When accessing an invalid address of the BFM buffer, the system reports that the BFM buffer of the assessed special module exceeds the range.

- Function description

Writingthe data from consecutive***S4*** double word units starting from ***S3*** into ***S4*** double word elements starting fromthe designated BFM buffer (address is ***S2***)of the designated special module (SN: ***S1***).

- Note

The execution time of the DTO instruction is relatively long, and related to the value of ***S4***.

- Application instance

SM0
DTO
0
8
16711935
1

LD SM0
DTO 0 8 16711935 1

Writing one double word data (16711935)into BFM #8 and BFM #9 buffers(they can constitute one double word) of No. 0 special module when the PLC is running.

6.8.5 VRRD: Instruction for readingthe value of an analog potentiometer

LAD: ——| |——[VRRD (S) (D)	**Applicable model**	**Reserve**
	Influenced flag bit	
IL: VRRD (S) (D)	**Step length**	**5**

Operand	Type	Applicable soft element													Indexing
S	WORD	Constant													
D	WORD								D				V		√

- Operand description

S: Designated analog potentiometer No.

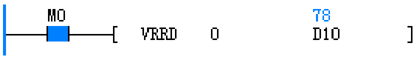
Settable range:: 0 to 255. If **S** is set outside this range, the system reports an operand error.

D: The element where the read analog potentiometer value is stored. Range:0 – 255.

- Function description

Reading the value of the designated analog potentiometer, and storing it into the designated element.

- Application instance



LD M0

VRRD 0 D10

Reading the value of No. 0 analog potentiometer in the system, and putting the read value into D10 when M0 is ON.

6.8.6 REFF: Instruction for setting input filtering constant

LAD: ├─┤ ├──[REFF (S)]	**Applicable model**	VC1S VC1
	Influenced flag bit	
IL: REFF *(S)*	**Step length**	**3**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

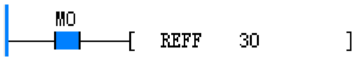
- Operand description

  S: Input filtering constant

  - VC1: The actual valid data is 0, 8, 16, 32 and 64. Any parameter less than 8 is processed as 0 and less than 16 is processed as 8. The parameter less than 32 is processed as 16 and less than 64 is processed as 32, and other data are processed as 64.
- Function description

  Setting the input filtering constant of X0－X7.
- Note

  The input filtering constant is valid only for the port used as normal input, and it is invalid for the port used as high-speed input.
- Application instance



LD M0
REFF 30

Setting the input filtering constant time to 30ms when X10 is ON.

6.8.7 REF: Instruction for immediately refreshing I/O

LAD: ├─┤ ├──[REF (D) (S)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: REF *(D)* *(S)*	**Step length**	**5**

Operand	Type	Applicable soft element													Indexing
D	BOOL		X	Y											
S	INT	Constant													

- Operand description

  D: The start X or Y element to be refreshed

  Designating the start soft element No. as an integral multiple of 8. For example, X0, X10, X20…or Y0, Y10, Y20…. The min. bit is 0.

  S: Number of I/O ports to be refreshed

  It should always be a multiple of 8, for example, 8, 16, ……, 256, and so on. Other values(except a multiple of 8) are wrong.
- Function description

  Generally, the PLC does not refresh its I/O points before the user program ends. However, if it is required to read the latest input state or immediately refresh the output state during the operation process, you can use the REF instruction.
- Note

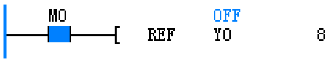
  1. The subscript number of input port (Xn, Yn) should be an integer multiple of 8.

  2. Number of the refreshed (port) should also be an integer multiple of 8.

  3. Generally, the REF instruction is used to refresh I/O immediately between the FOR-NEXT instruction and the CJ instruction.

  4. You can also use the REF instruction to obtain the latest input information and output the operation result without delay during the execution of the interrupt subprogram with I/O.

  5. To refresh arelay output, you need to consider the response time.
- Application instance



LD M0
REF Y0 8

Immediately outputting the statesat Y0 to Y7 regardless of the scan cycle when M0 is ON.

6.8.8 EROMWR: EEPROM write instruction

LAD: ├──┤ ├──[EROMWR *(S1)* *(S2)*]	**Applicable model**	**VC1**
	Influenced flag bit	
IL: EROMWR *(S1)* *(S2)*	**Step length**	**6**

Operand	Type	Applicable soft element														Indexing
S1	INT								D						R	
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: The start address of the elements being written(Range:D6000﹣D6999)

  S2: Number of the elements being written (S2＜16, and S1＋S2＜D7000)

- Note

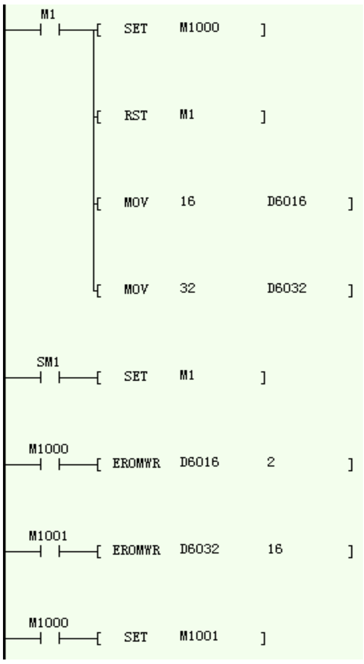
  One EEPROM instruction operation makes the scan cycle 2-5 ms longer. It is recommended to set ***S1*** to 6000 plus an integer multiple of 16, such as D6000, D6016 and D6032.

- Function description

  1. Generally, partial PLC data can be stored throughusing the function for saving data at power outage. However, during the calculation, you can save the intermediate data into EEPROM with the ERPROM instruction.

  2. This instruction is executed upon the rising edge.

- Application instance



```
LD        M1
SET       M1000
RST       M1
MOV       16 D6016
MOV       32 D6032
LD        SM1
SET       M1
LD        M1000
EROMWR    D6016 2
LD        M1001
EROMWR    D6032 16
LD        M1000
SET       M1001
```

In the preceding example, two sets of D elements are stored in the EEPROM. SM1 and M1 makes M1000 generate a rising edge in the second scan cycle, and triggers the execution of the first EROMWR instruction. M1001 makes the second rising edge, and triggers the execution of the second EROMWR instruction.

6.8.9 PR: Printing instruction

LAD: ├──┤ ├──[PR *(S)* *(D)*]	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	
IL: PR *(S)* *(D)*	**Step length**	**5**

Operand	Type	Applicable soft element												Indexing
S	INT								D		C	T		R	√
D	BOOL			Y											

- Operand description

  S: The start number of the soft elements that save data.

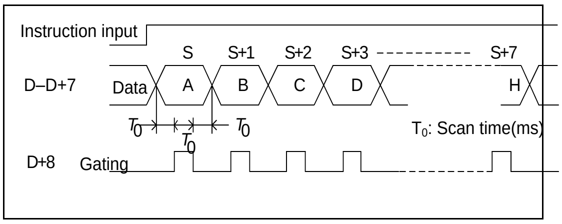
  D: The start number of Y elements that output data.

- Function description

1. Outputting the data saved in LSB of S–S+7 (1 byte) to D–D+7 in mode of hours and minutes, and fixing the enabling signal to Y0.

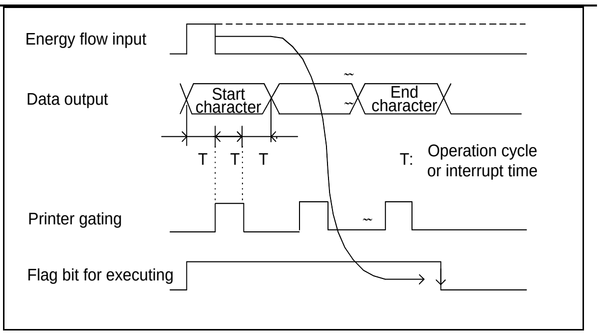
2. SM79 is a flag that indicates the printing instruction is being executed. SM79 is switched on during the printing process, and reset after the printing is completed.

The sequence diagram is shown below.


Instruction input
S
S+1
S+2
S+3
S+7
D–D+7
Data
A
B
C
D
H
T_0
T_0
T_0
T_0: Scan time(ms)
D+8
Gating

3. When the special register SM78 is OFF, the serial output is fixed at 8 bytes; when it is on, the serial output may be 1 to 16 characters; the code H00 (NUL code) indicates that the previous character is the last character of a string.

When SM78 is ON, the sequence diagram is as follows.


Energy flow input
Data output
Start character
End character
T
T
T
T:
Operation cycle or interrupt time
Printer gating
Flag bit for executing

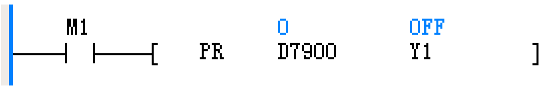
Resetting SM79 when the energy flow is invalid.

Note

1. Printing only once when the energy flow is valid.

2. SM79 (which indicates that the printing instruction is being executed) can be used to control the ON/OFF of the printing instruction.

- Application instance


M1
PR
0
D7900
OFF
Y1

- Note

1. This instruction is applicable only to transistor output modules.

2. This instruction is executed with the scan cycle.

3. Only one instruction can be executed at a time. When the printing is done, SM79 is reset.

6.8.10 TKY: Numeric key input instruction

LAD:	Applicable model	VC2 VC3 VC5
	Influenced flag bit	
IL: TKY (S) (D1) (D2)	Step length	7

Operand	Type	Applicable soft element																			Indexing
S	BOOL	X	Y	M	S	SM	LM														
D1	INT							KnY	KnM	KnLM	D	SD	C	T	V	Z		R			√
D2	BOOL		Y	M	S	SM	LM														

- Operand description

S: Start bit of the data entered through numeric key(occupying 10 bits).

D1: Data storage unit.

D2: Number of the soft element corresponding to the ON/OFF state of the input key (occupying 11 points).

- Function description

1. S to S+9 are used for keypad input, and the input data is stored in D1; D2 to D2+9 output the information entered through keypad; D2+10 detects keypad input, that is, when any one input is ON, this bit is set.

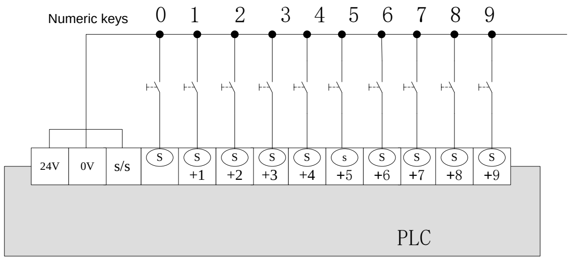
1) Inputting the value of D1

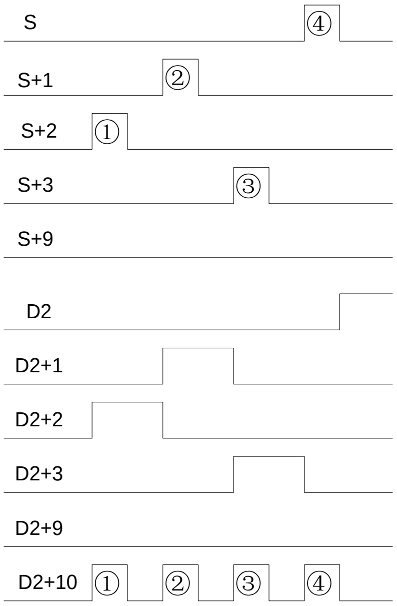
In the figure below, the value 2130 is stored in D1 after you press the numeric keys in the order of 1, 2, 3, and 4.

2) Key information of D2-D2+10

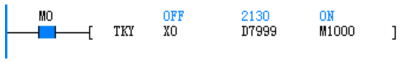
Whether the key information of D2-D2+9 is ON or OFF depends on the key you press.

When any one of the keys 0-9 is pressed, the keypad detection output of D2+10 is ON.


Numeric keys
0 1 2 3 4 5 6 7 8 9
24V
0V
s/s
S
+1
+2
+3
+4
+5
+6
+7
+8
+9
PLC


S
④
S+1
②
S+2
①
S+3
③
S+9
D2
D2+1
D2+2
D2+3
D2+9
D2+10
①
②
③
④

- Application instance


M0
TKY
OFF
X0
2130
D7999
ON
M1000

LD M0
TKY X0 D7999
M1000

When you press X2, X1, X3, and X0 in sequence, the content of D7999 becomes 2130. After X2 is pressed, M1002 is set to ON until another key is pressed,and so is the case when other keys are pressed. After any one of the keys is pressed, the keypad detection output M1010 is ON only during the pressing time.

- Note

1. When multiple keys are pressed simultaneously, only the information input by the key pressed first is valid.

2. When the energy flow is OFF, the content of D1 does not change, but D2 to D2+10 are set to OFF.

3. If the inputted data exceeds 9999, the MSB overflows first.

4. After an input key is pressed, the keypad detection output D2 corresponding to the key is set to ON until another input key.

5. Only one TKY instruction can be used in the program, and but you can use it multiple times through indexing.

6.9 Real-time clock instructions

6.9.1 TRD: Real-time clock read instruction

<table>
<tr><td colspan="9" rowspan="2">LAD:
├──┤ ├──[TRD (D)]</td><td colspan="3">Applicable model</td><td colspan="5">VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="5"></td></tr>
<tr><td colspan="9">IL: TRD <i>(D)</i></td><td colspan="3">Step length</td><td colspan="5">3</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>D</td><td>INT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
</table>

- Operand description

  D: Reading the start unit that storesthe system time, and storing the read time in 7 consecutive units starting from the unit assigned by ***D***.

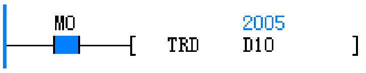
- Function description

  Reading the system time and storing it in the storage unit assigned by ***D***.

- Note

  The TRD instruction fails if there are something wrong in the system clock setting.

- Application instance



```
LD    M0
TRD   D10
```

Sending the system time to 7 units starting from D10 respectively when M0 is ON.

The execution results of the instruction are listed below:

	Element	Project	Clock data		Element	Project
Special data register for the real-time clock	SD60	Year	2000–2099	————————→	D10	Year
	SD61	Month	1–12	————————→	D11	Month
	SD62	Day	1–31	————————→	D12	Day
	SD63	Hour	0–23	————————→	D13	Hour
	SD64	Minute	0–59	————————→	D14	Minute
	SD65	Second	0–59	————————→	D15	Second
	SD66	Week	0–6	————————→	D16	Week

6.9.2 TWR: Real-time clock write instruction

LAD: ─┤ ├──[TWR (S)]	**Applicable model**	VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: TWR *(S)*	**Step length**	**3**

Operand	Type	Applicable soft element														Indexing
S	INT								D				V		R	√

- Operand description

S: The soft element where the system time is to be written

	Element	Project	Clock data		Element	Project
Data for clock setting	D10	Year	2000–2099	————→	SD60	Year
	D11	Month	1–12	————→	SD61	Month
	D12	Day	1–31	————→	SD62	Day
	D13	Hour	0–23	————→	SD63	Hour
	D14	Minute	0–59	————→	SD64	Minute
	D15	Second	0–59	————→	SD65	Second
	D16	Week	0–6	————→	SD66	Week

- Function description

You can use the TWR instruction to correct the system time when the system time is different from the actual time.

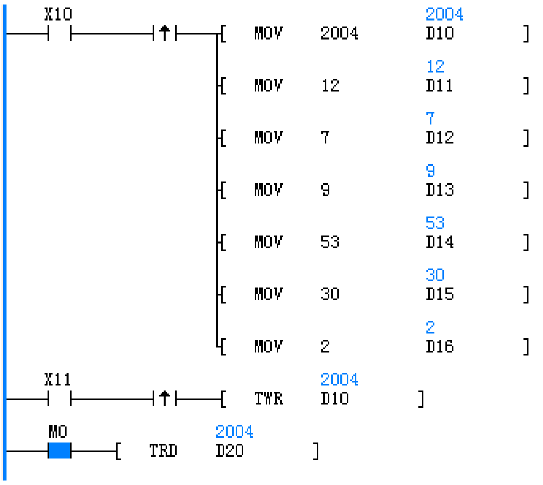
- Note

1. The written time data needs touse the solar calendar, otherwise the instruction cannot be executed successfully.

2. It is suggested touse the edge trigger as the execution condition of this instruction.

- Application instance

Changing the system time with the TWR instruction, as shown in the following figure.



```
LD    X10
EU
MOV   2004  D10
MOV   12    D11
MOV    7    D12
MOV    9    D13
MOV   53    D14
MOV   30    D15
MOV   2     D16
LD    X11
EU
TWR   D10
LD    M0
TRD    D20
```

1. Writing the time set values into 7 consecutive units starting from D10 upon detecting the rising edge of X10.

2. Writing the values of 7 consecutive units starting from D10 into the system time upon detecting the rising edge of X11.

3. Reading the system time and storing it in D20 when M0 is ON.

6.9.3 TADD: Clock addition instruction

LAD: ├─┤ ├──[TADD (S1) (S2) (D)]	**Applicable model**	VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag SM80, carry flag SM81**
IL: TADD *(S1) (S2) (D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT								D	SD			V		R	√
S2	INT								D	SD			V		R	√
D	INT								D				V		R	√

- Operand description

  S1: Clock data 1

  Storing the time data in the three storage units assigned by ***S1***. If the data is not compliant with the time format, the system reports an error, indicating that the value of the instruction operand is invalid.

  S2: Clock data 2

  Storing another time data in the three storage units assigned by ***S2***. If the data is not compliant with the time format, the system reports an error, indicating that the value of the instruction operand is invalid.

  D: Storage unit of time result

  Data obtained after the time addition operation is stored in the three storage units assigned by ***D***. The result affects the carry flag SM81 and the zero flag SM80.

- Function description

  Performing the addition operation on the data with the time format, and executing the operation rule in the time format.

- Note

  The time data in the operation need to conform to the time format:

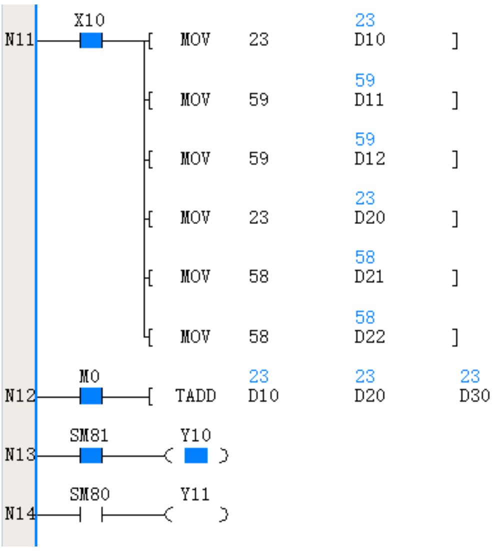
  Setting range of"Hour": 0﹣23.

  Setting range of"Minute": 0﹣59

  Setting range of"Second": 0﹣59

- Application instance

S1			S2			D	
D10	23 hours	＋	D20	23 hours	＝	D30	23 hours
D11	59 minutes		D21	58 minutes		D31	58 minutes
D12	59 seconds		D22	58 seconds		D32	57 seconds



LD X10
MOV 23 D10
MOV 59 D11
MOV 59 D12
MOV 23 D20
MOV 58 D21
MOV 58 D22
LD M0
TADD D0 D20 D30
LD SM81
OUT Y10
LD SM80
OUT Y11

1. Sending the time data to three storage units starting from D10 and three storage units starting from D20 when X10 is ON.
2. Increasing the three storage units starting from D10 by the three storage units starting from D20, and storing the obtained result in the three storage units starting from D30 when M0 is ON.
3. Setting the carry flag (SM81) to ON and the zero flag (SM80) to OFF.

6.9.4 TSUB: Clock subtraction instruction

LAD: |—| |—[TSUB (S1) (S2) (D)]	**Applicable model**	VC1 VC2 VC3 VC5
	Influenced flag bit	**Zero flag SM80, carry flag SM82**
IL: TADD *(S1) (S2) (D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT								D	SD			V		R	√
S2	INT								D	SD			V		R	√
D	INT								D				V		R	√

- Operand description

S1: Clock data 1

Storing the time data in the three storage units assigned by ***S1***. If the data is not compliant with the time format, the system reports an error, indicating that the value of the instruction operand is invalid.

S2: Clock data 2

Storing another time data in the three storage units assigned by ***S2***. If the data is not compliant with the time format, the system reports an error, indicating that the value of the instruction operand is invalid.

D: Storage unit of time result

Data obtained after the time addition operation is stored in the three storage units assigned by ***D***. The result affects the carry flag SM81 and the zero flag SM80.

- Function description

Performing the subtraction operation on the data with the time format, and executing the operation rule in the time format.

- Note

The time data in the operation need toconform to the time format:

Setting range of "Hour": 0 – 23

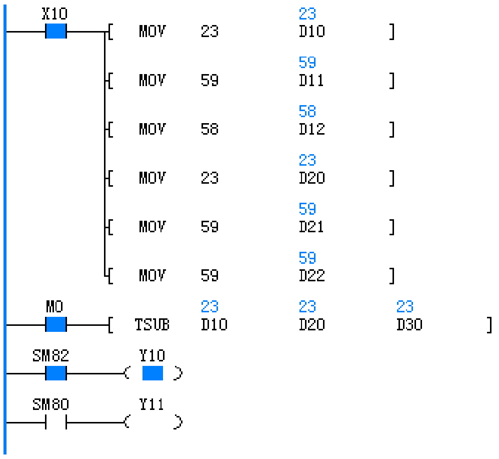
Setting range of "Minute": 0 – 59

Setting range of "Second": 0 – 59

- Application instance

S1			S2			D	
D10	23 hour	−	D20	23 hour	=	D30	23 hour
D11	59 minute		D21	59 minute		D31	59 minute
D12	58 second		D22	59 second		D32	59 second

- Application instance



```
LD   X10
MOV  23  D10
MOV  59  D11
MOV  58  D12
MOV  23  D20
MOV  59  D21
MOV  59  D22
LD   M0
TSUB  D10  D20
D30
LD   SM82
OUT  Y10
LD   SM80
OUT  Y11
```

1. Sending the time data to the three storage units starting from D10 and the three storage units starting from D20 when X10 is ON.

2. Subtracting the three storage units starting from D10 by the three storage units starting from D20, and storing the obtained result in the three storage units starting from D30 when M0 is ON.

3. Setting the carry flag (SM82) to ON and the zero flag (SM80) to OFF.

6.9.5 HOUR: Chronograph instruction

<table>
<tr><td colspan="10">LAD:
┤├──[HOUR (S) (D1) (D2)]</td><td colspan="3">Applicable model</td><td colspan="4">VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="10"></td><td colspan="3">Influenced flag bit</td><td colspan="4"></td></tr>
<tr><td colspan="10">IL: HOUR <i>(S)</i> <i>(D1)</i> <i>(D2)</i></td><td colspan="3">Step length</td><td colspan="4">8</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S</td><td>INT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
<tr><td>D1</td><td>INT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>D2</td><td>BOOL</td><td></td><td></td><td>Y</td><td>M</td><td>S</td><td>LM</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

- Operand description

  S: Hour comparison data. Data range: 0－32767

  D1: Time storage unit

  The data unit of ***D1*** stores hours while the data unit of ***D1***＋1 stores seconds.

  D2: Alarm output address

  When the data of ***D1*** is greater than or equal to the data designated by ***S***, the alarm point is changed to ON output.

- Function description

  Making the judgment on the time when the input contact is in the ON state (unit: hour).

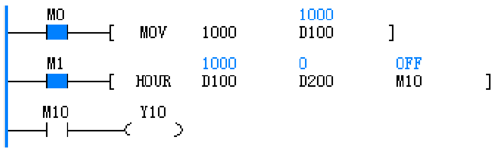
- Note

  1. Designate ***D1*** as the soft element unit for power-off storage to ensure present data can be used after PLC power is cut off. If normal soft element unit is used, current data will be cleared when PLC power is cut off or RUN → STOP operation is being carried out.

  2. Even if the alarm output ***D2*** is ON, it can continue counting.

  3. This instruction hour is 16-bit integer data. When the hour data is larger than 32767, starts from 0 again.

- Application instance



```
LD    M0
MOV   1000  D100
LD    M1
HOUR  D100  D200  M10
LD    M10
OUT   Y10
```

1. Setting the comparison data of the HOUR instruction when M0 is ON.

2. HOUR performs the time accumulation operation on the input contacts when M1 is ON.

3. When the accumulation time, that is, time of keeping M1 in the ON state, is greater than or equal to 1000, M10 is in the ON state.

6.9.6 DCMP: (=、<、>、<>、>=、<=)Date comparison instruction

<table>
<tr><td rowspan="2">LAD:
DCMP= (S1) (S2) (D)
DCMP< (S1) (S2) (D)
DCMP> (S1) (S2) (D)
DCMP<> (S1) (S2) (D)
DCMP>= (S1) (S2) (D)
DCMP<= (S1) (S2) (D)</td><td>Applicable model</td><td>VC1 VC2 VC3 VC5</td></tr>
<tr><td>Influenced flag bit</td><td></td></tr>
<tr><td>IL:
DCMP＝ (S1) (S2) (D)
DCMP＜ (S1) (S2) (D)
DCMP＞ (S1) (S2) (D)
DCMP＜＞ (S1) (S2) (D)
DCMP＞＝ (S1) (S2) (D)
DCMP＜＝ (S1) (S2) (D)</td><td>Step length</td><td>7</td></tr>
</table>

Operand	Type	Applicable soft element														Indexing
S1	INT								D	SD			V		R	√
S2	INT								D	SD			V		R	√
D	BOOL			Y	M	S	LM				C	T				

- Operand description

  S1: Date comparison data 1, which occupies the start three word units designated by ***S1***. Data of these three units need to comply with the format of solar calendar, otherwise the system reports an operand error.

  S2: Date comparison data 2, which occupies the start three word units designated by ***S2***. Data of these three units need to comply with the format of solar calendar, otherwise the system reports an operand error.

  D: Comparison state output. When the data comply with the comparison condition, setting D to ON, otherwise, it is OFF.

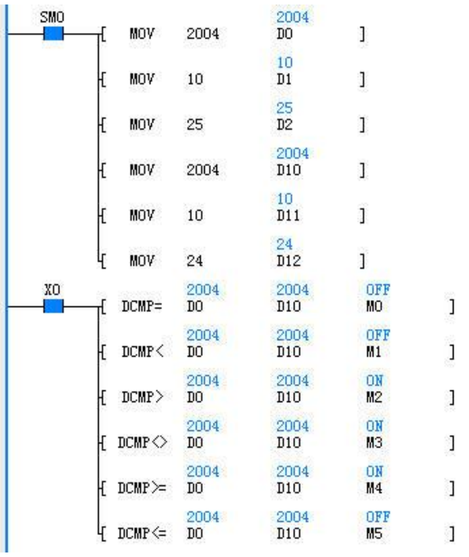
- Function description

  Conducting the BIN comparison on the date data starting from ***S1*** and ***S2*** respectively, and assigning the comparisonresult to ***D***.

- Note

  The date data starting from ***S1*** and ***S2*** need to comply with the format of the solar calendar, otherwise,the system reports an operand error(eg: 2004, 9, 31 and 2003, 2,29 are illegal).

- Application instance



```
LD      SM0
MOV 2004 D0
MOV 10   D1
MOV 25   D2
MOV 2004 D10
MOV 10   D11
MOV 24   D12
LD        X0
DCMP= D0 D10 M0
DCMP< D0 D10 M1
DCMP> D0 D10 M2
DCMP<> D0 D10 M3
DCMP>= D0 D10 M4
DCMP<= D0 D10 M5
```

Conducting the BIN comparison on the date data starting from D0 and D10 respectively, and assigning the comparison result to the destination data (M0, etc.).

6.9.7 TCMP: (=、<、>、<>、>=、<=)Time comparison instruction

LAD:	**Applicable model**	VC1 VC2 VC3 VC5
[TCMP= (S1) (S2) (D)] [TCMP< (S1) (S2) (D)] [TCMP> (S1) (S2) (D)] [TCMP<> (S1) (S2) (D)] [TCMP>= (S1) (S2) (D)] [TCMP<= (S1) (S2) (D)]	**Influenced flag bit**	
IL: **TCMP= (S1) (S2) (D)** **TCMP< (S1) (S2) (D)** **TCMP> (S1) (S2) (D)** **TCMP<> (S1) (S2) (D)** **TCMP>= (S1) (S2) (D)** **TCMP<= (S1) (S2) (D)**	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT								D	SD			V		R	√
S2	INT								D	SD			V		R	√
D	BOOL			Y	M	S	LM				C	T				

- Operand description

  S1: Time comparison data 1, which occupies the start three word units designated by ***S1***. Data of these three units need to comply with the 24-hour time format, otherwise the system reports an operand error.

  S2: Time comparison data 2, which occupies the start three word units designated by ***S2***. Data of these three units need to comply with the 24-hour time format, otherwise the system reports an operand error.

  D: Comparison state output. When the data comply with the comparison condition, setting D to ON, otherwise, it is OFF.

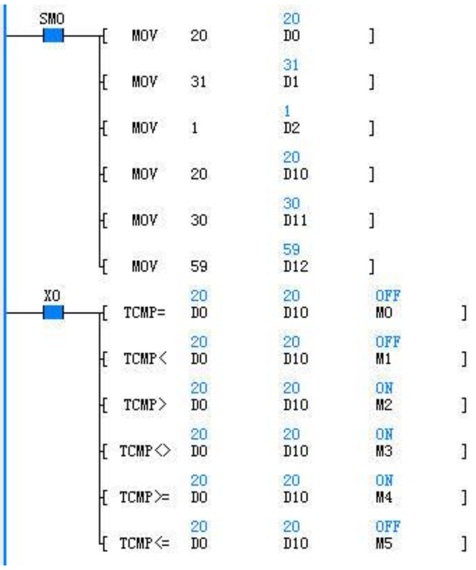
- Function description

  Conducting the BIN comparison on the time data starting from***S1*** and ***S2*** respectively, and assigning the comparisonresult to ***D***.

- Note

  The time data starting from***S1*** and ***S2*** need to comply with the 24-hour time format, otherwise,the system reports an operand error(eg: 24, 10, 31 and 13, 59, 60 are illegal).

- Application instance


SM0
MOV 20 D0 20
MOV 31 D1 31
MOV 1 D2 1
MOV 20 D10 20
MOV 30 D11 30
MOV 59 D12 59
X0
TCMP= D0 20 D10 20 M0 OFF
TCMP< D0 20 D10 20 M1 OFF
TCMP> D0 20 D10 20 M2 ON
TCMP<> D0 20 D10 20 M3 ON
TCMP>= D0 20 D10 20 M4 ON
TCMP<= D0 20 D10 20 M5 OFF

```
LD       SM0
MOV   20   D0
MOV   31   D1
MOV   1    D2
MOV   20   D10
MOV   30   D11
MOV   59   D12
LD        X0
TCMP=   D0   D10   M0
TCMP<   D0   D10   M1
TCMP>   D0   D10   M2
TCMP<> D0   D10   M3
TCMP>= D0   D10   M4
TCMP<= D0   D10   M5
```

Conducting the BIN comparison on the time data starting from D0 and D10 respectively, and assigning the comparison result to the destination data (M0, etc.).

6.9.8 HTOS: Instruction for converting hour-minute-second data to seconds

LAD:	**Applicable model**	VC2 VC3 VC5
─┤ ├──[HTOS (S) (D)]	**Influenced flag bit**	
IL: HTOS (S) (D)	**Step length**	**5**

Operand	Type	Applicable soft element													Indexing
S	INT		KnX	KnY	KnM	KnS	T	C	D	SD				R	√
D	INT			KnY	KnM	KnS	T	C	D	SD				R	√

- Operand description

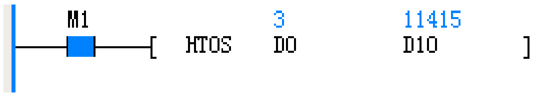
  S: Number of the start soft element that stores the time data to be converted.

  D: Number of the soft element that stores the converted time data.

- Function description

  Converting the time data (hours, minutes, and seconds) of S-S+2 into seconds, and storing the obtained result in D.

- Application instance



LD M1
HTOS D0 D10

1. Converting the time data (hours, minutes, and seconds) starting from D0 into seconds, and storing the obtained resultin D10 when M1=ON.D10=11415 when D0=3, D1=10 and D2=15.

6.9.9 STOH: Instruction for converting seconds to hour-minute-second data

LAD:	**Applicable model**	VC2 VC3 VC5
─┤ ├──[STOH (S) (D)]	**Influenced flag bit**	
IL: STOH (S) (D)	**Step length**	**5**

Operand	Type	Applicable soft element													Indexing
S	INT		KnX	KnY	KnM	KnS	T	C	D	SD				R	√
D	INT			KnY	KnM	KnS	T	C	D	SD				R	√

- Operand description

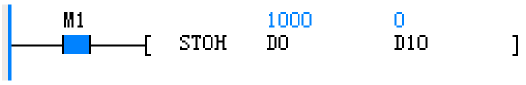
  S: Number of the soft element that stores the time data to be converted.

  D: Number of the start soft elementthat stores the converted time data.

- Function description

  Converting the second data of S into hours, minutes, and seconds, and storing the obtained result in D, D+1, and D+2 respectively.

- Note
- Application instance



LD M1
STOH D0 D10

1. Converting the second data of D0 into the time data (hours, minutes, and seconds), and storing the obtained result in three units starting from D10 when M1=ON.D10=0, D11=16, and D12=40 when D0=1000.

6.10 High-speed I/O instructions

6.10.1 HCNT: Instruction for driving the high-speed counter

LAD: ├──┤ ├──[HCNT *(D)* *(S)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: HCNT *(D)* *(S)*	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
D	DINT										C					
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√

- Operand description

  D: Assigning the counter number. Settable range: C236—C263

  S: Assigning the comparison constant, a 32-bit data with symbol. The data range: -2147483648 — +2147483647.

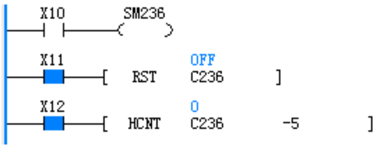
- Function description

  Driving the designated hardware high-speed counter. Performing the hardware high-speed counting on all high-speed counters only when it is continuously driven, and judging the action of NO contacts of the high-speed counter according to ***S***.

- Note

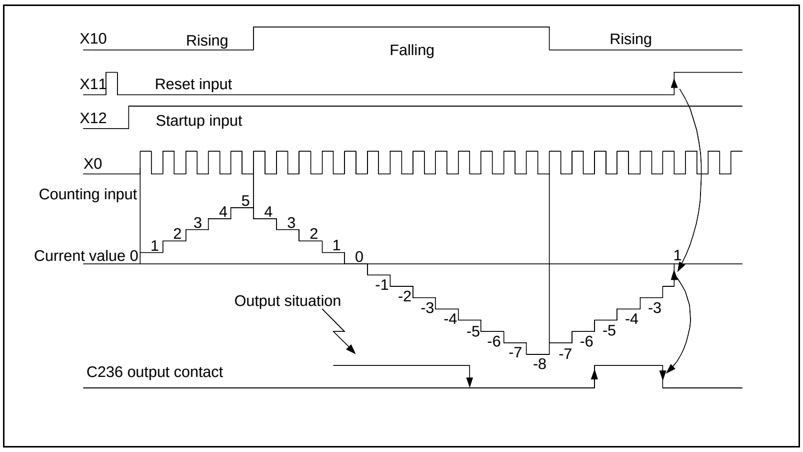
  There are hardware conflicts between the HCNT instruction, SPD instruction, external input interrupt, and pulse capture. You need to note that the use conditions of all high-speed I/O. Refer to Chapter 8"Operating guide for high-speed input function" for use.

- Application instance



```
LD     X10
OUT    SM236
LD     X11
RST    C236
LD     X12
HCNT   C236   -5
```

The time sequence operation of the program operation is shown below:



1. Initializing the hardware counter corresponding to C236 when X12 changes from OFF to ON. X0 is the pulse input end of C236 while C236 counts the external pulse of X0. X0 is a general input point when X12 is OFF, and C236 cannot count the external pulse of X0.

2. Action on the contacts: Setting the contact of C236 when the current value of the counter C236 is increased from -6 to -5. Resetting the contact of C236 when the current value of counter C236 is decreased from -5 to -6.

3. Executing the RST instruction when X11 is ON, clearing the C236 data, and disconnecting the C236 contact.

4. When a power failure occurs, you can set the data and contact state of the high-speed counter in the system block of the upper software.

6.10.2 DHSCS: Instruction of setting the high-speed count comparison

LAD: ├──┤ ├──[DHSCS *(S1)* *(S2)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DHSCS *(S1)* *(S2)* *(D)*	**Step length**	**10**

Operand	Type	Applicable soft element	Indexing

S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT										C					
D	BOOL			Y	M	S										

- Operand description

  S1: Data to be compared by the high-speed counter, 32-bit DINT data, ranging from -2147483648 to +2147483647.

  S2: High-speed counter. The applicable range: C236 – C263

  D: Bit element output object, immediately setting the output of Y, M, and S regardless of the scan period.

- Note

  1. The DHSCS instruction needs to work together with the HCNT instruction. DHSCS can be executed correctly only when the high-speed counter is driven by HCNT.

  2. The result compared by the DHSCS instruction takes effect when external pulses of the high-speed counter are inputted. Therefore, the DHSCS instruction does not generate any action even if the high-speed counter value is changed through the DMOV or MOV instruction.

  3. Like general instructions, multiple DHSCS (DHSCI, DHSCR, DHSZ, DHSP, and DHST) instructions can be used simultaneously, but the total number of these instructions cannot exceed 8. The first 8 instructions are executed in order, and the 9th or later instructions are not effective and therefore not executed.

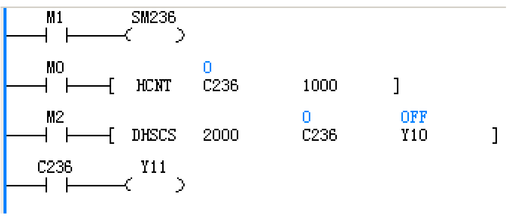
  4. The max. frequency supported by the high-speed counter of the PLCs. If you use the DHSCS, DHSCI, DHSCR, DHSZ, DHSP, or the DHST instruction, it is limited by the max. response frequency and comprehensive frequency. For details, refer to Chapter 8"Operating guide for high-speed input function".

- Function description

  1. A high-speed counter counts in interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON. When the value of the high-speed counter is equal to ***S1*** in the DHSCS instruction, the bit element designated by ***D*** is set immediately, or in the case of a Y element, the Y element is outputted immediately.

  2. This instruction can be used to immediately output the comparison result based on the comparison setting for the current value of the high-speed counter.

- Application instance



```
LD      M1
OUT     SM236
LD      M0
HCNT C236 1000
LD      M2
DHSCS 2000 C236 Y10
LD      C236
OUT     Y11
```

1. When M1 is ON, C236 counts in the interrupt mode when X0 changes from OFF to ON (for the input frequency of X0, refer to the instruction for high-speed I/O). When C236 changes from 999 to 1000, the C236 contact is set, and when C236 changes from 1001 to 1000, the C236 contact is reset. When Y11 is driven by C236, the execution of Y11 is determined by the scan cycle of the user program.

2. When M2 is ON and the DHSCS high-speed instruction meets the requirements stated in the preceding "Note", Y10 is outputted immediately if C236 reaches 2000, regardless of the scan cycle.

3. When M0 is ON, SM236 is driven, and the C236 counter counts down. When M0 is OFF, SM236 is not driven, and the C236 counter counts up.

6.10.3 DHSCI: Instruction for triggering interrupt based on comparison of high-speed count

LAD: ├──┤ ├──[DHSCI *(S1)* *(S2)* *(S3)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DHSCI *(S1)* *(S2)* *(S3)*	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT										C					
S3	INT	Constant														

- Operand description

S1: Data to be compared by the high-speed counter, 32-bit DINT data, ranging from -2147483648 to +2147483647.

S2: High-speed counter. The applicable range: C236–C263

S3: Interrupt number. Range of the interrupt number: 33-40

- Function description

A high-speed counter can count in interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON. When the value of the high-speed counter is equal to ***S1*** in the DHSCS instruction, it enters into the interrupt subprogram designated by ***S3***. You can write a program to be executed immediately in the interrupt subprogram.

- Note

1. The DHSCI instruction needs to worktogether with the HCNT instruction. DHSCI can be executed correctly only when the high-speed counter is driven by HCNT.

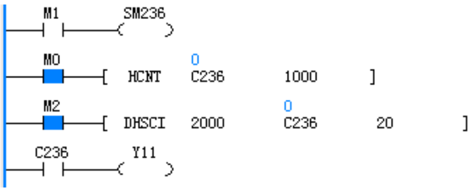
2. The result compared by the DHSCI instruction takes effect when external pulses of the high-speed counter are inputted. Therefore, the DHSCI instruction does not generate any action even if the high-speed counter value is changed through the DMOV or MOV instruction.

3. Like general instructions, multiple DHSCI (DHSCS, DHSCR, DHSZ, DHSP, and DHST) instructions can be used simultaneously, but the total number of these instructions cannot exceed 8.The first 8 instructions are executed in order, and the 9th or later instructions are not effective and therefore not executed.

4. The max. frequency supported by the high-speed counter of the PLCs. If you use the DHSCS, DHSCI, DHSCR, DHSZ, DHSP, or the DHST instruction, it is limited by the max. response frequency and comprehensive frequency. For details, refer to Chapter 8"Operating guide for high-speed input function".

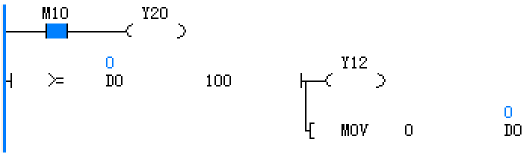
- Application instance

The User main program is shown as below:



```
LD      M1
OUT     SM236
LD      M0
DHSCI   2000   C236   20
LD      C236
OUT     Y11
```

The interrupt program whose user interrupt No. is 20 is shown as below:



```
LD      M10
OUT     Y20
LD>=    D0   100
OUT     Y12
MOV     0    D0
```

1. When M1 is ON, C236 counts in the interrupt mode when X0 changes from OFF to ON (for the input frequency of X0, refer to the instruction for high-speed I/O). When C236 changes from 999 to 1000, the C236 contact is set, and when C236 changes from 1001 to 1000, the C236 contact is reset. When Y11 is driven by C236, the execution of Y11 is determined by the scan cycle of the user program.

2. When M2 is ON and the DHSCI high-speed instruction meets the requirements stated in the preceding "Note", the interrupt subprogram whose interrupt No. is 20 responds immediately and executes the user program in the interrupt program when C236 reaches 2000.

3. When M1 is ON, SM236 is driven, and the C236 counter counts down. When M1 is OFF, SM236 is not driven, and the C236 counter counts up.

4. Entering into the No.20 interrupt program when C236 has the pulse input and C236 is 2000. Driving Y20 when M10 is ON. But the output of Y20 is related to the scan cycle of the user program. Meanwhile, driving Y12 and clearing the data of D0 when D0 is detected to be larger than 100.

6.10.4 DHSPI: Triggering interrupt Instruction based on comparison of absolute high-speed output positions

LAD: ├─┤ ├─[DHSPI (S1) (S2) (S3)]	**Applicable model**	VC2 VC3
	Influenced flag bit	
IL: DHSPI *(S1) (S2) (S3)*	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT									SD						
S3	INT	Constant														

- Operand description

S1: Data to be compared by the high-speed output position element, 32-bit DINT data, ranging from -2147483648 to +2147483647.

S2: High-speed output position element. The applicable range:

S3: Interrupt number. Range of the interrupt number: 53,54,55,56,57,58,59, and 60.

- Function description

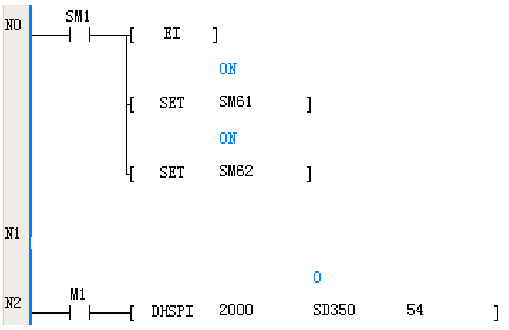
When the value of the high-speed output position element is equal to ***S1*** in the DHSPI instruction, it enters into the interrupt subprogram designated by ***S3***. You can write a program to be executed immediately in the interrupt subprogram.

- Note

1. When data is written in an SD element, no position-based interrupt is triggered. After the data is written, position-based interrupt is triggered by a specified position.

- Application instance

The User main program is shown as below:



The interrupt number of the interrupt subprogram can be set to 54 or other position-based interrupt source of high-speed output, and then the program to be executed when passing the position can be written in the interrupt subprogram.

6.10.5 DHSCR: Instruction for resetting the high-speed count comparison

LAD: |—| |—[DHSCR *(S1)* *(S2)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DHSCR *(S1)* *(S2)* *(D)*	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT										C					
D	BOOL			Y	M	S					C					

- Operand description

  S1: Data to be compared by the high-speed counter, 32-bit DINT data, ranging from -2147483648 to +2147483647.

  S2: High-speed counter. The applicable range: C236－C263.

  D: Bit element output object, immediately resetting the output of Y, M, S, and C regardless of the scan period. If D is a C element, it shall be ***S2***.

- Function description

  A high-speed counter can count in interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON. When the value of the high-speed counter is equal to ***S1*** in the DHSCR instruction, the bit element designated by ***D*** is reset immediately, or in the case of a Y element, the Y element is outputted immediately.2. This instruction can be used to immediately output the comparison result based on the comparison setting for the current value of the high-speed counter.

- Note

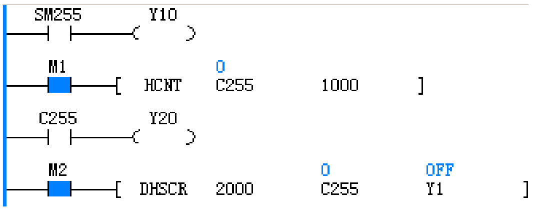
  1. The DHSCR instruction needs to work together with the HCNT instruction. DHSCR can be executed correctly only when the high-speed counter is driven by HCNT.

  2. The result compared by the DHSCR instruction takes effect when external pulses of the high-speed counter are inputted. Therefore, the DHSCR instruction does not generate any action even if the high-speed counter value is changed through the DMOV or MOV instruction.

  3. Like general instructions, multiple DHSCR (DHSCI, DHSCS, DHSZ, DHSP, and DHST) instructions can be used simultaneously, but the total number of these instructions cannot exceed 8.The first 8 instructions are executed in order, and the 9th or later instructions are not effective and therefore not executed.

  4. The max. frequency supported by the high-speed counter of the PLCs. If you use the DHSCS, DHSCI, DHSCR, DHSZ, DHSP, or DHST instruction, it is limited by the max. response frequency and comprehensive frequency. For details, refer toChapter 8"Operating guide for high-speed input function".

- Application instance



```
LD     SM255
OUT    Y10
LD     M1
HCNT   C255   1000
LD     C255
OUT    Y20
LD     M2
DHSCR  2000   C255   Y1
```

1. When M1 and X7 are both ON, C255 counts the phase difference of X3 and X4 in the interrupt mode (for the input frequency of X0, refer to the instruction for high-speed I/O). When C255 changes from 999 to 1000, the C255 contact is set, and when C255 changes from 1001 to 1000, the C255 contact is reset. When Y20 is driven by C255, the execution of Y20 is determined by the scan cycle of the user program.

2. When M2 is ON and the DHSCR high-speed instruction meets the requirement stated in the preceding "Note", Y1 is outputted immediately if C255 reaches 2000, regardless of the scan cycle.

3. When the input pulse of X3 is ahead of the one of X4, SM255 is ON; when the input pulse of X4 is ahead of the one of X3, SM255 is OFF.

4. When X7 (start signal of C255) is OFF, the C255 counter cannot count.

5. When M1 and X7 are both ON, if X5 is ON, the C255 counter is cleared and the C255 auxiliary contact is also cleared.

6.10.6 DHSZ: High-speed count range comparison instruction

LAD: DHSZ (S1) (S2) (S3) (D)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DHSZ *(S1)* *(S2)* *(S3)* *(D)*	**Step length**	**13**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S3	DINT										C					
D	BOOL			Y	M	S										

- Operand description

  S1: Data 1 to be compared by the high-speed counter, 32-bit DINT data, ranging from -2147483648 to +2147483647.

  S2: Data 2 to be compared by the high-speed counter,32-bit DINT data, ranging from 2147483648 to +2147483647.

  S3: High-speed counter. The applicable range: C236 – C263

  D: Bit element output object, immediately setting the output of Y, M, and S regardless of the scan period.

- Function description

  1. A high-speed counter can count in interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON.

  2. Setting the bit element designated by ***D***, and resetting the two bit elements following the one designated by ***D*** when the value of the high-speed counter is less than ***S1*** in the instruction.

  3. Resetting the bit element designated by ***D***, and setting the two bit elements following the one designated by ***D*** when the value of the high-speed counter is greater than or equal to ***S1*** and less than or equal to ***S2***.

  4. Resetting the bit element designated by ***D***, and resetting the two bit elements following the one designated by ***D*** when the value of high-speed counter is larger than ***S***2 in the DHSZ instruction.

  5. If it is Y element, Y element outputs the corresponding state immediately, and the output action is irrelevant to the scan cycle of the program.

- Note

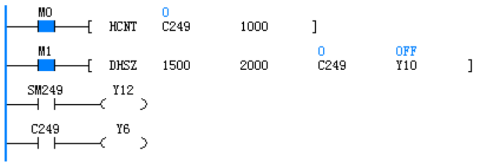
  1. The DHSZ instruction needs to worktogether with the HCNT instruction. DHSZ can be executed correctly only when the high-speed counter is driven by HCNT.

  2. The result compared by the DHSZ instruction takes effect when external pulses of the high-speed counter are inputted. Therefore, the DHSZ instruction does not generate any action even if the high-speed counter value is changed through the DMOV or MOV instruction.

  3. Like general instructions, multiple DHSCZ (DHSCI, DHSCS, DHSCR, DHSP, and DHST) instructions can be used simultaneously, but the total number of these instructions cannot exceed 8.The first 8 instructions are executed in order, and the 9th or later instructions are not effective and therefore not executed.

  4. The max. frequency supported by the high-speed counter of the PLCs. If you use the DHSCS, DHSCI, DHSCR, DHSZ, DHSP, or DHST instruction, it is limited by the max. response frequency and comprehensive frequency. For details, refer toChapter 8"Operating guide for high-speed input function".

- Application instance



```
LD      M0
HCNT    C249    1000
LD      M1
DHSZ    1500 2000 C249    Y10
LD      SM249
OUT     Y12
LD      C249
OUT     Y6
```

1. When M0 is ON, C249 counts up when X2 changes from OFF to ON, or counts down when X3 changes from OFF to ON (for the input frequency of X0, refer to the instruction for high-speed I/O). When C249 changes from 999 to 1000, the C249 contact is set and when C249 changes from 1001 to 1000, the C249 contact is reset. When Y6 is driven by C249, the execution of Y6 is determined by the scan cycle of the user program.

2. When M1 is ON and the DHSZ high-speed instruction meets the requirements stated in the preceding "Note", the state of Y10, Y11, and Y12 are as below:

(1) C249＜1500: Y10: ON; Y11, Y12: OFF。

(2) 2000≥C249≥1500: Y10, Y12: OFF; Y11: ON。

(3) C249＞2000: Y10, Y11: OFF; Y12: ON。

The outputs of Y10, Y11 and Y12 are not affected by the scan cycle.

3. When M0 is ON, if X2 counts up from OFF to ON,SM249 is reset. If X3 counts down from OFF to ON, SM249 is set.

6.10.7 DHST: High-speed count table comparison instruction

LAD: |—| |—[DHST *(S1)* *(S2)* *(S3)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DHST *(S1)* *(S2)* *(S3)*	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	DINT								D						R	
S2	INT	Constant														
S3	DINT										C					

- Operand description

  S1: Start unit of the data forthe table comparison (Start number of D element). The subsequent three D elements with consecutive serial numbers are used to specify the data to be compared by the high-speed counter, the serial number and corresponding output state of the Y element. The D elements connected to these four consecutive serial numbers are collectively referred to beone record.

  S2: Number of records to be compared. The data range: 1﹣128

  S3: High-speed counter. The applicable range: C236﹣C263.

- Function description

  1. A high-speed counter can count in interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON.

  2. When the value of the high-speed counter is equal to the data to be compared currently, the corresponding Y element state is output according to the recorded data, and the output object can be the Y element only.

  3. The output action is irrelevant to the scan cycle, and the Y element designated by the current record outputs the designated state immediately.

  4. If it is desired that the the user program executesthe immediate output operation according to the comparison data designated by a certain table and Y elements, you can use the DHST instruction.

- Note

  1. The DHST instruction needs to work together with the HCNT instruction. DHST can be executed correctly only when the high-speed counter is driven by HCNT.

  2. The result compared by the DHST instruction takes effect when external pulses of the high-speed counter are inputted. Therefore, the DHST instruction does not generate any action even if the high-speed counter value is changed through the DMOV or MOV instruction.

  3. Like general instructions, multiple DHST (DHSCI, DHSCS, DHSCR, DHSP, and DHSZ) instructions can be used simultaneously, but the total number of these instructions cannot exceed 8.The first 8 instructions are executed in order, and the 9th or later instructions are not effective and therefore not executed. If DHSP is a valid instruction in the user's instruction, DHST is not executed. On the contrary, if DHST is a valid instruction,DHSP is not executed. Only one instruction (DHST or DHSP) can be valid at the same time in the user program.

  4. The max. frequency supported by the high-speed counter of the PLCs. If you use the DHSCS, DHSCI, DHSCR, DHSZ, DHSP, or DHST instruction, it is limited by the max. response frequency and comprehensive frequency. For details, refer to Chapter 8"Operating guide for high-speed input function".

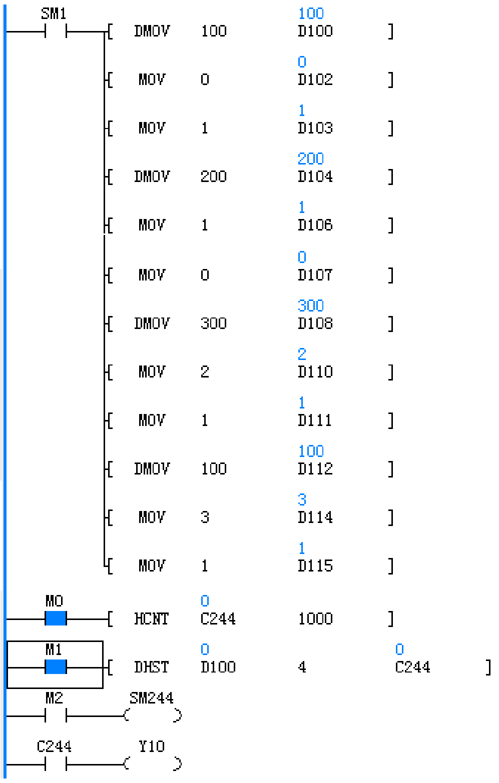
- Application instance

  The table data is shown as below:

Comparison data		Output Y number	Set/Reset	Operation process
MSB	LSB			

D100=0	D101=100	D102=0	D103=1	1 ↓
D104=0	D105=200	D106=1	D107=0	2 ↓
D108=0	D109=300	D110=2	D111=1	3 ↓
D112=0	D113=300	D114=3	D115=1	4 ↓ Return to 1

The LAD is shown as below:


SM1
DMOV 100 100 D100
MOV 0 0 D102
MOV 1 1 D103
DMOV 200 200 D104
MOV 1 1 D106
MOV 0 0 D107
DMOV 300 300 D108
MOV 2 2 D110
MOV 1 1 D111
DMOV 100 100 D112
MOV 3 3 D114
MOV 1 1 D115
M0
HCNT 0 C244 1000
M1
DHST 0 D100 4 0 C244
M2
SM244
C244
Y10

```
LD      SM1
DMOV    100   D100
MOV     0     D102
MOV     1     D103
DMOV    200   D104
MOV     1     D106
MOV     0     D107
DMOV    300   D108
MOV     2     D110
MOV     1     D111
DMOV    100 D112
MOV     3     D114
MOV     1     D115
LD      M0
HCNT    C244   1000
LD      M1
DHST    D100   4   C244
LD      M2
OUT     SM244
LD      C244
OUT     Y10
```

1. Assigning the initial value to D100﹣D115, and generating the table to be compared in the first scan cycle of the user program.

2. When M0 is ON, C244 counts when X0 changes from OFF to ON (for the input frequency of X0, refer to the instruction for high-speed I/O). When C244 changes from 999 to 1000, the C244 is set and when C244 changes from 1001 to 1000, the C244 is reset. When Y10 is driven by C244, the execution of Y10 is determined by the scan cycle of the user program.

3. When M1 is ON and the DHST high-speed instruction meets the requirements stated in the preceding "Note", the DHST high-speed instructionstarts from No.1 record of the table and enters the comparison of No.2 record only after No. 1 record is completed. The comparison of the next record can be entered onlyafter the previous instruction is completed. When the comparison of the last record is completed, it returns to the comparison of the first record again and sets SM83. SD90 indicates the record No. to be compared currently while SD88 and SD89 indicate the data to be compared currently. The comparison result is outputted immediately, regardless of the scan cycle.

4. When M2 is ON, SM244 is ON, and C244 counts down. If M2 is OFF, SM244 is OFF, and C244 counts up.

6.10.8 DHSP: Instruction for pulse output based on high-speed count table comparison

<table>
<tr><td colspan="9" rowspan="2">LAD:
├──┤ ├──[DHSP (S1) (S2) (S3)]</td><td colspan="3">Applicable model</td><td colspan="4">VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="4"></td></tr>
<tr><td colspan="9">IL: DHSP <i>(S1) (S2) (S3)</i></td><td colspan="3">Step length</td><td colspan="4">10</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S1</td><td>DINT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td></td><td></td><td>R</td><td></td></tr>
<tr><td>S2</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>S3</td><td>DINT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>C</td><td></td><td></td><td></td><td></td></tr>
</table>

- Operand description

S1: Start unit of the data forthe table comparison (Start number of D element). The subsequent three D elements with consecutive serial numbers are used to specify the data to be compared by the high-speed counter, and the data outputted to SD86 and SD87. The D elementsconnected to these four consecutive serial numbers are collectively referred to beone record.

S2: Number of records to be compared. The data range: 1–128.

S3: High-speed counter. The applicable range: C236–C263.

- Function description

1. A high-speed counter can count in interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON.

2. When the value of the high-speed counter is equal to the comparison data of the current record, SD86 and SD87 are modified according to output data of current data.

3. If it is needed to make user program determine high speed output or other data assignment value according to a certain table, use DHSP table comparison instruction. For instance, user can assign SD86 and SD87 (double word) as the output frequency operand of PLSY instruction to make PLSY output frequency adjust according to table comparison result.

3. You can use the DHSP instruction when you hope the user program to determine the assignment of high-speed output or other data based on a certain table. For example, SD86 and SD87 (double word) can be specified as the output frequency operands of the PLSY instruction, so as to adjust the PLSY output frequency according to the table comparison result.

- Note

1. The DHSP instruction needs to work together with the HCNT instruction. DHSP can be executed correctly only when the high-speed counter is driven by HCNT.

2. When DHSP is used in conjunction with PLSY, the data transmitted to SD86 and SD 181 need to meet the frequency output requirement of PLSY. For details, refer to the PLSY instruction.

3. If you want to stop the comparison at the last row, you need to set the data sent to SD86 and SD87 by the last cell of the comparison table to 0. In this case, the other DHST and DHSP instructions are invalid, but the DHSP instruction is not counted in the total instruction number of other high speed instructions.

4. The result compared by the DHSP instruction takes effect when external pulses of the high-speed counter are inputted. Therefore, the DHSP instruction does not generate any action even if the high-speed counter value is changed through the DMOV or MOV instruction.

5. Like general instructions, multiple DHSP (DHSCI, DHSCS, DHSCR, DHST, and DHSZ) instructions can be used simultaneously, but the total number of these instructions cannot exceed 8.The first 8 instructions are executed in order, and the 9th or later instructions are not effective and therefore not executed.

6. If DHSP is a valid instruction in the user instruction, DHST is not executed, on the contrary, if DHST is a valid instruction, DHSP is not executed. Only one instruction (DHST or DHSP) can be valid at the same time in the user program, and the others are invalid.

7. The max. frequency supported by the high-speed counter of the PLCs. If you use the DHSCS, DHSCI, DHSCR, DHSZ, DHSP, or DHST instruction, it is limited by the max. response frequency and comprehensive frequency. For details, refer to Chapter 8"Operating guide for high-speed input function".

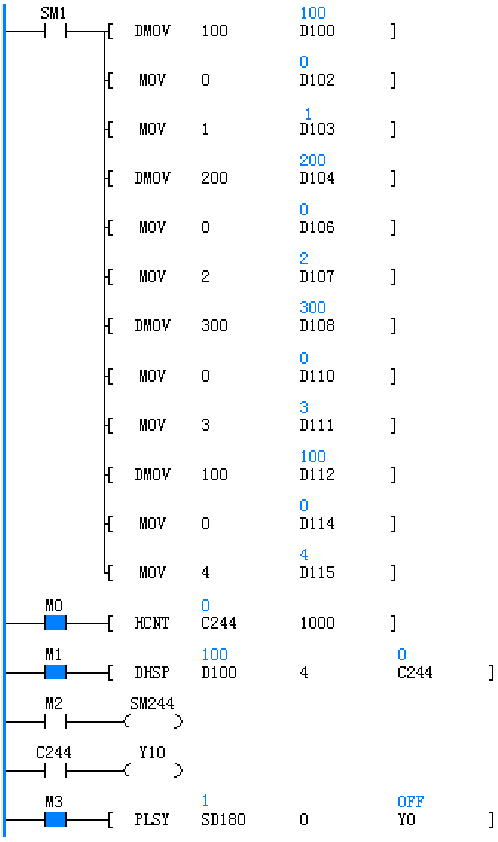
- Application instance

Table data is shown as below:

Comparison data	Data outputted to SD86 and SD87	Operation process

MSB	LSB	MSB	LSB	
D100=0	D101=100	D102=0	D103=1	1↓
D104=0	D105=200	D106=0	D107=2	2↓
D108=0	D109=300	D110=0	D111=3	3↓
D112=0	D113=100	D114=0	D115=4	4↓ Return from 1

The LAD is shown as below:


SM1
DMOV 100 100 D100
MOV 0 0 D102
MOV 1 1 D103
DMOV 200 200 D104
MOV 0 0 D106
MOV 2 2 D107
DMOV 300 300 D108
MOV 0 0 D110
MOV 3 3 D111
DMOV 100 100 D112
MOV 0 0 D114
MOV 4 4 D115
M0
HCNT 0 C244 1000
M1
DHSP 100 D100 4 0 C244
M2 SM244
C244 Y10
M3
PLSY 1 SD180 0 OFF Y0

```
LD      SM1
DMOV    100   D100
MOV      0    D102
MOV      1    D103
DMOV    200   D104
MOV      0     D106
MOV      2     D107
DMOV    300   D108
MOV      0     D110
MOV      3     D111
DMOV     100   D112
MOV      0     D114
MOV      4     D115
LD       M0
HCNT     C244   1000
LD       M1
DHSP      D100   4   C244
LD       M2
OUT       SM244
LD         C244
OUT         Y10
LD          M3
PLSY       SD86   0   Y0
```

1. Assigning the initial value to D100－D115, and generating the table to be compared in the first scan cycle of the user program.

2. When M0 is ON, C244 counts when X0 changes from OFF to ON (for the input frequency of X0, refer to the instruction for high-speed I/O). When C244 changes from 999 to 1000, the C244 is set and when C244 changes from 1001 to 1000, the C244 is reset. When Y10 is driven by C244, the execution of Y10 is determined by the scan cycle of the user program.

3. When M1 is ON and the DHSP high-speed instruction meets the requirements stated in the preceding "Note", the DHSP high-speed instructionstarts from the record No.1 of the table and enters the comparison of No.2 record only after No. 1 record is completed. The comparison of the next record can be entered onlyafter the previous instruction is completed. When the comparison of last record is completed, it returns to the comparison of first record again and sets SM83. SD90 indicates the record No. to be compared currently while SD88 and SD89 indicate the data to be compared currently. The comparison result is outputted immediately, regardless of the scan cycle. The output operands obtained after the comparison are put into SD86 and SD87, which is not affected by the scan cycle. If you want to stop at the last row in the comparison table, you need to set the data sent to SD86 and SD87 by the final table to 0.

4. When M2 is ON, SM244 is ON, and C244 counts down. If M2 is OFF, SM244 is OFF, and C244 counts up.

6.10.9 SPD: Frequency measuring instruction

LAD: ⊣⊢ [SPD (S1) (S2) (D)]	Applicable model	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: SPD (S1) (S2) (D)	Step length	7

Operand	Type	Applicable soft element														Indexing
S1	BOOL		X													
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT								D				V		R	√

- Operand description

  S1: Input point. Settable range: X0﹣X7

  S2: Time unit for input point detection. Unit: ms. Operand S2>0

  D: Pulse detection data storage unit. When the count value exceeds 65535, an automatic overflow is performed.

- Function description

  Detecting the number of input pulses of X0–X7 within the specified time (ms), and storing the obtained result in the designated soft element unit.

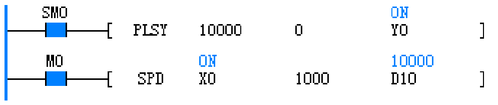
- Note

  1. There is a hardware conflict between SPD and HCNT, external input interrupt, and pulse capture. For details, refer to Chapter 8"Operating guide for high-speed input function" for use.

  2. For VC1, the input points of SPD are within the range of X0 to X7.

  3. The max. pulse input frequency of SPD is 10kHz, and an detection error may occur if the input frequency exceeds 10kHz.

- Application instance



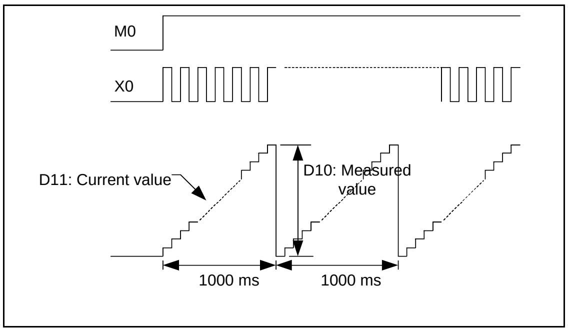
LD SM0

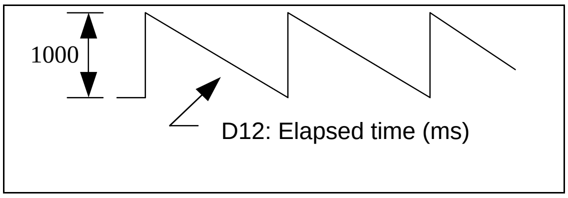
PLSY 10000 0 Y0

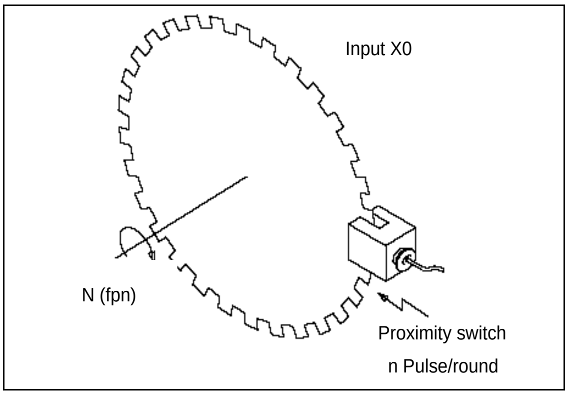
LD M0

SPD X0 1000 D10

The time sequence operation of program is shown as below:







1. Counting the input pulse designated by X0 within 1000ms, and storing the count result to the storage unit of D10 when M0 is ON, in which D11 is the current count value within 1000ms, and D12 is the elapsed time within 1000ms.

2. Data of D10 is proportional to the rotation speed in the above diagram.

3. Counting whenever X0 changes from OFF to ON, and storing the counting values in D10 at every 1000ms.

6.10.10 PLSY: High-speed pulse output instruction

LAD: ├──┤ ├──[PLSY (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3
	Influenced flag bit	
IL: PLSY (S1) (S2) (D)	**Step length**	**9**

Operand	Type	Applicable soft element													Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	BOOL			Y												

- Operand description

S1: Specified frequency (Hz)

When ***S1*** is not within the setting range, the system reports that the instruction operand is illegal and no system hardware resource is occupied. When the content of ***S1*** is changed while the instruction is running, the output frequency changes accordingly.

S2: Number of generated pulses (PLS).

If the set operand is not within the setting range, the system reports that the instruction operand is illegal, the pulse does not output, and no system hardware resource is occupied. When ***S2*** is 0, the pulse always outputs when the instruction is valid.

If the content of ***S2*** is changed while the instruction is running, the change takes effect in the next round of the drive.

D: High-speed pulse output point.

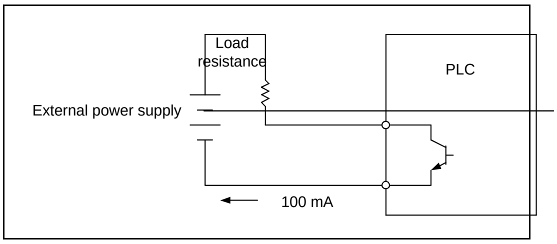
- Function description

Generating a specified amount of the high-speed pulse outputaccording to the frequency specified by the instruction. To output the high-speed pulses, the load current on the output transistor of the PLCs needs to be large enough, but it cannot exceed the rated load current.

- Note

1. The PLCs need to adopt the transistor output mode.2. When the PLCsare executing the high-speed pulse output, it is required to use the load current specified by the PLC output transistor described below.

3. The output circuit (transistor) for PLSY, PWM and PLSR is shown as follows.



4. With the large load current, the OFF time of the transistoris relatively long. When executing the PWM, PLSY and PLSR instructions, it is required the output port of the transist or be connected to the corresponding load. When the output waveform does not conform to the instruction operand, it is viable to increase the load current of the transistor (the transistor load≤100mA).

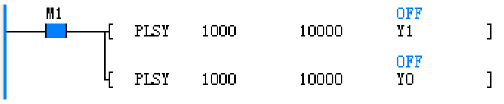
5. When the high-speed instruction is executed effectively (including output completion), other operations on the same portare invalid. Only when the high-speed pulse output instruction is invalid can other instructions operate this port.

6. Using multiple PLSR instructions can obtain separate high-speed pulse output at high-speed output points, or using the PWM (or PLSR) instruction to obtain separate high-speed pulse output at different output points.

7. When multiple PWM, PLSY or PLSR instructions operate on the same port, the first valid instruction controls the output state of the port and the later valid instructions do not affect the state of the output points.

8. Similar to other high-speed instructions (DHSCS, DHSCR,DHSZ, DHSP, DHST, and HCNT), the PLSY instruction needs to meet the requirements for high-speed input and high-speed pulse output in the system.

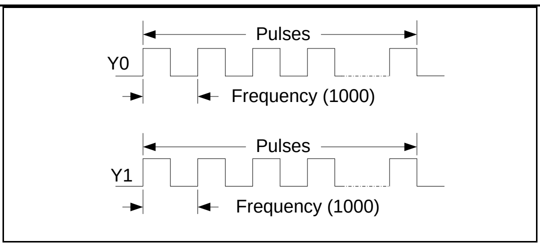
- Application instance



LD M1

PLSY 1000 10000 Y1

PLSY 1000 10000 Y0



1. When M1 is ON, 10000 pulses are output through Y0 and Y1 ports at the frequency of 1000 Hz. and pulse output is stopped after 10000 pulses are outputted. When M1 changes from OFF to ON, the next round of output starts. When M1 is OFF, the port output is OFF.

2. The duty cycle of the pulse is 50% ON, and 50% OFF. The output control is not affected by the scan cycle and is handled by the interrupts. At the high frequency output, the output duty cycle at the Y ports is related to the load. The waveforms obtained from the output terminals are related to the user's output load. When the load cannot exceed the rated load current, the smaller the load, the closer the output waveform is to the set operand.

3. SM270 is used to control the output of Y0 while SM290 is used to control the output of Y1, and when SM290 is 1, the output pulse of Y1 is disabled.

4. SM271 and SM291 indicate the output flags of Y0 and Y1 respectively. The flag is cleared when the output is completed or M0 is OFF.

5. SD160 indicates the MSB of the number of the Y0 output pulses in PLSY and PLSR instructions.

SD161 indicates the LSB of the number of the Y0 output pulses in PLSY and PLSR instructions.

SD180 indicates the MSB of the number of the Y1 output pulses in PLSY and PLSR instructions.

SD181 indicates the LSB of the number of the Y1 output pulses in PLSY and PLSR instructions.

6. SD160–SD180 can be modified through "DMOV ××× SD160"or"MOV ××× SD180", or through the monitoring.

7. If you want to use the number of input pulses to control the output pulse frequency of PLSY, refer to the DHSP instruction for details

6.10.11 PLSR: Instruction for count pulse output with acceleration/deceleration

LAD: |—| |—[PLSR (S1) (S2) (S3) (D)	**Applicable model**	VC1S VC1 VC2 VC3
	Influenced flag bit	
IL: PLSR *(S1)* *(S2)* *(S3)* *(D)*	**Step length**	**10**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D1	BOOL			Y												

- Operand description

S1: Max frequency (Hz). Settable range: 10–20000 (Hz).

If the operand specified indirectly is larger than 20000, it is treated as 20000, if it is less than 10, it is treated as 10. In that case, the system an error, indicating that the value of the instruction operand is invalid, and high-speed pulse is outputted by default.

S2: Total number of output pulses (PLS). Settable range: 110–2147483647. If the set operand is not withinthe range, the system reports an error, indicating that the value of the instruction operand is invalid, outputs no pulse, and no hardware resources corresponding to this instruction is occupied.

S3: Acceleration/deceleration time (ms)

If ***S1***×***S3***<100000, ***S3*** is regarded as 100000/***S1***, the system reports a parameter error of the PLSR instruction, and the acceleration/deceleration time is uncertain.

If ***S1***×***S3***>S2×909, S3 is regarded as S2×909/***S1***, the system reports a parameter error of the PLSR instruction, and the acceleration/deceleration time is uncertain.

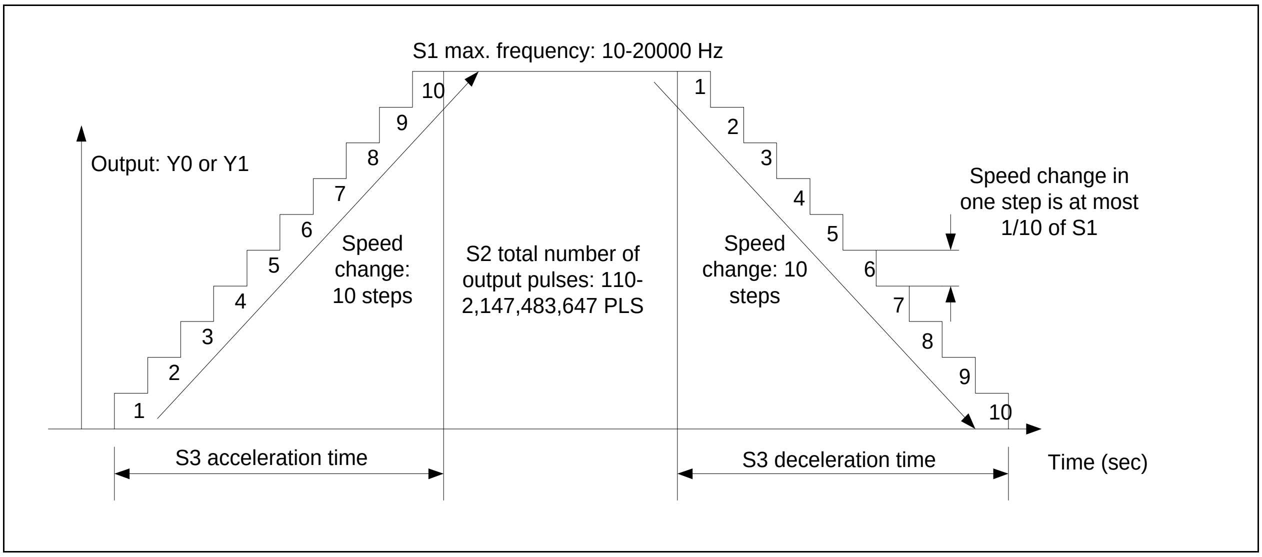
The speed change is evenly divided into 10 steps during the acceleration/deceleration is treated, and each step is ***S1***/10.

D: High-speed pulse output point.

- Function description

The PLSR instruction is a high-speed pulse output instruction with acceleration/deceleration function for fix-dimension transmission. Targeting the specified max. frequency, the pulse output is accelerated evenly and after the number of output pulses reaches the preset value, the pulse output is decelerated evenly.

The operation process is shown in the following figure:


S1 max. frequency: 10-20000 Hz
Output: Y0 or Y1
1
2
3
4
5
6
7
8
9
10
Speed change: 10 steps
S2 total number of output pulses: 110-2,147,483,647 PLS
Speed change: 10 steps
Speed change in one step is at most 1/10 of S1
S3 acceleration time
S3 deceleration time
Time (sec)

- Note

1. The output frequency of this instruction is 10–20000Hz. If the speed variation during acceleration/deceleration exceeds this range, the operand is adjusted automatically within the range. This instruction is not affected by the scan cycle.

2. The PLCs need to use the transistor output. During the high-speed pulse output, the output transistor is connected according to the specified load current. The waveforms obtained from the output terminals are related to the user's output load. When the load cannot exceed the rated load current, the smaller the load, the closer the output waveform is to the set operand.

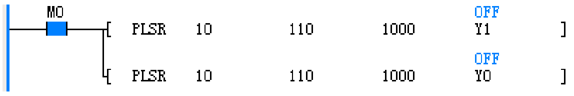
3. Whenthe high-speed instruction is executed effectively (including output completion), other operations on the same portare invalid. Only when the high-speed pulse output instruction is invalid can other instructions operate this port.

4. Usingtwo PLSR instructions can obtain separate high-speed pulse output at output points, or using the PWM (or PLSR) instruction to obtain separate high-speed pulse output at different output points.

5. When multiple PWM, PLSY or PLSR instructions operate on the same port, the first valid instruction controls the output state of the portand the later valid instructions do not affect the state of the output points.

6. Similar to other high-speed instructions (DHSCS, DHSCR,DHSZ, DHSP, DHST, and HCNT), the PLSR instruction needs to meet the requirements for high-speed input and high-speed pulse output in the system.

- Application instance



LD M0

PLSR 10 110 1000 Y1

PLSR 10 110 1000 Y0

1. When M0 is ON, pulses are output through Y0 and Y1 ports at the set frequency, and pulse output is stopped after 110 pulses are outputted. When M0 changes from OFF to ON, the next round of pulse output starts. When M0 is OFF, the port output is OFF.

2. All operandsare not changed during the execution of the instruction, they are processed according to the operand which is valid first. The new operand becomes valid only when M0 changes from ON→OFF→ON.

3. SM270 is used to control the output of Y0 while SM290 is used to control the output of Y1, and when SM270 or SM290 is 1, output pulses corresponding to the output points are disabled.

4. SM271 and SM291 indicate the output flags of Y0 and Y1 respectively. The corresponding SM271 or SM291 is OFF when the output is completed or M0 is OFF. The corresponding SM271 or SM291 is ON when the output is going on.

5. SD160－SD180 correspond to:

   1) SD160: The MSB of the number of the Y0 output pulses in PLSY and PLSR instructions.

   2) SD161: The LSB of the number of the Y0 output pulses in PLSY and PLSR instructions.

   3) SD180: The MSB of the number of the Y1 output pulses in PLSY and PLSR instructions.

   4) SD181: The LSB of the number of the Y1 output pulses in PLSY and PLSR instructions.

6. SD160–SD180 can be modified through "DMOV ××× SD160"or"MOV ××× SD180", or through the monitoring.

6.10.12 PLS: Envelopepulse output instruction

LAD: ┤├──[PLS *(S1)* *(S2)* *(D1)*]	**Applicable model**	VC1 VC2 VC3
	Influenced flag bit	
IL: PLS *(S1)* *(S2)* *(D1)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D1	BOOL			Y												

- Operand description

S1: Start address of D element designated by the parameter

S2: Number of the output segments. Range: 0﹣255.

D1: High-speed pulse output point.

- Function description

1. Using the Auto Studio instruction wizard to generate the PLS instruction, which can be called like subprograms. When the energy flow is turned on, the system outputsthe corresponding pulses according to the configuration. You can control ON or OFF of the pulse generator and set the frequency and pulse number.

2. There is no output when the segment number is 0.

3. High-speed pulse output can be turned off by setting SM270,SM290 and SM310 to ON, and other flag bits are shared with high-speed outputs.

4. The contents of the subprogram PLS_SET generated by the Auto Studio software are as follows (set n as its D element number and M as the total number of the segments):

LD SM0

DMOV Frequency of the first segment procedure Dn

DMOV Pulse number of the first segment procedure Dn＋2

DMOV Frequency of the second segment procedure Dn＋4

DMOV Pulse number of the second segment procedure Dn＋6

DMOV Frequency of the third segment procedure Dn＋8

DMOV Pulse number of the third segment procedure Dn＋10

………

DMOV Frequency of the M segment procedure Dn＋4M－4

DMOV Pulse number of the M segment procedure Dn＋4M－2

DMOV Max. speed Dn＋4M

MOV Min. speed Dn＋4M＋2

MOV Acc eleration time Dn＋4M＋3

MOV Deceleration time Dn＋4M＋4

- Note

1. It is recommended to use the PLS instruction generated by the PTO instruction wizard. If you write the PLS instruction manually, note that the number of pulses in each procedure cannot be too small. At the set acceleration speed, the number of pulses in each procedure must be larger than the min. number of pulses required for the frequency conversion.

2. P indicates the number of pulses in a certain procedure, F_N indicates the frequency of the N segment, $F_{\max}$ and $F_{\min}$ indicate the max. speed and min. speed, and T_{up} and T_{down} indicate the acceleration and deceleration time in ms.

1) When the speed of the procedure N is larger than that of procedure N-1, the number of the pulses in the procedure N need to meet the following conditions:

$$P \geq \frac{(F_N + F_{N-1}) \times (F_N - F_{N-1}) \times T_{up}}{2000 \times (F_{\max} - F_{\min})}$$

2) When the speed of procedure N is less than that of procedure N-1, the number of the pulses in the procedure N need to meet the following conditions:

$$P \geq \frac{(F_N + F_{N-1}) \times (F_N - F_{N-1}) \times T_{down}}{2000 \times (F_{\max} - F_{\min})}$$

3. In particular:

1) When N=1, the frequency of the procedure N－1 takes $F_{\min}$, and then $F_{\min}$ is put into the above formula.

2) When the number of procedures is 1, namely there is only one segment, the number of the pulses needs to meet the following condition:

$$P \geq \frac{(F_1 + F_{\min}) \times (F_1 - F_{\min}) \times (T_{up} + T_{down})}{2000 \times (F_{\max} - F_{\min})}$$

3) The number of the pulses in the last procedure needs tomeet the following formula:

$$P \geq \frac{(F_M + F_{M-1}) \times (F_M - F_{M-1}) \times (T_{up} + T_{down})}{2000 \times (F_{\max} - F_{\min})}$$

4) The frequency designated in each procedure needs to be within the setting range of the max. or min.speed.

4. The PLCs need to use the transistor output. During the high-speed pulse output, the output transistor is connected according to the specified load current. The waveforms obtained from the output terminalsare related to the user's output load. When the load cannot exceed the rated load current, the smaller the load, the closer the output waveform is to the set operand.

5. When the high-speed instruction is executed effectively (including output completion), other operations on the same portare invalid. Only when the high-speed pulse output instruction is invalid can other instructions operate this port.

6. PLSY, PLSR, PLS, and positioning instructions can output the high-speed pulses through ports. It is not allowed to use these instructions for high-speed pulse output on the same port at the same time.

6.10.13 PWM: Pulse output instruction

LAD: PWM (S1) (S2) (D)									**Applicable model**		VC1S VC1 VC2 VC3					
									Influenced flag bit							
IL: PWM *(S1) (S2) (D)*									**Step length**		7					
Opera	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	BOOL			Y												

- Operand description

S1: Assigned pulse width (ms/us)

Settable range: 0﹣32767 (ms), when ***S1*** is larger than 32767, the system reports that the instruction operand is illegal and no system hardware resource is occupied.

If the content of ***S1*** is changed while the instruction is running, the output pulse is also changed. When SM272 is 0, ***S1*** is in the unit of ***ms***; when SM272 is 1, ***S1*** is in the unit of us.

S2: Assigned pulse cycle (ms)

Settable range: 1﹣32767, when the set operand is not within the setting range, the system reports that the instruction operand is illegal, the pulse does not output, and no system hardware resource is occupied.

If the content of ***S2*** is changed while the instruction is running, the output pulse is also changed. When SM84 is 0, ***S1*** is in the unit of ***ms***; when SM84 is 1, ***S1*** is in the unit of us.

S2 needs to be larger than or equal to ***S1***, otherwise the system reports an operand error, and no system resource is occupied.

D: High-speed pulse output point.

- Function description

The PWM pulse whose output width is ***S1*** and the pulse cycle is ***S2*** is outputted at the port designated by ***D***.

Note

1. When ***S1*** is 0, the high-speed output port output is always OFF. When ***S1***=***S2***, the high-speed output port output is always ON.

2. The waveforms obtained from the output terminals are related to the user's output load. When the max. output current is fulfilled, the smaller the load, the closer the output waveform is to the set operand. In order to output the high-speed pulses, the load current on the output transistors of the PLCs needs to be large enough within the range of the rated load current.

3. When the high-speed instruction is executed effectively (including output completion), other operations on the same portare invalid. Only when the high-speed pulse output instruction is invalid can other instructions operate this port.

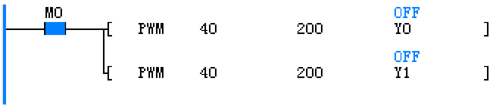
4. Using two PWM instructions can obtain separate pulse output atoutput ports, or using the PLSY (or PLSR) instruction to obtain separate high-speed pulse output at different output ports.

5. When multiple PWM, PLSY or PLSR instructions operate on the same port, the first valid instruction controls the output state of the port and the later valid instructions do not affect the state of the output points.

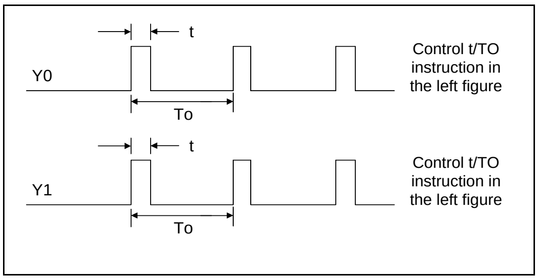
6. Similar to other high-speed instructions (DHSCS, DHSCR,DHSZ, DHSP, DHST, and

HCNT), the PWM instruction needs to meet the requirements for high-speed I/O in the system.

- Application instance


M0
PWM 40 200 OFF Y0
PWM 40 200 OFF Y1

LD M0
PWM 40 200 Y0
PWM 40 200 Y1


t
Y0
To
Control t/TO instruction in the left figure
t
Y1
To
Control t/TO instruction in the left figure

Where "t" is pulse width and T0 is pulse cycle.

1. When M0 is ON, Y0 and Y1 output the PWM pulseswith the width of 40ms and the cycle of 200 ms. When M0 is OFF, theoutput is OFF. The output state is not affected by the scan cycle.

2. SM270 is used to disable the output of Y0, while SM290 is used to disable the output of Y1. When SM270 and SM290 are ON, the output is stopped.

3. SM271 and SM291 indicate the output flag of Y0 and Y1 respectively. When M0 is OFF, SM271 and SM291 are OFF.

6.11 Control calculation instructions

6.11.1 PID: Function instruction

LAD: ├──┤ ├──[PID (S1) (S2) (S3) (D)		**Applicable model**	VC1S VC1 VC2 VC3 VC5
		Influenced flag bit	
IL: PID ***(S1)*** ***(S2)*** ***(S3)*** ***(D)***		**Step length**	9

Operand	Type	Applicable soft element													Indexing
S1	INT								D					R	√
S2	INT								D					R	√
S3	INT								D					R	√
D	INT								D					R	√

- Operand description

D: Operation result output (MV) when executing the program

S1: Set target value (SV)

S2: Current measurement value (PV)

S3: Sampling time (Ts). Range: 1–32767 (ms). It need to be set longer than the calculation time.

S3+1: Word for setting the action, alarm and upper/lower limit function.

Bit	Set value and definition	
	0	1
0	Positive feedback	Adverse feedback
1	Input variation alarm invalid	Input variation alarm valid
2	Output variation alarm invalid	Output variation alarm valid
3 ~ 4	Reserved	
5	Output value upper/lower limit value setup is invalid	Output value upper/lower limit value setup is valid
6 ~ 15	Reserved	

S3+2: Input filtering constant (α). Range:0–99[%]. When it is 0,there is no input filtering.

S3+3: Proportional gain (Kp). Range:1–32767[%].

S3+4: Integral time (TI). Range:0–32767 (×100ms). When it is 0, it is processed as ∞ (no integral).

S3+5: Differential gain (KD). Range:0–100[%]. When it is 0, there is no differential gain.

S3+6: Differential time (TD).Range:0–32767 (×10ms). When it is 0, it is processed as no differential.

S3+7–***S3+14***: Internal data register for PID operation.

S3+15: Alarm set value of the PID input variation (positive change). Range:0–32767 (when bit 1 of ***S3+1*** is 1).

S3+16: Alarm set value of the PID input variation (negative change). Range:0–32767 (when bit 1 of ***S3+1*** is 1).

S3+17: Alarm set value of the PID output variation (positive change). Range:0–32767 (when bit 2 and bit 5 of ***S3+1*** are 1 and 0 respectively).

Set value of the output upper limit. Range:-32768–+32767 (when bit 2 and bit 5 of ***S3+1*** are 0 and 1 respectively).

S3+18: Alarm set value of the PID output variation (negative change). Range: 0–32767 (when bit 2 and bit 5 of ***S3+1*** are 1 and 0 respectively).

Set value of the output lower limit. Range: -32768–+32767 (when bit2 and bit5 of ***S3+1*** are 0 and 1 respectively).

S3+19: PID alarm output.

- Bit 0: Input variation (positive change) overflows.
- Bit 1: Input variation (negative change) overflows.
- Bit 2: Output variation (positive change) overflows.
- Bit 3: Output variation (negative change) overflows.

In which, S3–S3＋6are mandatory user set operands while S3＋15–S3＋19 areoptional user set operands. You can set the operand s through the PID instruction wizard of Auto Studio.

- Function description

1. Performing the PID operation when the energy flow is valid and the sampling time is reached.

2. Multiple PID instruction can be executed simultaneously by multiple times (on limit on the loop number). But you need to note that the S1, S2, S3 or D soft element number used by the operation cannot be overlayed.

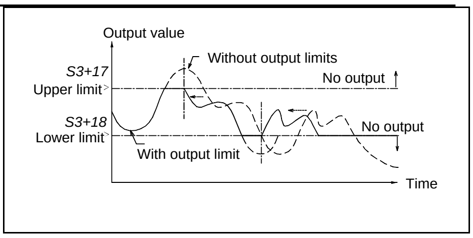
3. The PID instruction is applicable to the timed interrupt subprograms, general subprograms, and main programs. Under this condition, you need to confirm operand set unit and clear the internal processing data of ***S3+7*** before executing the PID instruction.。

4. The input filtering constant can smooth the changes of measured value.

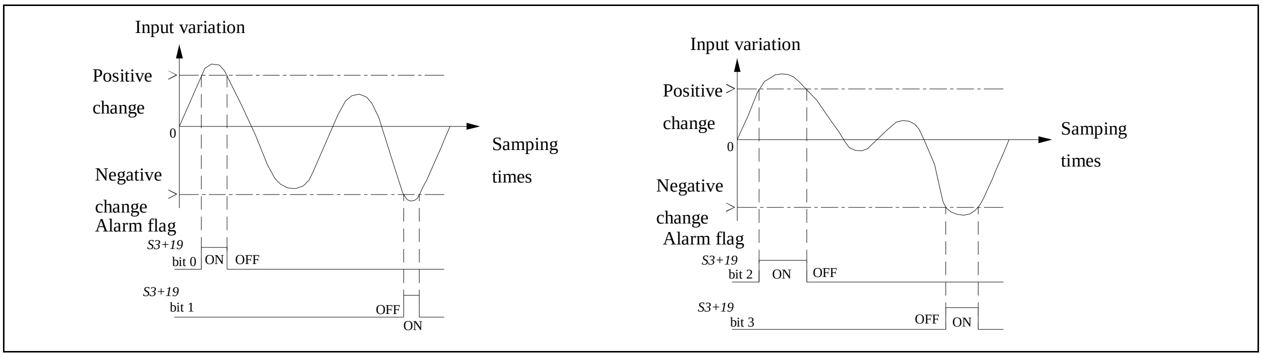
5. The differential gain can ease the dramatic change of output value.

6. Action direction: Setting the forward (positive feedback) and reverse actions (adverse feedback) of the system via bit 0 of ***S3＋1***.

7. Output upper/lower limit setting: When you set the output upper/lower limit to be valid (bit 5 and bit 2 of ***S3＋1***are ON and OFF respectively) , the integral item of PID can be prevented from becoming too large. The output value is shown as below:



8. Alarm setting: When you set the output upper/lower limit to be valid (in ***S3＋1,*** bit 1 is ON, bit 2 is ON, and bit 5 is OFF) , the PID instruction compares the I/O variation with the preset value in ***S3＋15*** －***S3＋18***. If the current value is larger than the preset value, PID reports an alarm, and the corresponding function bits in ***S3＋19*** are set immediately after executing the PID instruction. In this way, you can monitor the IO variation. The current output values are shown as below:



9. Basic expression of the PID instruction:

Action direction	PID expression
Positive action	$\Delta MV = KP\left\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I}EV_n + D_n\right\}$ $EV_n = PV_{nf-1} - SV$ $D_n = \frac{T_D}{T_S + \alpha_D * T_D}(PV_{nf} + PV_{nf-2} - 2PV_{nf-1}) + \frac{\alpha_D * T_D}{T_S + \alpha_D * T_D} * D_{n-1}$ $MV_n = \sum \Delta MV$
Reverse action	$\Delta MV = KP\left\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I}EV_n + D_n\right\}$ $EV_n = SV - PV_{nf-1}$ $D_n = \frac{T_D}{T_S + \alpha_D * T_D}(2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{\alpha_D * T_D}{T_S + \alpha_D * T_D} * D_{n-1}$ $MV_n = \sum \Delta MV$

Symbol description is shown below:

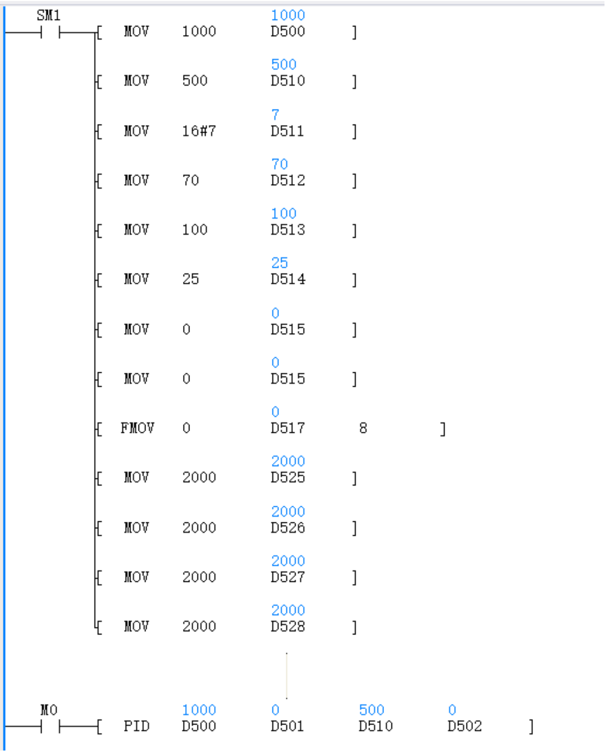
Symbol	Description		Symbol	Description
EV_n	Current sampling deviation		D_n	Current differential item
EV_{n-1}	Deviation before one cycle		D_{n-1}	Differential item before one cycle
SV	Target value		KP	Proportional gain
PV_{nf}	Current sampling value (after filtering)		T_S	Sampling cycle
PV_{nf-1}	Sampling value before one cycle (after filtering)		T_I	Integral time
PV_{nf-2}	Sampling value before two cycles (after filtering)		T_D	Differential time
ΔMV	Output variation		α_D	Differential gain
MV	Current operation quantity			

- Application instance

// PID initialization, if the control operands are the same, you can initialize the operands only once.

```
LD      SM1                 //Initialization, executed once only
MOV     1000      D500      //Setting the target value
MOV     500     D510          //Sampling time (Ts). Range: 1 – 32767 (ms). It needs to be larger than the
                        //calculation time
MOV     7       D511    //Action direction
MOV     70      D512          //Input filtering constant (α). Range: 0 – 99[％]. When it is 0, there is no input filtering
MOV     100     D513          //Proportional gain (Kp). Range: 1 – 32767[％]
MOV     25      D514          //Integral time (TI). Range: 0 – 32767 (×100ms). When it is 0, it is processed as
                        //∞ (no integral)
MOV     0       D515          //Differential gain (KD). Range: 0 – 100[％]. When it is 0, there is no differential gain
MOV     63      D516          //Differential time (TD). Range: 0 – 32767 (×10 ms). When it is 0, it means no differential
                              //processing
FMOV        0       D517       8    //Clearing the area for storing the transit data of PID operation
MOV     2000    D525          //Alarm set value of the input variation (positive change). Range: 0 – 32767
MOV     2000    D526          //Alarm set value of the input variation (negative change). Range:0 – 32767
MOV     2000    D527          //Alarm set value of the output variation (positive change). Range: 0 – 32767
MOV     2000    D528          //Alarm set value of the output variation (negative change). Range:0 – 32767
//PID instruction execution operation
LD      M0                  //User-controlled PID operation program
PID     D500 D501 D510 D502   //PID instruction: PID S1 S2 S3
```

The LAD of the above instruction is shown below:



```
SM1
──┤ ├──┬─[ MOV    1000     1000 D500    ]
       ├─[ MOV    500      500 D510     ]
       ├─[ MOV    16#7     7 D511       ]
       ├─[ MOV    70       70 D512      ]
       ├─[ MOV    100      100 D513     ]
       ├─[ MOV    25       25 D514      ]
       ├─[ MOV    0        0 D515       ]
       ├─[ MOV    0        0 D515       ]
       ├─[ FMOV   0        0 D517     8       ]
       ├─[ MOV    2000     2000 D525    ]
       ├─[ MOV    2000     2000 D526    ]
       ├─[ MOV    2000     2000 D527    ]
       └─[ MOV    2000     2000 D528    ]
M0
──┤ ├────[ PID    1000 D500    0 D501    500 D510    0 D502    ]
```

The PLCs initialize the PID operands only in the first scan cycle. When M0 is ON, the current measured value is read from the external A/D modules (the actual situation could be different), assigned to the measurement value unit D501, and the PID operation is executed. The operation result is converted into analog signals through the external D/A modules (the actual situation could be different), and fed to the controlled system.

- Note

1. For ***D***, you need to designate it into the data register outside of the Saving Range. Otherwise, it needs tobe cleared (LD SM0 MOV 0 D****) in the first operation.
2. The PID instruction needs to occupy 20 consecutive data registers starting from ***S3***.
3. The max. error of sampling time TS is － (one scancycle＋1ms) – ＋ (one scan cycle). When the value of TSis relatively small, the PID effect is affected. It is recommended to use the PID instruction in timed interrupt.
4. When you set the PID output upper/lower limit to be valid, if the value of upper limit is less than that of the lower limit, the system reports an operand error, and does not execute the PID operation.
5. When you set the alarm of I/O variation to be valid, the set value of ***S3＋15*** – ***S3＋18*** cannot be negative, otherwise the system reports an operand error, and does not execute the PID operation.
6. When bit 2 and bit 5 of ***S3＋1*** are set to ON, the system determines that the setting is invalid (it is equivalent of that bit 2 and bit 5 are set to OFF), and therefore does not report alarms for upper limit, lower limit, or variation exceeding.
7. When the set value of the PID control operand (***S3*** – ***S3＋6*** unit) is outside the valid range, the system reports an operand error, and does not execute the PID operation.
8. When the sampling time is less than or equal to one scan cycle, if there is data overflow or result overflow during the operation, no alarm occurs, and the PID operation continues.
9. All PID operands need to be initialized before the initial execution of the PID instruction. If the operands remain

the same during the operation, and the related operand elements are not covered by other programs, you can initialize the PID operands only once. If the data in the transit data registers are changed during the PID operation, the operation result is incorrect.

6.11.2 RAMP: Ramp signal output instruction

LAD: ├─┤ ├─[RAMP (S1) (S2) (D1) (S3) (D2)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: RAMP *(S1)* *(S2)* *(D1)* *(S3)* *(D2)*	**Step length**	**12**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D1	INT								D				V		R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D2	BOOL			Y	M	S	LM				C	T				

- Operand description

  S1: Startvalue

  S2: End value

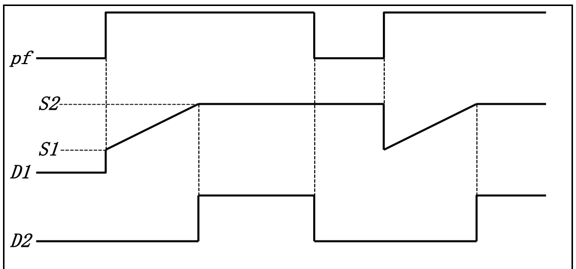
  D1: Output value

  S3: Number of steps (***S3***＞0, otherwise the system reports an operand error, and does not execute the operation)

  D2: Output state 0

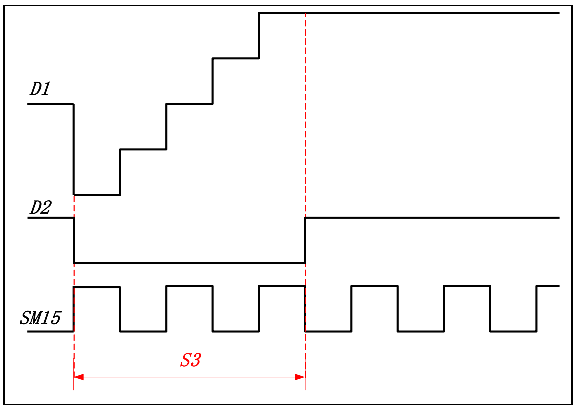
- Function description

  When a rising edge occurred to the energy flow and keeps ON, each scan cycle, based on the height and mobile scan times of the ramps, determines the incremental quantity and current output value. After reaching ***S2***, the output value (***D1***) stays in current state, and the state output position is ON. If a falling edge occurred to the energy flow, the output state (***D2***) is OFF, theoutput value (***D1***) stays in current state until the rising edge occurs to the energy flow again, the output value (***D1***) is initialized to be the value of ***S1***, and continues to generate next ramp operation, as shown below:



pf: state of the energy flow

The execution process of the ramp instructionis shown below (***S3***=5):



- Note

  1. If the result is not divisible when calculating the step length,you can adopt the "rounded off" method.

  2. The instruction generates the ramp data only once upon every rising edge.

  3. When ***S1***=***S2***, ***D1***=***S2***, and ***D2***=ON.

  4. The total number of RAMP, HACKLE, and TRIANGLE instructions cannot exceed 100 in the program.

- Application instance

  //Initializing the register upon the first scan cycle after the power-on

  LD SM1

  MOV 0 D0

  MOV 2000 D1

  //Executing the ramp function instruction when X0=ON

  LD X0

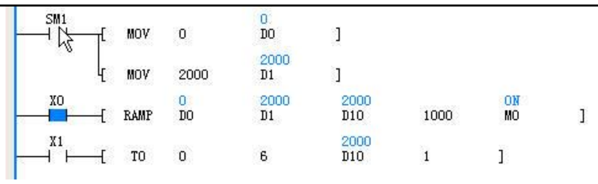
  RAMP D0 D1 D10 1000 M0

  //Sending the output value of ramp function to the external DA module when X1=ON, so as to generate ramp waveforms

  LD X1

  TO 0 6 D10 1

  The LAD of the above instructions is shown below:



1. When X0=ON, D10 (in the first cycle,D10=D0=0) increases by 2 (2000/1000) in each scan cycle. When D10=D1=2000, D10 keeps unchanged, and M0=ON.During the generation process of the ramp function, if the energy flow falls, the output state ***D2*** is OFF, the output value ***D1*** keeps its current state until the next rising edge arrives, D10=D0, and a new ramp process starts.

2. You can use an external special module to convert the data into the analog waveform.

6.11.3 HACKLE: Sawtooth wave signal output instruction

LAD: ├─┤ ├──[HACKLE (S1) (S2) (D1) (S3) (D2)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: HACKLE *(S1) (S2) (D1) (S3) (D2)*	**Step length**	**12**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D1	INT								D				V		R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D2	BOOL			Y	M	S	LM				C	T				

- Operand description

  S1: Startvalue

  S2: End value

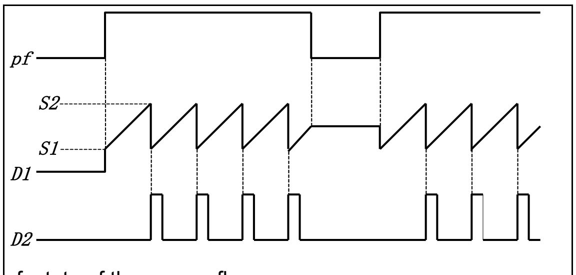
  D1: Output value

  S3: Number of steps (***S3***>0, otherwise the system reports an operand error, and does not execute the operation)

  D2: Output state

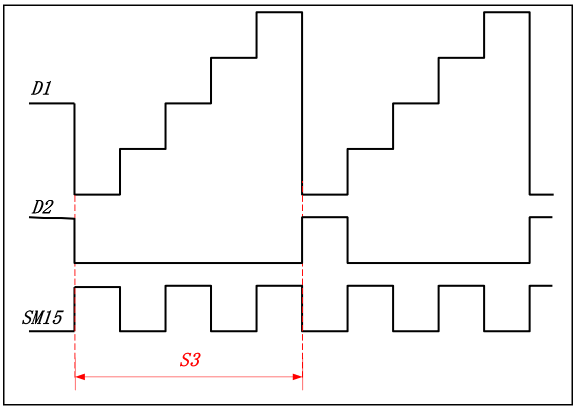
- Function description

  When theenergy flow is valid, the increment and current output value (***D1***) are determined for each scan cycle according to the height and steps of the sawtooth wave (***D1***).When the output value reaches ***S2***, it is initialized to ***S1***, and the state output bit (***D2***) is set to ON. In the next scan cycle, if the energy flow keeps ON, the state output bit (***D2***) is set to OFF, and the next sawtooth wave is generated. During the generation process of sawtooth wave function, if afalling edge occurs to the energy flow, the output state (***D2***) is OFF and the output value (***D1***) remains in the current state. When a rising edge occurs to the energy flow again, the output value (***D1***) is initialized to ***S1*** and a new sawtooth wave generation process starts, as shown below.



pf: state of the energy flow

The execution process of a sawtooth wave instruction is shown as follows (***S3***=5).



- Note

  1. If the result is not divisible when calculating the step length, you can adopt the "rounding-off" method.

2. The instruction generates a series of consecutive sawtooth wave data as long as the energy flow is valid.

3. When ***S1***=***S2***, ***D1***=***S2***, and ***D2***=ON (no counting pulse).

4. The total number of RAMP, HACKLE, and TRIANGLE instructions cannot exceed 100 in the program.

- Application instance

//Initializing the register upon the first scan cycle after the power-on

LD SM1

MOV 0 D0

MOV2000 D1

//Executing the sawtooth wave function instruction when X0=ON

LD X0

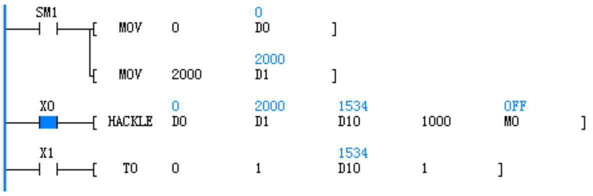
HACKLE D0 D1 D10 1000 M0

//Sending the output value of ramp function to the external DA module when X1=ON, so as to generate sawtooth waveforms

LD X1

TO 0 1 D10 1

The LAD of the above instructions is shown as follows.



1. When X0=ON, D10 (in the first cycle,D10=D0=0) increases by 2 (2000/1000) in each scan cycle. When D10=D1=2000, M0=ON. In the next scan cycle, if X0 keeps ON, D10=D0=0, and M0=OFF, the next sawtooth wave process starts. If the energy flow falls during the operation, the output state ***D2*** is OFF, the output value ***D1*** keeps its current state until the next rising edge arrives, the output value ***D1*** is initializedas ***S1***, and a new sawtooth wave process starts.

2. You can use an external special module to convert the data into the analog waveform.

6.11.4 TRIANGLE: Triangle wave signal output instruction

LAD: TRIANGLE *(S1)* *(S2)* *(D1)* *(S3)* *(D2)*	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: TRIANGLE *(S1)* *(S2)* *(D1)* *(S3)* *(D2)*	**Step length**	**12**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D1	INT								D				V		R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D2	BOOL			Y	M	S	LM				C	T				

- Operand description

S1: Startvalue

S2: End value

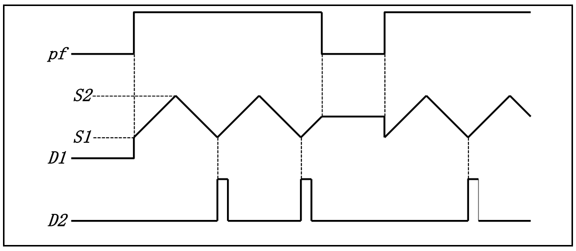
D1: Output value

S3: Number of steps (***S3***＞0, otherwise the system reports an operand error, and does not execute the operation)

D2: Output state

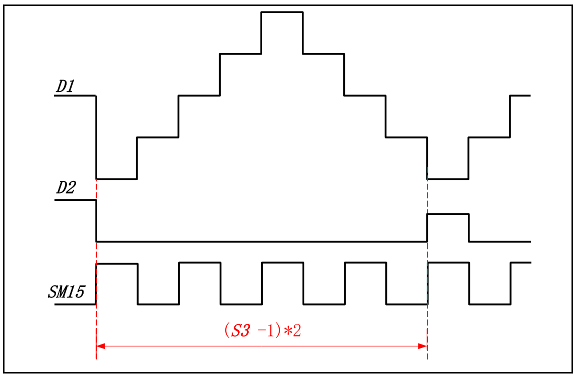
- Function description

When theenergy flow is valid, the increment and current output value (***D1***) are determined for each scan cycle according to the height and steps of the triangle wave (***D1***). When the output value reaches ***S2***, the first half slope of triangle wave has been completed, and the increment direction of the output valueis changed to continue the last half slope. When the output value (***D1***) reaches ***S1*** again, the state output bit (***D2***) is set to ON. In the next scan cycle, if the energy flow keeps ON, the state output bit (***D2***) is set to OFF, and the next triangle wave is generated. During the generation process of triangle wave function, if afalling edge occurs to the energy flow, the output state (***D2***) is OFF and the output value (***D1***)remains in the current state. When a rising edge occurs to the energy flow again, the output value (***D1***) is initialized to ***S1*** and a new triangle wave generation process starts, as shown below.



pf: state of the energy flow

The execution process of a triangle wave instruction is shown as follows (***S3***=5).



- Note

1. If the result is not divisible when calculating the step length, you can adopt the "rounding-off" method.

2. The instruction generates a series of consecutive triangle wave data as long as the energy flow is valid.

3. When ***S1***=***S2***, ***D1***=***S2***, and ***D2***=ON (no counting pulse).

4. The cycle of the triangle wave is (***S3***－1) ×2.

5. The total number of RAMP, HACKLE, and TRIANGLE instructions cannot exceed 100 in the program.

- Application instance

//Initializing the register upon the first scan cycle after the power-on

LD SM1

MOV 0 D0

MOV 2000 D1

//Executing the triangle wave function instruction when X0=ON

LD X0

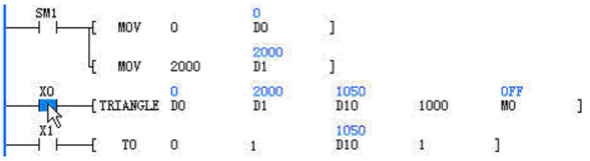
TRIANGLE D0 D1 D10 1000 M0

//Sending the output value of ramp function to the external DA module when X1=ON, so as to generate the triangle waveform

LD X1

TO 0 1 D10 1

The LAD of the above instructions is shown below:



```
 SM1                0
─┤ ├─┬─[  MOV   0        D0        ]
     │              2000
     └─[  MOV   2000     D1        ]
 X0       0         2000      1050                OFF
─┤ ├───[TRIANGLE D0     D1        D10       1000     M0      ]
 X1                           1050
─┤ ├───[  TO    0        1         D10       1        ]
```

1. When X0=ON, D10 (in the first cycle,D10=D0=0) increases by 2 (2000/1000) in each scan cycle.When D10=D1=2000, half triangle wave is completed, and D10 decreases by 2 in each scan cycle that follows. When D10=D0=0, a complete triangle wave is completed, and M0=ON. In the next scan cycle, if X0 keeps ON, and M0=OFF, the next triangle wave process starts. If the energy flow falls, the output state ***D2*** is OFF, the output value ***D1*** is initialized as ***S1***, and a new triangle wave process starts.

2. You can use an external special module to convert the data into the analog waveform.

6.11.5 ABSD: Cam absolute control instruction

<table>
<tr><td colspan="8" rowspan="2">LAD:
[ABSD <i>(S1)</i> <i>(S2)</i> <i>(D)</i> <i>(S3)</i>]</td><td colspan="2">Applicable model</td><td colspan="6">VC2 VC3 VC5</td></tr>
<tr><td colspan="2">Influenced flag bit</td><td colspan="6">Zero flag, carry flag, and borrow flag</td></tr>
<tr><td colspan="8">IL: ABSD <i>(S1)</i> <i>(S2)</i> <i>(D)</i> <i>(S3)</i></td><td colspan="2">Step length</td><td colspan="6">9</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S1</td><td>INT</td><td></td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td></td><td></td><td>D</td><td></td><td>C</td><td>T</td><td></td><td>R</td><td>√</td></tr>
<tr><td>S2</td><td>INT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>C</td><td></td><td></td><td></td><td>√</td></tr>
<tr><td>D</td><td>BOOL</td><td></td><td></td><td>Y</td><td>M</td><td>S</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>S3</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

- Operand description

S1: Start element number of storing the table data (rising edge, and falling edge). n＝4.

S2: Number of the counterused for monitoring the current value compared with the table data.

D: Outputting the number of the start bit element.

S3: Number ofrows in the table and number ofpoints of the outputted bit elements. Range: 1≤***S3***≤64.

- Function description

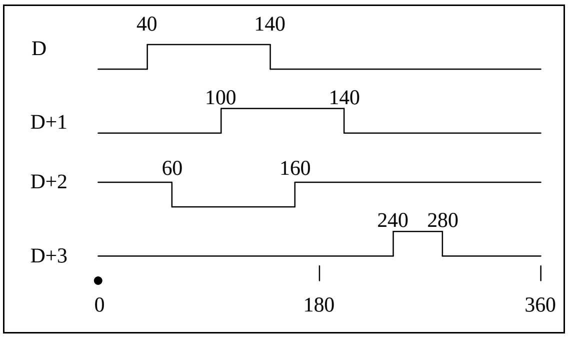
1. Comparing n rows of table data starting from ***S1***(occupying n rows×2 points) with current value ***S2*** of the counter, and performing ON/OFF control on the consecutive n-point ***D*** output.

2. Each rising/falling point can be changed by rewriting the data of ***S1***﹣***S1***＋n×2.

3. Firstly writing the data shown below into S1－S1＋2n＋1:

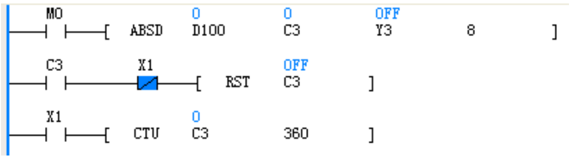
Rising point	Value example	Falling point	Value example	Output
S1	40	***S1***+1	140	***D***
S1+2	100	***S1***+3	200	***D***+1
S1+4	160	***S1***+5	60	***D***+2
S1+6	240	***S1***+7	280	***D***+3
…	…	…	…	…
S1+2n		***S1***+2n+1		***D***+n-1

4. After the instruction input is ON, the n point starting from ***D*** also changes as follows.



- Application instance

Outputting ON/OFF by rotating the platform once (0－360 degrees, rotation angle signal that indicates 1 degree per pulse).



```
    M0                0           0           OFF
──┤ ├──[  ABSD   D100        C3          Y3        8        ]
    C3         X1                 OFF
──┤ ├──────────┤↑├──[   RST      C3          ]
    X1                0
──┤ ├──[  CTU    C3          360         ]
```

Among them, M0 is the energy flow input, and X1 is a rotation angle signal indicating 1 degree per pulse.

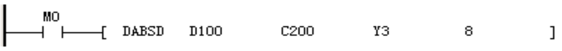
- Note

1. When the bit number of the bit soft elements is designated in ***S1***, k＝4.

2. When the counter number is designated in ***S2***, C0－C199 need to be designated for S2.

3. This instruction is affected by the scan cycle.

6.11.6 DABSD: Double word cam absolute control instruction

LAD:  	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: DABSD *(S1) (S2) (D) (S3)*	**Step length**	**11**

Operand	Type	Applicable soft element													Indexing
S1	DINT		KnX	KnY	KnM	KnS			D		C	T		R	√
S2	DINT										C				√
D	BOOL			Y	M	S									
S3	INT	Constant													

- Operand description

S1: Start element number of storing the table data (rising edge, and falling edge). n＝8.

S2: Number of the counter used for monitoring the current value compared with the table data. Range: C200﹣C255, and C256﹣C263.

D: Outputting the number of the start bit element.

S3: Number ofrows in the table and number of points of the outputted bit elements. Range: 1≤***S3***≤64.

- Function description

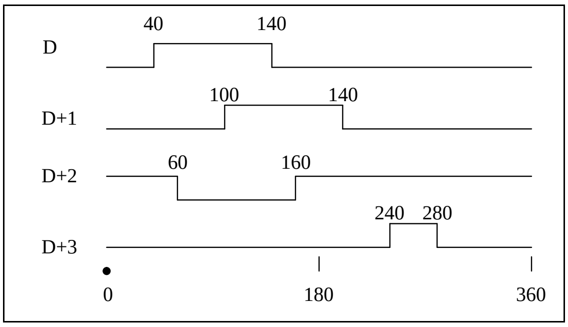
1. Comparing n rows of table data starting from ***S1***(occupying n rows×4 points) with current value ***S2*** of the counter, and performing ON/OFF control on the consecutive n-point ***D*** output.

2. Each rising/falling point can be changed by rewriting the data of [***S1***+1, ***S1***]﹣[***S1***+(n×2)+3, ***S1***+(n×2)+2].

3. Firstlyusing the transmission instruction, and writing the data shown below into [***S1***, ***S1***+1]﹣[***S1***, ***S1***+1]+4n+3.

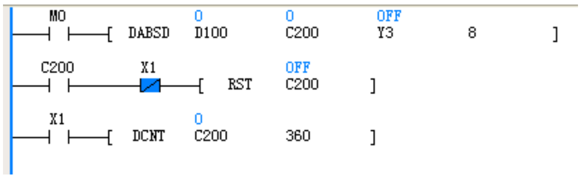
Rising point	Value example	Falling point	Value example	Output
[***S1***+1, ***S1***]	40	[***S1***+3, ***S1***+2]	140	***D***
[***S1***+5, ***S1***+4]	100	[***S1***+7, ***S1***+6]	200	***D***+1
[***S1***+9, ***S1***+8]	160	[***S1***+11, ***S1***+10]	60	***D***+2
[***S1***+13, ***S1***+12]	240	[***S1***+15, ***S1***+14]	280	***D***+3
……	……	……	……	……
[***S1***+4n+1, ***S1***+4n]		[***S1***+4n+3, ***S1***+4n+2]		***D***+n-1

4. After instruction input is ON, the n point starting with D will also change as below shows.



- Application instance

Outputting ON/OFF by rotating the platform once (0﹣360 degrees, rotation angle signal that indicates 1 degree per pulse).



Among them, M0 is the energy flow input, and X1 is a rotation angle signal indicating 1 degree per pulse.

Among them, M0 is the instruction input, and X1 is a rotation angle signal indicating 1 degree per pulse. At this time, the default value of SM200 is OFF, the DCNT instruction is incremented.

- Note

1. When the bit number of the bit soft elements is designated in ***S1***, n＝8.

2. When the counter number is designated in ***S2***,C200﹣C255 need to be designated for ***S2***.

3. This instruction is affected by the scan cycle.

6.11.7 ALT: Alternate output instruction

LAD: [ALT (D)]	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: ALT (*D*)	**Step length**	**11**

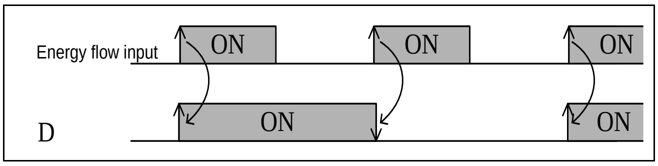
Operand	Type	Applicable soft element													Indexing
D	BOOL			Y	M	S									

- Operand description

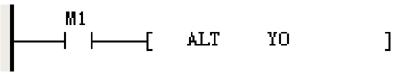
  D: Alternately outputted element address

- Function description

  When the energy flow is valid, soft elementsin each scan cycle act reversely, which is shown as below.



- Application instance



6.12 Communication instructions

6.12.1 MODBUS: Master station communication instruction

LAD: [MODBUS (S1) (S2) (S3)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: Modbus (*S1*) (*S2*) (*S3*)	**Step length**	**8**

Operand	Type	Applicable soft element													Indexing
S1	INT	Constant													
S2	INT	D	V											R	
S3	INT	D												R	√

- Operand description

  S1: Designated communication channel

  S2: Start address of data to be transmitted

  S3: Start address for storing the received data

- Function description

  1. When being as a master station, and the input conditions are met, the system transmits the data stored in the address unit starting from ***S2***, receives data, and stores it to the address unit starting from ***S3***.

  2. When being a slave station, the system needs no instruction for receiving and transmitting data.

  3. This instruction is executed upon the rising edge.

- Note

  1. Data is transmitted through Modbus. Whether you set the RTU or ASCII mode, only RTU-format data is stored in units starting from S2. The start character, end character, and checksum are not stored. In the transmission process, the required start character, end character, and check bit are automatically added to the data.

  2. You do not need to set the length for the data to be transmitted. The system automatically transmits the data according to the internally specified length based on the instruction, as shown below:

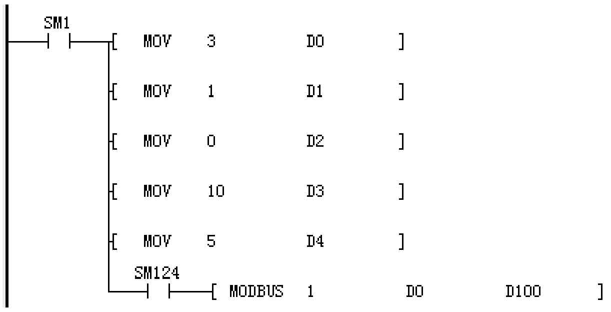
S2	Slave address
S2+1	Function code
S2+2	Data 1
	… …
S2+N+1	Data N

  3. Data is received through Modbus. Whether you set the mode to RTU or ASCII, the received data is stored in

RTUformat. That is, when you set the mode to ASCII, the system automaticallyconvertsthe data to the hex format, removes the startand end characters, and saves them in thedata area starting from ***S3***.

4. The sent and received data are stored in LSB of the word element, and MSB are not used.

- Application instance



LD SM1
MOV 3 D0
MOV 1 D1
MOV 0 D2
MOV 10 D3
MOV 5 D4
AND SM124
Modbus 1 D0 D100

1. Storing the data sent through the Modbus instruction into the element starting from D0.

2. Storing the received data in the elements starting from D100.

3. After receiving the data through Modbus, the system conducts CRC check, address check, and function code check. If there is any error, the error flag (SM136) is set, and the error details are recorded in the special register SD139.

4. Serial idle flags of SM114 and SM124 can also be used in Modbus to indicate the communication state of Modbus.

Modbus communication error code tableis shown as follows.

Abnormal code	Meaning of abnormal code
0x01	Illegal function code
0x02	Illegal register address
0x03	Wrong number of data
0x10	Communication timeout. The communication time exceeds the preset communication time limit
0x11	Error in receiving data frame
0x12	Operand error, operand (mode ormaster/slave) setting error
0x13	Error occurs when the local station number is the same as the one set by the instruction.

For details about the application instance,refer to Chapter 10"Communication function guide".

6.12.2 XMT: Free-port sending instruction

<table>
<tr><td colspan="11" rowspan="2">LAD:
─┤ ├──[XMT (S1) (S2) (S3)]</td><td colspan="3">Applicable model</td><td colspan="3">VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="3"></td></tr>
<tr><td colspan="11">IL: XMT <i>(S1) (S2) (S3)</i></td><td colspan="3">Step length</td><td colspan="3">7</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S1</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>S2</td><td>INT</td><td>D</td><td>V</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>R</td><td></td></tr>
<tr><td>S3</td><td>INT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td></td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td></td></tr>
</table>

- Operand description

S1: Designated communication channel. For VC1, the value range is 1 and 2.

S2: Start address of the data to be sent

S3: Number of bytes to be sent

- Function description

When the energy flow is turned on and the communication conditions are met, the data are sent through the designated channel and address.

- Note

1. Size of the communication frame: Size of the communication frame: The end character of the frame transmitted cannot exceed D7999 or V63, depending on the selected element type (D or V).

2. In the case of a shutdown, the transmission is stopped.

- Special register

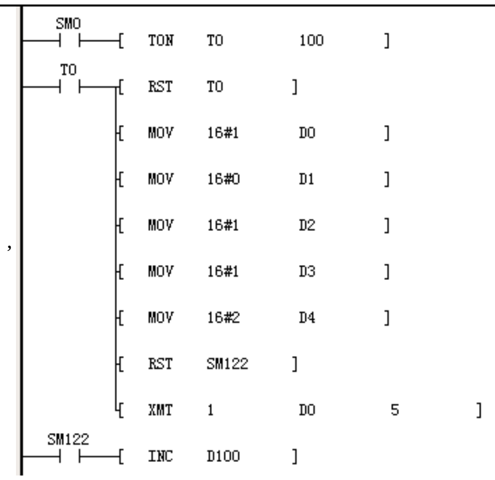
1. SM110/SM120: Transmission enable flag. This bit is set when the XMT instruction is used, cleared after the transmission is over, and reset when the current transmission is terminated.

2. SM112/SM122: Transmission completion flag. When it is judged that the transmission is completed, the transmission completion flag is set.

3. SM114/SM124: Idle flag. When the serial port has no communication task, it is set. It can be used as the check bit for the communication.

4. For details about the application instance, refer to Chapter 10"Communication function guide".

- Application instance



LD SM0
TON T0 100
LD T0
RST T0
MOV 16# 1 D0
MOV 16#0 D1
MOV 16#1 D2
MOV 16#1 D3
MOV 16#2 D4
RST SM122
XMT 1 D0 5
LD SM122
INC D100

The routine is to send one data frame at 10s interval.

The following data are sent through the serial port 1:

01	00	01	01	02

1. First, setting port 1 in the system block as the free port, and setting the baud rate, parity check, data bit, and stop bit.
2. Writing the data to be sent into the transmission buffer area. For VC2, only low bytes of the word element are sent.
3. Clearing the transmission completion flag (SM122) before sending the data.
4. Setting the transmission completion flag (SM122)when the transmission is done.

6.12.3 RCV: Free-port receiving instruction

<table>
<tr><td colspan="9" rowspan="2">LAD:
├─┤ ├──[RCV <i>(S1)</i> <i>(D)</i> <i>(S2)</i>]</td><td colspan="2">Applicable model</td><td colspan="6">VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td colspan="2">Influenced flag bit</td><td colspan="6"></td></tr>
<tr><td colspan="9">IL: RCV <i>(S1)</i> <i>(D)</i> <i>(S2)</i></td><td colspan="2">Step length</td><td colspan="6">7</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S1</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>D</td><td>INT</td><td>D</td><td>V</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>R</td><td></td></tr>
<tr><td>S2</td><td>INT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td></td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td></td></tr>
</table>

- Operand description

  S1: Designated communication channel. For VC1S, the value range is 1 and 2.

  D: Start address for storing the received data

  S2: Max. number of received bytes

- Function description

  When the energy flow is turned on and the communication conditions are met, the data are received through the designated channel and address.

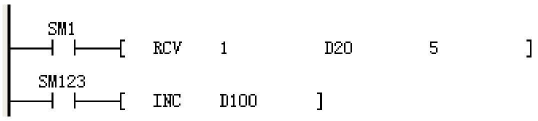
- Note

  1. Size of the communication frame: The end character of the frame received cannot exceed D7999 or V63,depending on the selected element type (D or V).

  2. In the case of a shutdown, the receiving is stopped.

  3. The value range of ***S1***: 0, 1, and 2.

- Application instance



LD SM1
RCV 1 D20 5
LD SM123
INC D100

1. When the energy flow is conducted, the RCV instruction is valid continuously. If you want to receive the data only once, you can use a rising edge or a special register that is valid only once, such as SM1, as the energy flow input.

2. For details about the application instance, refer to Chapter 10"Communication function guide".

- Special registers

  SM111 (SM121):Receiving the enable flag. This bit is set when the RCV instruction is used, cleared after the receiving is over, and reset when the current receiving is terminated.

  SM113 (SM123): Receiving the completion flag. When receiving is done, the receiving completion flag is set.

  SM114 (SM124): Idle flag. this flag is set when there is no communication task on the serial port, and it can be used as the check bit for the communication.

  SD111 (SD121): Start character which can be set in the system block.

  SD112 (SD122): End character which can be set in the system block.

  SD113 (SD123): Inter-character timeout time, namely the max. interval between receiving two characters, which can be set in the system block.

  SD114 (SD124): Frame time-out time, namely the time from when the energy flow is conducted to the time when the receiving ends, which can be set in the system block.

  SD115 (SD125): Receiving completion information code. The definitionsof the data bits are shown below:

Flag for ending user receiving	Flag for receiving the specified end word	Flag for receiving the maximum number of characters	Inter-character timeout flag	(Frame) receiving timeout flag	Parity check error flag	Reserved (You do not need to take them into consideration.)
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6 – 15

  SD116 (SD126): The characters currently received

  SD117 (SD127): Total number of the characters currently received.

6.12.4 MODRW: Modbus read/write instruction

LAD: MODRW (S1) (S2) (S3) (S4) (S5) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: MODRW *(S1) (S2) (S3) (S4) (S5) (D)*	**Step length**	**14**

Operand	Type	Applicable soft element													Indexing
S1	INT	Constant													
S2	INT	Constant	D	V										R	√
S3	INT	Constant	D	V										R	√
S4	INT	Constant	D	V										R	√
S5	INT	Constant	D	V										R	√
D	INT		D											R	√

- Operand description

S1:Designated communication channel, For VC1, the value range is 0, 1, and 2

S2: Address (Setting range of the slaveaddress: 1–247, and the broadcast address is applicable to the write element).

S3: Function code. VC1S/1L supports 01 (read coil), 02 (read discrete input), 03 (read register), 04 (read input register), 05 (write single coil), 06 (write single register), 15 (write multiple coils) , 16 (write multiple registers)

VC2/3/5 support 01 (read coil), 03 (read register), 15 (write multiple coils), and 16 (write multiple registers).

S4: Start address of to-be-read/written inverter elements.

S5: Number of to-be-read/written elements. Number of to-be-read/written elements of VC1S/1L is limited by the max frame length (256) of RTU, as shown below.

Function code	Name	Number of elements	Number ofrequired D elements
01	Read coils	1﹣2000	(S5＋15)/16
02	Read discrete input	1﹣2000	(S5＋15)/16
03	Read registers	1﹣125	S5
04	Read input registers	1﹣125	S5
05	Write single coil	0(fixed)	1
06	Write single register	0(fixed)	1
15	Write multiple coils	1﹣1968	(S5＋15)/16
16	Write multiple registers	1﹣123	S5

*The value of 05, 06 and S5 must be 0

The number of the word and bit elements cannot exceed 16 respectively, and all the bit elements are stored as one word.

D1: Storage address of to-be-read/written elements. For details about the number of elements required by VC1, refer to the above table.

- Function description

Sending the message and receiving the returned data when the energy flow is valid.

- Note

1. The number of elements cannot exceed 16.

2. Reading a maximum of 16 bit elements. The bit elements with small addresses are stored in LSB, and one byte stores 16 bits.

3. The returned abnormal code is the same as the Modbus instruction.

- Application instance

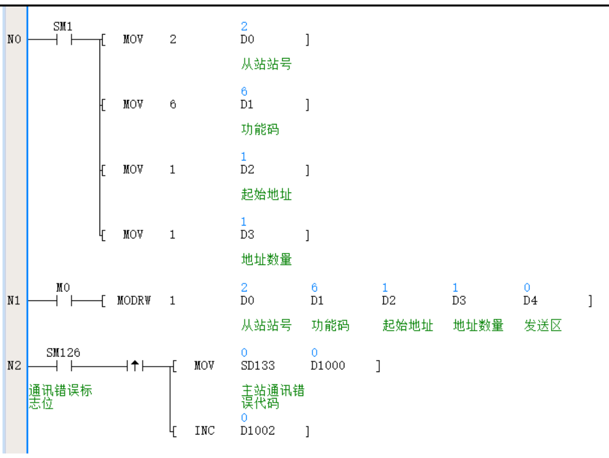
* The following application instance is currently only valid for the VC1 series PLCs.

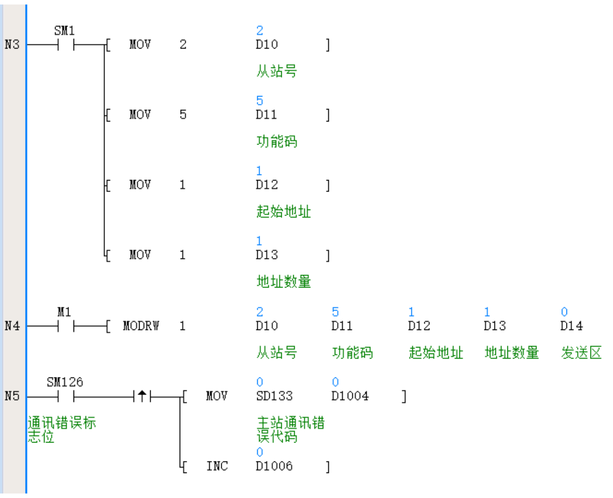
1. Standard polling

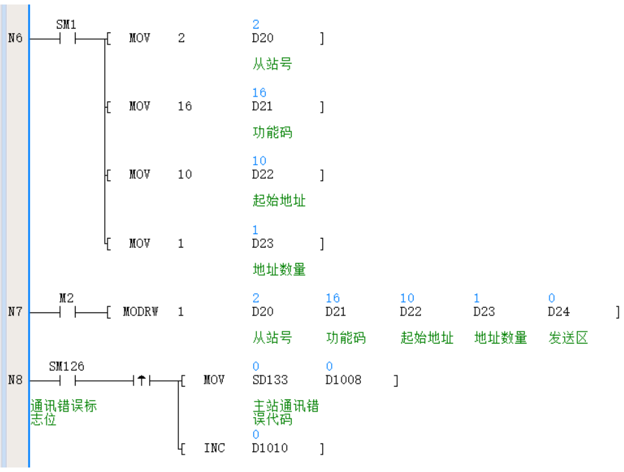
This instance is a simple polling instance, sets M0, M1, and M2, and three MODRW instruction access the device in turn according to the settings.

At runtime, if any one of M elements is reset, its corresponding MODRW instruction exits polling, but other MODRWs are still executed in polling mode. For instance, if M2 is reset, then the MODRW instructions corresponding to M0 and M1 access the device in turn.

Similarly, it is viable to insert one MODRW instruction during running. For instance, if M2 is set again, thenthree MODRW instructions access the device in turn.


N0
SM1
MOV 2 D0 从站号
MOV 6 D1 功能码
MOV 1 D2 起始地址
MOV 1 D3 地址数量
N1
M0
MODRW 1 D0 D1 D2 D3 D4
从站号 功能码 起始地址 地址数量 发送区
N2
SM126
通讯错误标志位
MOV SD133 D1000
主站通讯错误代码
INC D1002


N3
SM1
MOV 2 D10 从站号
MOV 5 D11 功能码
MOV 1 D12 起始地址
MOV 1 D13 地址数量
N4
M1
MODRW 1 D10 D11 D12 D13 D14
从站号 功能码 起始地址 地址数量 发送区
N5
SM126
通讯错误标志位
MOV SD133 D1004
主站通讯错误代码
INC D1006


N6
SM1
MOV 2 D20 从站号
MOV 16 D21 功能码
MOV 10 D22 起始地址
MOV 1 D23 地址数量
N7
M2
MODRW 1 D20 D21 D22 D23 D24
从站号 功能码 起始地址 地址数量 发送区
N8
SM126
通讯错误标志位
MOV SD133 D1008
主站通讯错误代码
INC D1010

When an error occursin the MODRW instruction,SM126 is set, SD133 indicate its error codes.The values of SM126, SD133 can be changed by other MODRW instructions, so their state need to be recorded before the next MODRW instruction is executed.

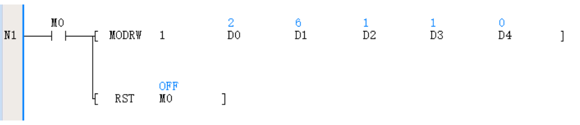
MODRW instruction error code is listed as below:

Code	Error name	Detailed description
1	Illegal function code	
2	Illegal address	
3	Illegal data	
4-15	Reserved	
16	Communication timeout	The communication time exceeds the max. communication time set by the user
17	Reserved	
18	Parameter setting error	Parameter (mode or master/slave) setting error
19	Parameter S2 error, namely slave address parameter error	The station no. is the same with the one set by the instruction, or address overlimit error occurs
20	Parameter D error, namely element address overflow	Element address overflows (The amount of the received or transmitted data exceeds the storage space of the element)
21	Instruction execution failed	
22	Address does not match	The received slave address does not match the requested slave address, and error code element stores the received slave address
23	Function code does not match	The received function code does not match the requested function code, and the error code element stores the received function code
24	Info frame error	Info frame error: it means element starting address does not match, the error code element stores the received element start address
25	Data length does not match	The received data length does not comply with protocol or the element number exceed the max limit specified by this function code
26	CRC/LRC check error	
27	Reserved	
28	Parameter S3 error, namely parameter error of the element address	Setting error in the start address of the instruction parameter element
29	Parameter S4 error, namely parameter error of the function codes	Instruction parameters set unsupported function codes or illegal function codes
30	Parameter S5 error, namely setting error in the number of the elements	Setting error in the instruction parameter elements
31	Reserved	
32	The parameter is non-modifiable	The parameter is non-modifiable

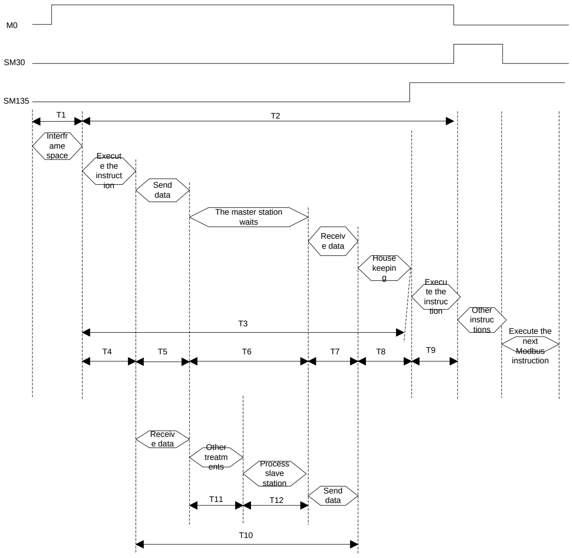
33	The parameter is non-modifiable during running	The parameter is non-modifiable during running (only EV3000 support)
34	Parameter is protected by a password	The parameter is protected by a password

2. Link time

You can use the following LAD to realize a communication between a Modbus master and slave stations. For a complete communication, the required time for each stage is shown below:


N1
M0
MODRW 1
2 D0
6 D1
1 D2
1 D3
0 D4
RST
OFF M0

One complete Modbus communication time (T_m) is comprised of T_1 and T_2, namely


M0
SM30
SM135
T1
T2
Interframe space
Execute the instruction
Send data
The master station waits
Receive data
House keeping
Execute the instruction
Other instructions
Execute the next Modbus instruction
T3
T4
T5
T6
T7
T8
T9
Receive data
Other treatments
Process slave station
Send data
T11
T12
T10

$$T_m = T_1 + T_2$$

In which, T_1 is guaranteed by the user, according to the Modbus communication protocol, the inter-frame interval time needs to be 3.5 character at least.

The length of one character is: Start bit (1 bit)+data length (7 bits or 8 bits)+check (0 bit or 1 bit)+stop bit (1 bit or 2 bits).

$$T_2 = (INT(\frac{T_3}{T_s})+1))T_s$$

In which, T_s is the max. scan cycle of the PLC.

$$T_3 = T_4 + T_5 + T_6 + T_7 + T_8 + T_9$$

T_4, T_8, and T_9 require less than 1 ms time.

$$T_5 = \frac{\text{Number of bytes to be sent} \times \text{Character length}}{\text{Baudrate (bps)}} \times 1000\,(\text{ms}) + 1\text{ms}$$

T_6: The waiting time of the master station depends on the slave station, which cannot exceed the timeout timeof the preset master mode.

$$T_7 = \frac{\text{Number of bytes to be received} \times \text{Character length}}{\text{Baudrate (bps)}} \times 1000\,(\text{ms}) + 1\text{ms}$$

The processing time of the slave station can be calculated by the following formula:

$$T_{10} = T_5 + T_{11} + T_{12} + T_7$$

In which, the max. scan time of T_{11}

T_{12} requires less than 1 ms time

For instance: Setting the communication specification as 19200, even parity, 8-bit data bit, 1-bit stop bit, RTU transmission mode, 10 characters to be sent, and 20 characters to be received. Then the processing time of the master station is calculated as follows:

$$T_5 = \frac{10 \times 10}{19200} \times 1000 + 1 = 6.2ms$$

$$T_7 = \frac{20 \times 10}{19200} \times 1000 + 1 = 11.4ms$$

$$T_4 = T_8 = T_9 \approx 1ms$$

Suppose $T_6 = 35ms$, then

$$T_3 = 1 + 6.2 + 35 + 11.4 + 1 + 1 = 55.6ms$$

Suppose the max. scan cycle is 10 ms

Then

$$T_2 = (INT(\frac{55.6}{15}) + 1))15 = 60ms$$

The processing time of the slave station

$$T_{10} = 6.2 + 15 + 1 + 11.4 = 33.6ms$$

6.13 Check instructions

6.13.1 CCITT: Check instruction

LAD: ├─┤ ├─[CCITT (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: CCITT (S1) (S2) (D)	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT								D				V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT								D				V		R	√

- Operand description

S1: Start unit ofdata to be checked

S2: Number of data to be checked (***S2***≥0, otherwise the system report an operand error)

D: Check result

- Function description

1. Performing the CCITT check operation on ***S2*** data starting from (***S1***) , and assigning the operation result to ***D***.

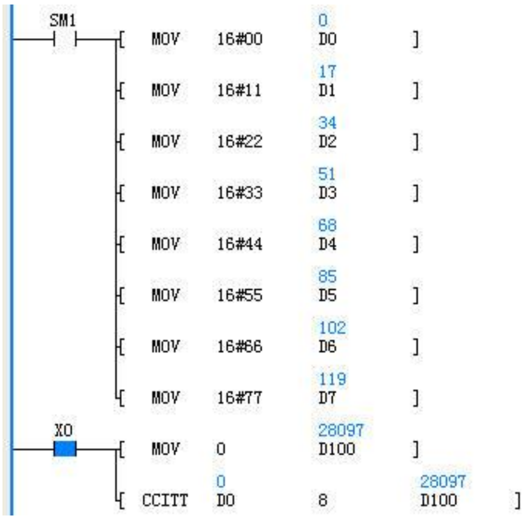
2. The polynomial of the CCITT check algorithm is: X^16＋X^12＋X^5＋1.

- Note

1. The system usually brings the content of ***D*** before the instruction execution into the operation when executing the instruction each time, so ***D*** needs to be initialized before executing the instruction.

2. Data in the data check area starting from ***S2*** unit are stored in byte mode by default, that is, the high byte is taken as 0, and the check result is 16bits.

- Application instance



```
LD      SM1
MOV     16#00 D0
MOV     16#11 D1
MOV     16#22 D2
MOV     16#33 D3
MOV     16#44 D4
MOV     16#55 D5
MOV     16#66 D6
MOV     16#77 D7
LD      X0
MOV     0   D100
CCITT D0 8 D100
```

Performing the CCITT check operation on 8 data starting from D0, and assigning the obtained result to D100 when X0=ON.

6.13.2 CRC16: Check instruction

LAD: ├─┤ ├─[CRC16 (S1) (S2) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: CRC16 (S1) (S2) (D)	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT								D				V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT								D				V		R	√

- Operand description

S1: Start unit ofdata to be checked

S2: Number ofdata to be checked (***S2***≥0, otherwise the system report an operand error)

D: Check result

- Function description

1. Performing the CRC16 check operation on ***S2*** data starting from (***S1***) , and assigning the operation result to ***D***.

2. The polynomial of the CRC16 check algorithm is: X^16＋X^15＋X^2＋1.

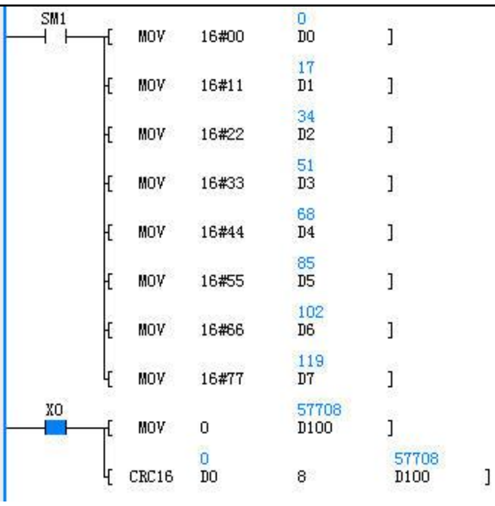
- Note

1. The system usually brings the content of ***D*** before the instruction execution into the operation when executing the instruction each time, so ***D*** needs to be initialized before executing the instruction.

2. Assigning 16#FFFF to the initial value of D element (checksum), and swopping the high/low bytes (high 8 bits and low 8 bits) when you use the standard Modbus CRC check.

3. Data in the data check area starting from ***S2*** unit are stored in byte mode by default, that is, the high byte is taken as 0, and the check result is 16bits.

- Application instance


SM1
MOV 16#00 0 D0
MOV 16#11 17 D1
MOV 16#22 34 D2
MOV 16#33 51 D3
MOV 16#44 68 D4
MOV 16#55 85 D5
MOV 16#66 102 D6
MOV 16#77 119 D7
X0
MOV 0 57708 D100
CRC16 0 D0 8 57708 D100

```
LD      SM1
MOV     16#00   D0
MOV     16#11   D1
MOV     16#22   D2
MOV     16#33   D3
MOV     16#44   D4
MOV     16#55   D5
MOV     16#66   D6
MOV     16#77   D7
LD      X0
MOV     0     D100
CRC16 D0 8 D100
```

Performing theCRC16 check operation on 8 data starting from D0, and assigning the obtained result to D100 when X0=ON.

6.13.3 LRC: Check instruction

LAD: ├─┤ ├──[LRC *(S1)* *(S2)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LRC *(S1)* *(S2)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT								D				V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT								D				V		R	√

- Operand description

S1: Start unit of data to be checked

S2: Number of data to be checked (***S2***≥0, otherwise the system report an operand error)

D: Check result

- Function description

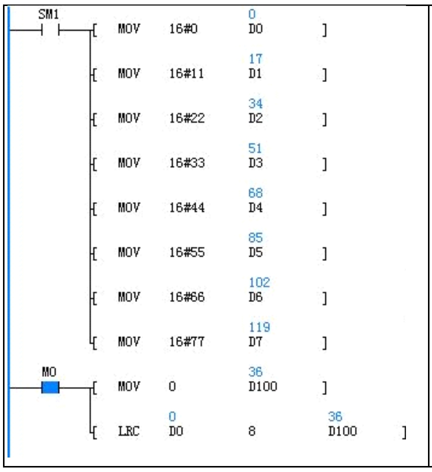
Performing theLRC check operation on ***S2*** data starting from (***S1***) , and assigning the operation result to ***D***.

- Note

1. The system usually brings the content of ***D*** before the instruction execution into the operation when executing the instruction each time, so ***D*** needs to be initialized before executing the instruction.

2. Data in the data check area starting from ***S2*** unit are stored in byte mode by default, that is, the high byte is taken as 0, and the check result is 8 bits stored in the low bytes of ***D***.

- Application instance


SM1
MOV 16#0 0 D0
MOV 16#11 17 D1
MOV 16#22 34 D2
MOV 16#33 51 D3
MOV 16#44 68 D4
MOV 16#55 85 D5
MOV 16#66 102 D6
MOV 16#77 119 D7
M0
MOV 0 36 D100
LRC 0 D0 8 36 D100

```
LD      SM1
MOV     16#00   D0
MOV     16#11   D1
MOV     16#22   D2
MOV     16#33   D3
MOV     16#44   D4
MOV     16#55   D5
MOV     16#66   D6
MOV     16#77   D7
LD      M0
MOV     0      D100
LRC     D0     8 D100
```

Performing the LRC check operation on 8 data starting from D0, and assigning the obtained result to D100 when X0=ON.

6.14 Enhanced bit processing instructions

6.14.1 ZRST: Instruction for resetting bits to 0 in batch

LAD: ┣━┫ ┣━[ZRST (D) (S)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: ZRST (*D*) (*S*)	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
D	BOOL			Y	M	S	LM				C	T				√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  D: Destination operand

  S: Source operand

- Function description

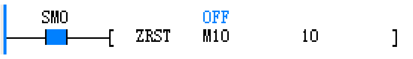
  Clearing ***S*** consecutive bit elements starting from ***D*** when the energy flow is valid.

- Note

  1. Clearing the counter value in C element when the C element is cleared.

  2. Clearing the timing value in T element when the T element is cleared.

- Application instance



LD SM0
ZRST M10 10

Clearing the data of 10 units starting from M10 (namely M10, M11, M12…M19) when SM0=ON.

6.14.2 ZSET: Instruction for resetting bits in batch

LAD: ┣━┫ ┣━[ZSET (D) (S)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: ZSET (*D*) (*S*)	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
D	BOOL			Y	M	S	LM				C	T				√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

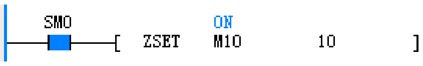
  D: Destination operand

  S: Source operand

- Function description

  Setting ***S*** consecutive bit elements starting from ***D*** to 1when the energy flow is valid.

- Application instance



LD SM0
ZSET M10 10

Setting the data of 10 units starting from M10 (namely M10, M11, M12…M19) to 1 when SM0=ON.

6.14.3 DECO: Decoding instruction

LAD: ┣━┫ ┣━[DECO (S) (D)]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DECO (*S*) (*D*)	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S: Source operand

  D: Destination operand

- Function description

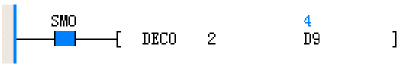
  Setting the bit ***S*** in the word element ***D*** to 1, and clearing other bits when the energy flow is valid.

- Note

  1. The valid range of ***S*** is 0 to 15.

  2. When ***S*** is larger than 15 or less than 0, and the energy flow is valid, the value of ***D*** is not changed, but the system reports a value error of the instruction operand.

- Application instance



LD SM0
DECO 2 D9

Setting the bit 2 in D9 to 1, and clearing other bits when the energy flow is valid.

6.14.4 ENCO: Encoding instruction

LAD: ├─┤ ├─[ENCO *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: ENCO *(S)* *(D)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	Z	R	√

- Operand description

  S: Source operand;

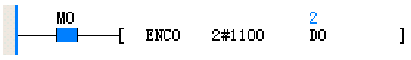
  D: Destination operand

- Function description

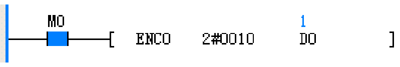
  Writing the bit number whose value is "1" in the word element ***S*** into ***D*** when the energy flow is valid.

- Note

  When the value of multiple bits in ***S*** are "1", the smallest position number will be written to ***D***, as shown in the following figure.



- Application instance



LD M0
ENCO 2#0010 D0

Operand 1 is 2#0010, and the bit 1 is "1", hence the result is 1, which is written into D0 when the energy flow is valid.

6.14.5 BITS: Instruction for counting on bit in word

LAD: ├─┤ ├─[BITS *(S)* *(D)*]	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: BITS *(S)* *(D)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

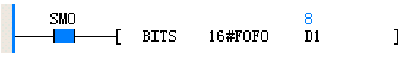
- Operand description

  S: Source operand; ***D***: Destination operand

- Function description

  Counting the number of "1" bitin the operand ***S***, and storing the statistical result in the operand ***D*** when the energy flow is valid.

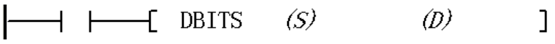
- Application instance



LD SM0
BITS 16#F0F0 D1

When the energy flow is valid, ***S*** in the BITS instruction is constant 16#F0F0 which includes 8 bits whose value is "1" (ON state). The calculation result is 8 and stored in ***D*** (D1).

6.14.6 DBITS: Instruction for counting on bit in double word

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: DBITS (S) (D)	**Step length**	**6**

Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	INT			KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

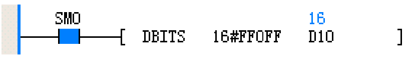
- Operand description

  S: Source operand; ***D***: Destination operand

- Function description

  Counting the number of "1" bit in the double word ***S***, and storing the statistical result in ***D*** when the energy flow is valid.

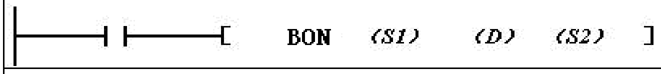
- Application instance



LD SM0
DBITS 16#FF0FF D10

When the energy flow is valid, ***S*** in the BITS instruction is constant 16#FF0FF which includes 16 bits whose value is "1" (ON state). The calculation result is 16 and stored in ***D*** (D10).

6.14.7 BON: Instruction for judging on bit in word

LAD: 	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	
IL: BON *(S1)* *(D)* *(S2)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V			√
D	BOOL		Y	M	S											
S2	INT								D				V		R	

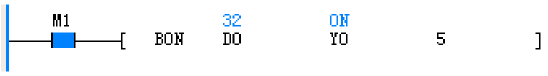
- Operand description

  S: Source operand; ***D***: Destination operand

- Function description

  Counting the state of the bit ***S2*** in the word ***S1***, and outputting the obtained result to ***D*** when the energy flow is valid.

- Application instance

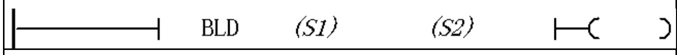


LD M1
BON D0 Y0 5

When the energy flow is valid, ***S1*** in the BON instruction is a constant D0, and its fifth bit state (ON) is output to D (Y0).

6.15 Word contact instructions

6.15.1 BLD: Word bit contact LD instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: BLD *(S1)* *(S2)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

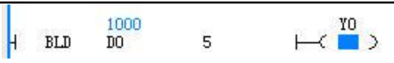
S1: Source operand

S2: Designated bit,0≤***S2***≤15, otherwise the system reports an operand error.

- Function description

  Taking the state of the bit ***S2*** in ***S1,*** and using the obtained valueto drive the subsequent operation.

- Application instance



BLD D0 5
OUT Y0

Taking the state of the bit 5 (ON) in D0 (1000: 2#0000001111101000), and using the obtained value to determine the state of Y0 in the subsequent operation.

6.15.2 BLDI: Word bit contact LDI instruction

LAD: BLDI *(S1)* *(S2)*	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: BLDI *(S1)* *(S2)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

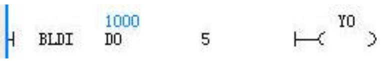
  S1: Source operand

  S2: Designated bit,0≤***S2***≤15, otherwise the system reports an operand error.

- Function description

  Performing the NOT operation on the state of the bit ***S2*** in ***S1***, and using the obtained value to drive the subsequent operation.

- Application instance



BLDI D0 5
OUT Y0

Performing the NOT operation (OFF) on the state of the bit 5 (ON) in D0 (1000: 2#0000001111101000), and using the obtained value to determine the output state of Y0 in the subsequent operation.

6.15.3 BAND: Word bit contact AND instruction

LAD^{note}: BLD *(S1)* *(S2)* Note: The logical relationship has been manifested in the diagram, so the BAND instruction is displayed as BLD in LAD	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: BAND *(S1)* *(S2)*	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

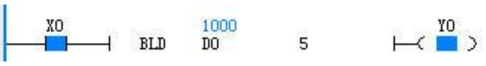
  S1: Source operand

  S2: Designated bit (0≤***S2***≤15, otherwise the system reports an operand error)

- Function description

  Taking the state of the bit ***S2*** in ***S1***, and using the obtained value in serial connection with other nodes to drive the subsequent operation.

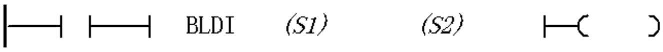
- Application instance



LD X0
BAND D0 5
OUT Y0

Taking the state of the bit 5 (ON) in D0 (1000: 2#0000001111101000), and using the obtained value in serial connection with other nodes (X0=ON) to determine the output state ofY0 in the subsequent operation.

6.15.4 BANI: Word bit contact ANI instruction

<table>
<tr><td rowspan="2">LAD^{note}:

Note: The logical relationship has been manifested in the diagram, so the BANI instruction is displayed as BLDI in LAD</td><td>Applicable model</td><td>VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td>Influenced flag bit</td><td></td></tr>
<tr><td>IL: BANI <i>(S1)</i> <i>(S2)</i></td><td>Step length</td><td>5</td></tr>
</table>

Operand	Type	Applicable soft element														Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

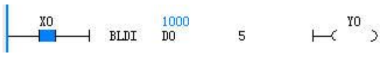
  S1: Source operand

  S2: Designated bit (0≤***S2***≤15, otherwise the system reports an operand error)

- Function description

  Performing the NOT operation on the state of the bit ***S2*** in ***S1***, and using the obtained value in serial connection with other nodes to drive the subsequent operation.

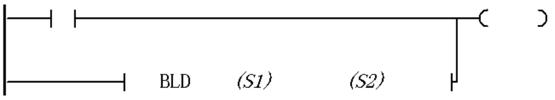
- Application instance



LD X0
BANI D0 5
OUT Y0

Performing the NOT operation (OFF) on the state of the bit 5 (ON) in D0（1000: 2#0000001111101000）, and using the obtained value in serial connection with other nodes (X0=ON) to determine the output state of Y0 in the subsequent operation.

6.15.5 BOR: Word bit contact OR instruction

<table>
<tr><td rowspan="2">LAD^{note}:

Note: The logical relationship has been manifested in the diagram, so the BOR instruction is displayed as BLD in LAD</td><td>Applicable model</td><td>VC1S VC1 VC2 VC3 VC5</td></tr>
<tr><td>Influenced flag bit</td><td></td></tr>
<tr><td>IL: BOR <i>(S1)</i> <i>(S2)</i></td><td>Step length</td><td>5</td></tr>
</table>

Operand	Type	Applicable soft element														Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

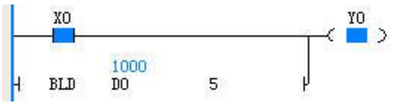
  S1: Source operand

  S2: Assignment bit (0≤***S2***≤15, otherwise operand error will be reported)

- Function description

  Taking the state of the bit ***S2*** in ***S1***, and using the obtained value in parallel connection with other nodes to drive the subsequent operation.

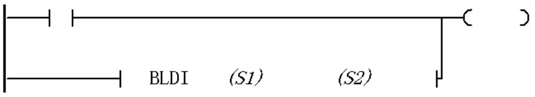
- Application instance



LD X0
BOR D0 5
OUT Y0

Taking the state of the bit 5 (ON) inD0 (1000: 2#0000001111101000), and using the obtained value in parallel connection with other nodes (X0=ON) to determine the output state ofY0 in the subsequent operation.

6.15.6 BORI: Word bit contact ORI instruction

LAD$^{\text{note}}$:  Note: The logical relationship has been manifested in the diagram, so the BORI instruction is displayed as BLDI in LAD	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: BORI (*S1*) (*S2*)	**Step length**	**5**

Operand	Type	Applicable soft element													Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

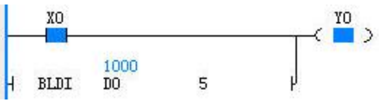
  S1: Source operand

  S2: Designated bit (0≤***S2***≤15, otherwise the system reports an operand error)

- Function description

  Performing the NOT operation on the state of the bit ***S2*** in ***S1***, and using the obtained value in parallel connection with other nodes to drive the subsequent operation.

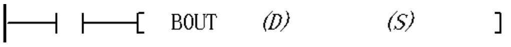
- Application instance



LD X0
BORI D0 5
OUT Y0

Performing the NOT operation (OFF) on the state of the bit 5 (ON) in D0 (1000: 2#0000001111101000), and using the obtained value in parallel connection with other nodes (X0=ON) to determine the output state of Y0 in the subsequent operation.

6.15.7 BOUT: Word bit coil output instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: BOUT (*D*) (*S*)	**Step length**	**5**

Operand	Type	Applicable soft element													Indexing
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

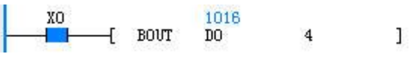
  S1: Source operand

  S2: Designated bit (0≤***S2***≤15, otherwise the system reports an operand error)

- Function description

  Assigningthe state of the current energy flow to the bit ***S*** of ***D***.

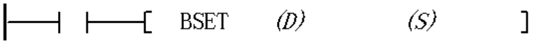
- Application instance



LD X0
BOUT D0 4

Assigning the state of the current energy flow(X0=ON) to the bit 4 in D0 (1000:2#0000001111101000). After execution: D0=1016 (2#0000001111111000).

6.15.8 BSET: Word bit coil setinstruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: BSET (*D*) (*S*)	**Step length**	**5**

Operand	Type	Applicable soft element	Indexing

D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

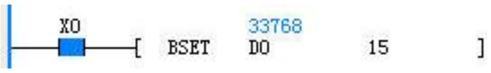
  D: Destination operand

  S2: Designated bit (0≤***S2***≤15, otherwise the system reports an operand error)

- Function description

  Setting the bit ***S*** of ***D*** element.

- Application instance



LD X0
BSET D0 15

Setting thebit 15 in D0 (1000: 2#0000001111101000) when the energy flow is valid. After execution, D0=33768 (2#1000001111101000).

6.15.9 BRST: Word bit coil reset instruction

LAD: BRST (D) (S)	Applicable model	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: **BRST (*D*) (*S*)**	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

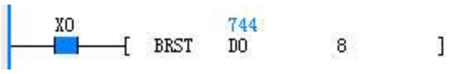
  D: Destination operand

  S2: Designated bit (0≤***S2***≤15, otherwise the system reports an operand error)

- Function description

  Clearing the bit ***S*** of ***D*** element.

- Application instance



LD X0
BRST D0 8

Clearing thebit 8 in D0 (1000: 2#0000001111101000) when the energy flow is valid. After execution, D0=744 (2#0000001011101000).

6.16 Comparison contact instructions

6.16.1 LD (=, <, >, <>, >=, <=): Integer comparison LD※instruction

LAD:	Applicable model	VC1S VC1 VC2 VC3 VC5
= (S1) (S2) < (S1) (S2) > (S1) (S2) <> (S1) (S2) >= (S1) (S2) <= (S1) (S2)	**Influenced flag bit**	
IL: LD= (S1) (S2) **LD< (S1) (S2)** **LD> (S1) (S2)** **LD<> (S1) (S2)** **LD>= (S1) (S2)** **LD<= (S1) (S2)**	**Step length**	**5**

Operand	Type	Applicable soft element	Indexing

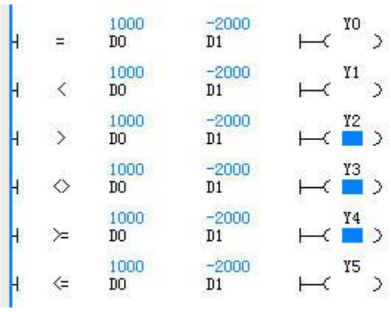
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Comparison parameter 1

  S2: Comparison parameter 2
- Function description

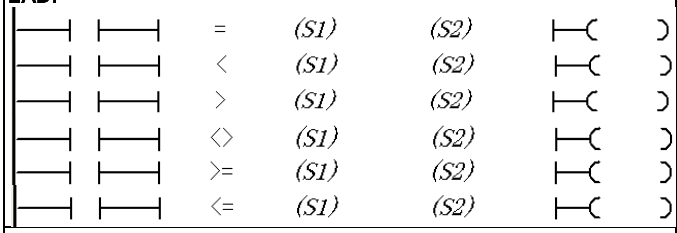
  Conducting the BIN comparison on the content of ***S1*** and ***S2***, and using the comparison result to drive the subsequent operation.
- Application instance


=
1000
D0
-2000
D1
Y0
<
1000
D0
-2000
D1
Y1
>
1000
D0
-2000
D1
Y2
<>
1000
D0
-2000
D1
Y3
>=
1000
D0
-2000
D1
Y4
<=
1000
D0
-2000
D1
Y5

```
LD=    D0 D1
OUT    Y0
LD<    D0 D1
OUT    Y1
LD>    D0 D1
OUT    2
LD<>   D0 D1
OUT    Y3
LD>=   D0 D1
OUT    Y4
LD<=   D0 D1
OUT    Y5
```

Conducting the BIN comparison on the data of D0 and D1, and using the comparisonresult to determine the output state of the subsequent elements.

6.16.2 AND (=, <, >, <>, >=, <=): Integer comparison AND ※instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: AND= **(S1)** **(S2)** **AND<** **(S1)** **(S2)** **AND>** **(S1)** **(S2)** **AND<>** **(S1)** **(S2)** **AND>=** **(S1)** **(S2)** **AND<=** **(S1)** **(S2)**	**Step length**	**5**

=
(S1)
(S2)
<
(S1)
(S2)
>
(S1)
(S2)
<>
(S1)
(S2)
>=
(S1)
(S2)
<=
(S1)
(S2)

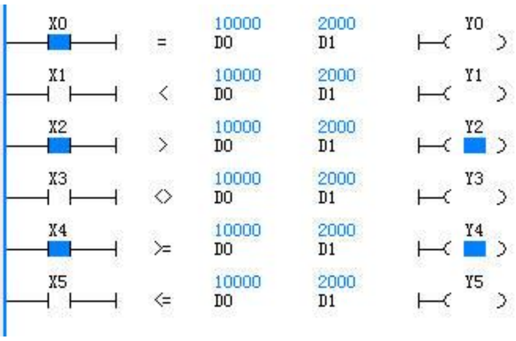
Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

  S1: Comparison parameter 1

  S2: Comparison parameter 2
- Function description

  Conducting the BIN comparison on the content of ***S1*** and ***S2***, and using the comparison result in serial connection with other nodes to drive the subsequent operation.
- Application instance

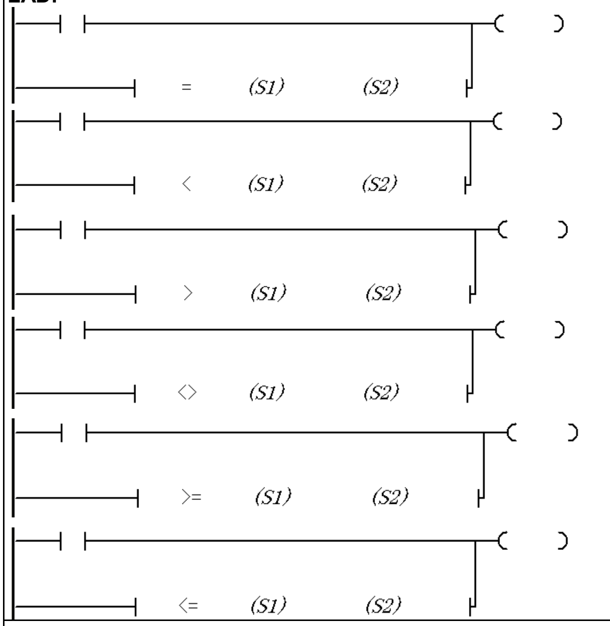

X0
=
10000
D0
2000
D1
Y0
X1
<
10000
D0
2000
D1
Y1
X2
>
10000
D0
2000
D1
Y2
X3
<>
10000
D0
2000
D1
Y3
X4
>=
10000
D0
2000
D1
Y4
X5
<=
10000
D0
2000
D1
Y5

```
LD      X0
AND=    D0 D1
OUT     Y0
LD      X1
AND<    D0 D1
OUT     Y1
LD      X2
AND>    D0 D1
OUT     Y2
LD      X3
AND<>   D0 D1
```

Conducting the BIN comparison on the data of D0 and D1, and using the comparison result in serial connection with other nodes to determine the output state of the subsequent elements.

```
OUT       Y3
LD        X4
AND>=     D0 D1
OUT       Y4
LD        X5
AND<=     D0 D1
OUT       Y5
```

6.16.3 OR (=, <, >, <>, >=, <=): Integer comparison OR※instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: OR= **(S1) (S2)** **OR< (S1) (S2)** **OR> (S1) (S2)** **OR<> (S1) (S2)** **OR>= (S1) (S2)** **OR<= (S1) (S2)**	**Step length**	**5**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- Operand description

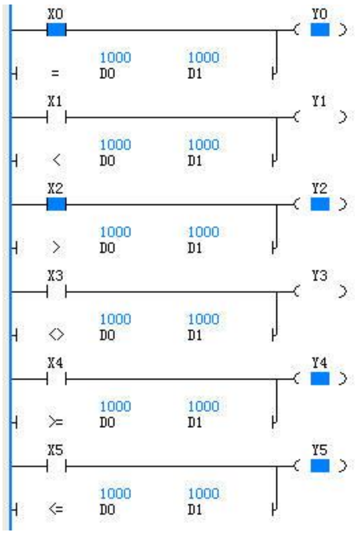
  S1: Comparison parameter 1

  S2: Comparison parameter 2

- Function description

  Conducting the comparison on the content of ***S1*** and ***S2***, and using the comparison result in parallel connection with other nodes to drive the subsequent operation.

- Application instance



Conducting the comparison on the data of D0 and D1, and using the comparisonresult in parallel connection with other nodes to determine the output state of the subsequent elements.

```
LD      X0
OR=  D0 D1
OUT  Y0
LD          X1
OR<  D0 D1
OUT         Y1
LD           X2
OR<>        D0 D1
OUT         Y2
LD           X3
OR>=        D0 D2
OUT         Y3
LD           X4
OR>=        D0 D1
OUT         Y4
LD           X5
OR<=        D0 D1
OUT         Y5
```

6.16.4 LDD (=, <, >, <>, >=, <=): Long integer comparison LDD※instruction

LAD: D= (S1) (S2) D< (S1) (S2) D> (S1) (S2) D<> (S1) (S2) D>= (S1) (S2) D<= (S1) (S2)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LDD= **(S1)** **(S2)** **LDD<** **(S1)** **(S2)** **LDD>** **(S1)** **(S2)** **LDD<>** **(S1)** **(S2)** **LDD>=** **(S1)** **(S2)** **LDD<=** **(S1)** **(S2)**	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√

- Operand description

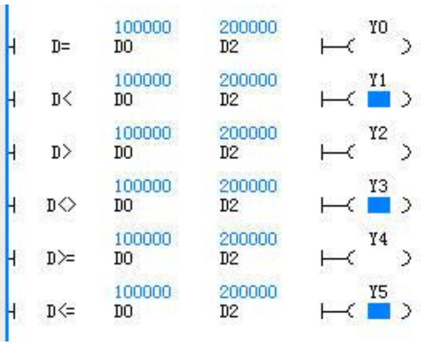
  S1: Comparison parameter 1

  S2: Comparison parameter 2

- Function description

  Conducting the comparison on the content of ***S1*** and ***S2***, and using the comparison result to drive the subsequent operation.

- Application instance



```
LD=D0 D2
OUTY0
LD<D0 D2
OUT    Y1
LD<> D0 D2
OUT      Y2
LD>= D0 D2
OUT    Y3
LD>= D0 D2
OUT    Y4
LD<=D0 D2
OUT Y5
```

Conducting the comparison on (D0,D1) and (D2,D3), and using the comparisonresult to determine the output state of the subsequent elements.

6.16.5 ANDD (=, <, >, <>, >=, <=): Long integer comparison ANDD※instruction

LAD: D= (S1) (S2) D< (S1) (S2) D> (S1) (S2) D<> (S1) (S2) D>= (S1) (S2) D<= (S1) (S2)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: ANDD= (S1) (S2) **ANDD< (S1) (S2)** **ANDD> (S1) (S2)** **ANDD<> (S1) (S2)** **ANDD>= (S1) (S2)** **ANDD<= (S1) (S2)**	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√

- Operand description

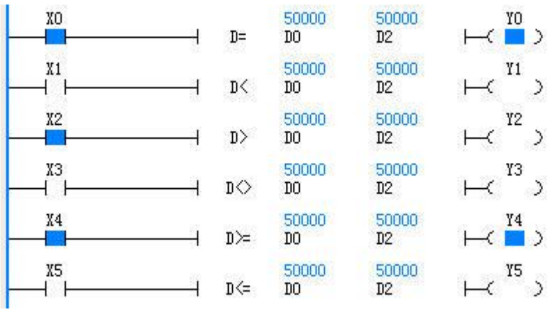
  S1: Comparison parameter 1

  S2: Comparison parameter 2

- Function description

  Conducting the comparison on the content of ***S1*** and ***S2***, and using the comparison result in serial connection with other nodes to drive the subsequent operation.

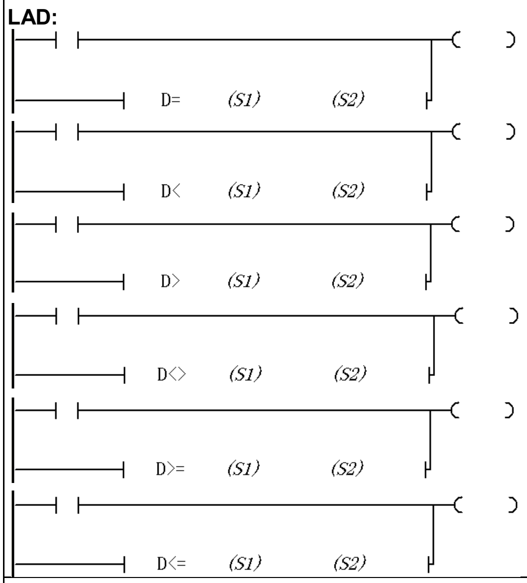
- Application instance


X0
D=
50000
D0
50000
D2
Y0
X1
D<
50000
D0
50000
D2
Y1
X2
D>
50000
D0
50000
D2
Y2
X3
D<>
50000
D0
50000
D2
Y3
X4
D>=
50000
D0
50000
D2
Y4
X5
D<=
50000
D0
50000
D2
Y5

```
LD      X0
LDD=    D0 D2
OUT     Y0
LD      X1
LDD<    D0 D2
OUT     Y1
LD      X2
LDD<>   D0 D2
OUT     Y2
LD      X3
LDD<>   D0 D2
OUT     Y3
LD      X4
LDD>=   D0 D2
OUT     Y4
LD      X5
LDD<=   D0 D2
OUT     Y5
```

Conducting the comparison on (D0,D1) and(D2,D3), and using the comparisonresult in serial connection with other nodes to determine the output state of the subsequent elements.

6.16.6 ORD (=, <, >, <>, >=, <=): Long integer comparison ORD※instruction

LAD: 	**Applicable model** **Influenced flag bit**	VC1S VC1 VC2 VC3 VC5
IL: ORD＝ (S1) (S2) **ORD＜ (S1) (S2)** **ORD＞ (S1) (S2)** **ORD＜＞ (S1) (S2)** **ORD＞＝ (S1) (S2)** **ORD＜＝ (S1) (S2)**	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√

- Operand description

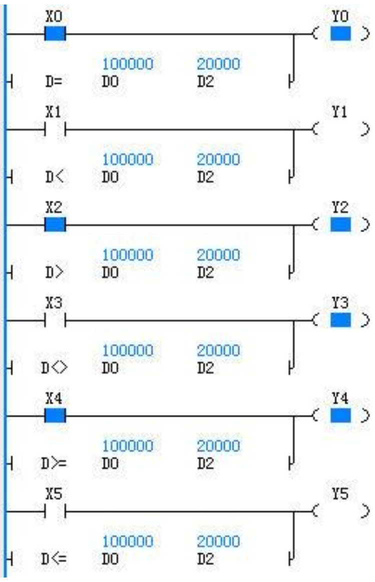
  S1: Comparison parameter 1

  S2: Comparison parameter 2

- Function description

  Conducting the comparison on the content of ***S1*** and ***S2***, and using the comparison result in parallel connection with other nodes to drive the subsequent operation.

- Application instance



```
LD        X0
ORD=      D0 D2
OUT       Y0
LD        X1
ORD<      D0 D2
OUT       Y1
LD        X2
ORD<>     D0 D2
OUT       Y2
LD        X3
ORD>=     D0 D2
OUT       Y3
LD        X4
ORD>=     D0 D2
OUT       Y4
LD        X5
ORD<=     D0 D2
OUT       Y5
```

Conducting the comparison on (D0,D1) and (D2,D3), and using the comparison result in parallel connection with other nodes to determine the output state of the subsequent elements.

6.16.7 LDR: Floating-point number comparison instruction

LAD: R= (S1) (S2) R< (S1) (S2) R> (S1) (S2) R<> (S1) (S2) R>= (S1) (S2) R<= (S1) (S2)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: LDR= **(S1)** **(S2)** **LDR<** **(S1)** **(S2)** **LDR>** **(S1)** **(S2)** **LDR<>** **(S1)** **(S2)** **LDR>=** **(S1)** **(S2)** **LDR<=** **(S1)** **(S2)**	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	REAL	Constant							D				V		R	√
S2	RAEL	Constant							D				V		R	√

- Operand description

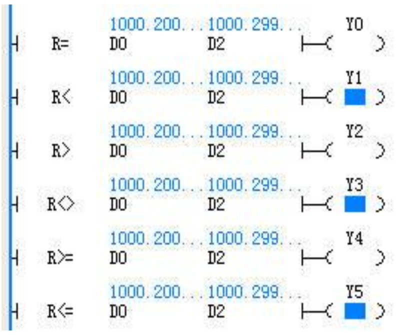
  S1: Comparison parameter 1

  S2: Comparison parameter 2

- Function description

  Conducting the comparison on the content of ***S1*** and ***S2***, and using the comparison result to drive the subsequent operation.

- Application instance



```
LDR=    D0 D2
OUT     Y0
LDR<    D0 D2
OUT      Y1
LDR>     D0 D2
OUT      Y2
LDR<>    D0 D2
OUT      Y3
LDR>=    D0 D2
OUT      Y4
LDR<=   D0 D2
OUT       Y5
```

Conducting the comparison on (D0,D1) and (D2,D3), and using the comparison result to determine the output state of the subsequent elements.

6.16.8 ANDR: Floating-point number comparison instruction

LAD: R= (S1) (S2) R< (S1) (S2) R> (S1) (S2) R<> (S1) (S2) R>= (S1) (S2) R<= (S1) (S2)	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: ANDR= (S1) (S2) **ANDR< (S1) (S2)** **ANDR> (S1) (S2)** **ANDR<> (S1) (S2)** **ANDR>= (S1) (S2)** **ANDR<= (S1) (S2)**	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√

- Operand description

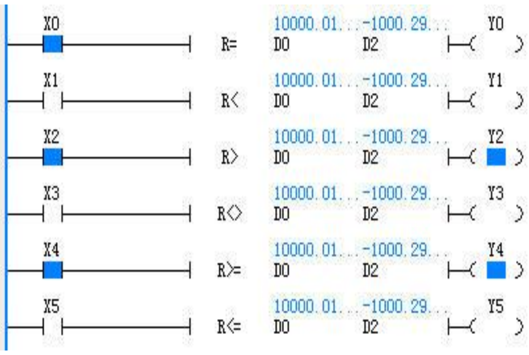
  S1: Comparison parameter 1

  S2: Comparison parameter 2

- Function description

  Conducting the comparison on the content of ***S1*** and ***S2***, and using the comparison result in serial connection with other nodes to drive the subsequent operation.

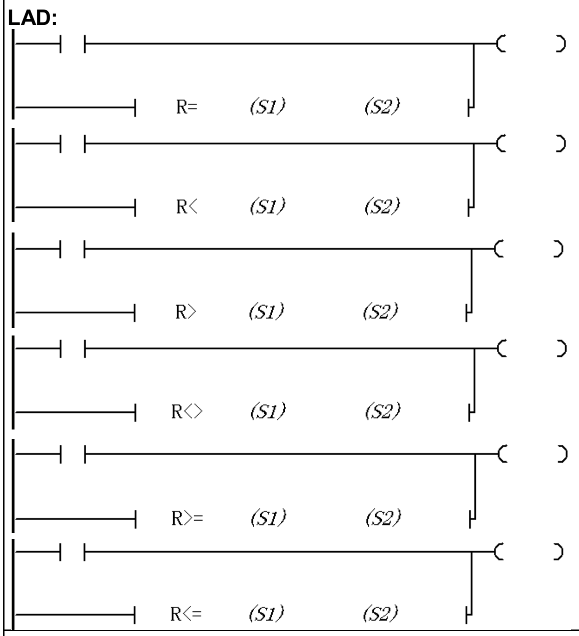
- Application instance


X0
R=
10000.01...
-1000.29...
D0
D2
Y0
X1
R<
10000.01...
-1000.29...
D0
D2
Y1
X2
R>
10000.01...
-1000.29...
D0
D2
Y2
X3
R<>
10000.01...
-1000.29...
D0
D2
Y3
X4
R>=
10000.01...
-1000.29...
D0
D2
Y4
X5
R<=
10000.01...
-1000.29...
D0
D2
Y5

```
LD          X0
ANDR=       D0 D2
OUT         Y0
LD          X1
ANDR<       D0 D2
OUT         Y1
LD          X2
ANDR<>      D0 D2
OUT         Y2
LD          X3
ANDR<>      Y3
LD          X4
ANDR>=      D0 D2
OUT         Y4
LD          X5
ANDR<=      D0 D2
OUT         Y5
```

Conducting the comparison on (D0,D1) and(D2,D3), and using the comparison result in serial connection with other nodes to determine the output state of the subsequent elements.

6.16.9 ORR: Floating-point number comparison instruction

LAD: 	**Applicable model**	VC1S VC1 VC2 VC3 VC5
	Influenced flag bit	
IL: ORR＝ (S1) (S2) **ORR＜ (S1) (S2)** **ORR＞ (S1) (S2)** **ORR＜＞ (S1) (S2)** **ORR＞＝ (S1) (S2)** **ORR＜＝ (S1) (S2)**	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√

- Operand description

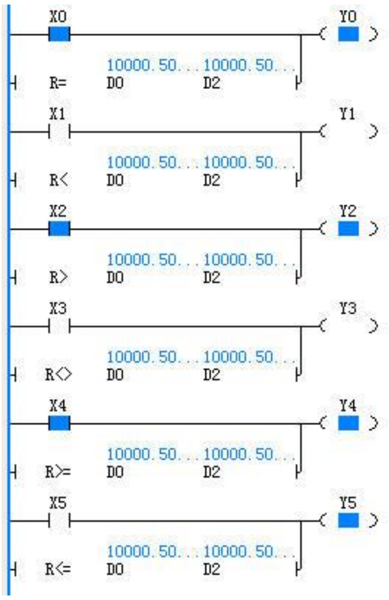
  S1: Comparison parameter 1

  S2: Comparison parameter 2

- Function description

  Conducting the comparison on the content of ***S1*** and ***S2***, and using the comparison result in parallel connection with other nodes to drive the subsequent operation.

- Application instance



LD	X0
ORR=	D0 D2
OUT	Y0
LD	X1
ORR<	D0 D2
OUT	Y1
LD	X2
ORR>	D0 D2
OUT	Y2
LD	X3
ORR<>	D0 D2
OUT	Y3
LD	X4
ORR>=	D0 D2
OUT	Y4
LD	X5
ORR<=	D0 D2
OUT	Y5

Conducting the comparison on (D0,D1) and (D2,D3), and using the comparison result in parallel connection with other nodes to determine the output state of the subsequent elements.

6.16.10 CMP: Instruction for setting integer comparison to ON

LAD: ├─┤ ├─[CMP (S1) (S2) (D)]	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	
IL: CMP *(S1) (S2) (D)*	**Step length**	**7**

Operand	Type	Applicable soft element														Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	BOOL			Y	M	S										

- Operand description

S1: Number of data or soft element to be compared.

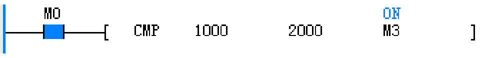
S2: Number of data or soft element that is the source for comparison.

D: Number of the start element that is used to output the result

- Function description

Executing the instruction, and comparing ***S1*** and ***S2*** when the energy flow is valid. Making one of (***D***) (***D***+1) (***D***+2) ON according to its result (less, equal, or larger).

- Application instance



LD m0
CMP 1000 2000 M3

6.16.11 LCMP: Instruction for setting long integer comparison to ON

LAD: ├─┤ ├─[*LCMP (S1) (S2) (D)*]	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	
IL: LCMP *(S1) (S2) (D)*	**Step length**	**9**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V	Z	R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V	Z	R	√
D	BOOL			Y	M	S										

- Operand description

S1: Comparison value 1.

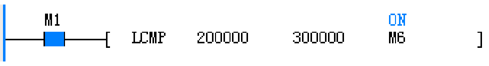
S2: Comparison value 2.

D: Number of the start element that is used to output the result.

- Function description

Executing the instruction, and comparing ***S1*** and ***S2*** when the energy flow is valid. Making one of (***D***) (***D***+1) (***D***+2) ON according to its result (less, equal, or larger).

- Application instance



LD m1
LCMP 200000 300000 M6

6.16.12 RCMP: Instruction for setting floating-point number comparison to ON

LAD: RCMP *(S1) (S2) (D)*	**Applicable model**	VC2 VC3 VC5
	Influenced flag bit	
IL: RCMP *(S1) (S2) (D)*	**Step length**	**9**

Operand	Type	Applicable soft element													Indexing
S1	REAL	Constant							D					R	√
S2	REAL	Constant							D					R	√
D	BOOL			Y	M	S									

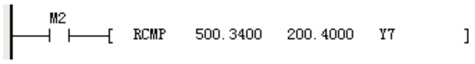
- Operand description

  S1: Comparison value 1.

  S2: Comparison value 2.

  D: Number of the start element that is used to output the result.
- Function description

  Executing the instruction, and comparing ***S1*** and ***S2*** when the energy flow is valid. Making one of (***D***) (***D***+1) (***D***+2) ON according to its result (less, equal, or larger).
- Application instance



```
LD      m2
RCMP 500.3400    200.4000
Y7
```

6.17 Bulk data processing instructions

6.17.1 BKADD: Bulk data addition operation instruction

LAD: BKADD *(S1) (S2) (D) (S3)*	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: BKADD *(S1) (S2) (D) (S3)*	**Step length**	**9**

Operand	Type	Applicable soft element													Indexing
S1	INT								D	SD	C	T	V	R	√
S2	INT	Constant							D	SD	C	T	V	R	√
D	INT								D	SD	C	T	V	R	√
S3	INT	Constant							D				V	R	

- Operand description

  S1: Number of the start soft element for savingthe data on which the addition operation is to be performed

  S2: Constant for performing the addition operation, or number of the start soft element that saves data on which the addition operation is to be performed

  D: Number of the start soft element for saving the operation result

  S3: Data quantity
- Function description

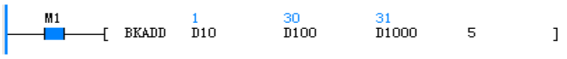
  1. Executing the instruction, increasing S3 units 16-bit data starting from S1 byS3 units 16-bit data (BIN) starting from S2, and storing the operation result in S3 units starting from D when the energy flow is valid.

  2. You can directly specify a 16-bit constant in S2. Sequentially performing, when S2 is a constant, the addition operation on S3 units16-bit data starting from S1 and S2, and storing the operation result in S3 units starting from D.

- Note

When the operation result overflows, the carry flag bit is not setto ON.

- Application instance

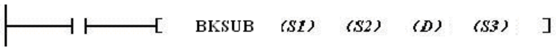


LD M1

BKADD D10 D100 D1000 5

Sequentially increasing the contents of 5 units starting from D10 by that of 5 units starting from D100, and storing the operation results in 5 units starting from D1000 when M1=ON. In this case, D1000=D10+D100, D1001=D11+D101,……, D1004=D14+D104.

6.17.2 BKSUB: Bulk data subtraction operation instruction

LAD: 	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **BKSUB *(S1) (S2) (D) (S3)***	**Step length**	**9**

Operand	Type	Applicable soft element												Indexing
S1	INT								D	SD	C	T	V	R	√
S2	INT	Constant							D	SD	C	T	V	R	√
D	INT								D	SD	C	T	V	R	√
S3	INT	Constant							D				V	R	

- Operand description

S1: Number of the start soft element for savingthe data on which the subtraction operation is to be performed

S2: Constant for performing the addition operation, or number of the start soft element that saves data on which the subtraction operation is to be performed

D: Number of the start soft element for saving the operation result

S3: Data quantity

- Function description

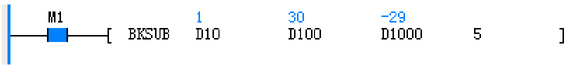
1. Executing the instruction, subtracting S3 units 16-bit data starting from S1 by S3 units 16-bit data (BIN) starting from S2, and storing the operation result in S3 units starting from D when the energy flow is valid.

2. You can directly specify a 16-bit constant in S2. Sequentially performing, when S2 is a constant, the subtraction operation on S3 units 16-bit data starting from S1 and S2, and storing the operation result in S3 units starting from D.

- Note

When the operation result overflows, the carry flag bit is not setto ON.

- Application instance

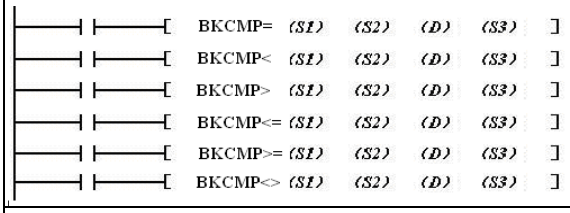


LD M1

BKSUB D10 D100 D1000 5

Sequentially subtracting the contents of 5 units starting from D10 by that of 5 units starting from D100, and storing the operation results in 5 units starting from D1000 when M1=ON. In this case, D1000=D10-D100, D1001=D11-D101, ……, D1004=D14-D104.

6.17.3 BKCMP=,>,<,<>,<=,>=: Bulk data comparison instruction

LAD: 	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: **BKCMP=,>,<,<>,<=,>= *(S1) (S2) (D) (S3)***	**Step length**	**9**

Operand	Type	Applicable soft element	Indexing

S1	INT	Constant							D	SD	C	T	V	R	√
S2	INT								D	SD	C	T	V	R	√
D	BOOL		Y	M	S	LM	SM								
S3	INT	Constant							D				V	R	

- Operand description

  S1: Number of the start soft element to be compared or saved

  S2: Number of the startsoft element that stores the comparison source

  D: Number of the start soft elementthat saves thecomparison result

  S3: Data quantity

- Function description

  1. Comparing the S3 units 16-bit data starting from S1 with S3 units 16-bit data (BIN) starting from S2, and storing the operation result in S3 units starting from D.

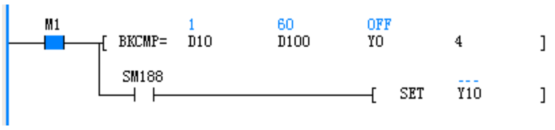
  2. You can directly specify a 16-bit constant in S1. Sequentially performing, when S1 is a constant, the comparison on S3 units16-bit data starting from S1 with that of S2, and storing the operation result in S3 units starting from D.

  3. Settingthe comparison set flag bit of the data block (SM188) when S3 units comparison results starting from D are ON.

- Note

  When the operation result overflows, the carry flag bit is not set ON.

- Application instance



LD M1

BKCMP= D10 D100 Y0 4

LD SM188

SET Y10

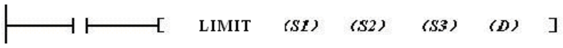
When M1=ON, the contents of the four units starting from D10 are compared with the contents of the four units starting from D100, and the result is stored in the four units starting from Y0. In addition, when the comparison result is all ON, Y10 turns ON.

Sequentially comparing the contents of 4 units starting from D10 by that of 4 units starting from D100, and storing the operation results in 4 units starting fromY0 when M1=ON. In this case, Y0 is set ON.

6.18 Datasheet instructions

6.18.1 LIMIT: Upper/lower limit control instruction

LAD:



Applicable model VC3 VC5

Influenced flag bit **Zero flag, carry flag, and borrow flag**

IL: LIMIT (S1) (S2) (S3) (D)

Step length **9**

Operand	Type	Applicable soft element													Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S3	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

- Operand description

  S1: Lower limit value

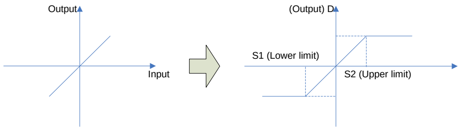
  S2: Upper limit value

  S3: Input value to be controlled by the upper and lower limits

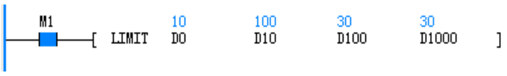
  D: Number of the start soft element that stores the output value that has been controlled by the upper and lower limits

- Function description

  You can judge whether the input value specified in S3 is within the range of the upper and lower limits specified by S1 and S2 so as to control and save the value in D. In this case, when S3<S1, D=S1; when S3>S2, D=S2; when S1 <=S3<=S2, D=S2.



- Application instance



LD M1

LIMITD0 D10 D100 D1000

Performing the limit control of D0－D10 on the content of D100, and storing the obtained result in D1000 when M1=ON.In this case, D0(10)<=D100(30)<=D10(100), and D1000=30.

6.18.2 DBAND: Deadband control instruction

LAD: DBAND (S1) (S2) (S3) (D)	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: DBAND *(S1) (S2) (S3) (D)*	**Step length**	9

Operand	Type	Applicable soft element													Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S3	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

- Operand description

  S1: Lower limit value of the deadband

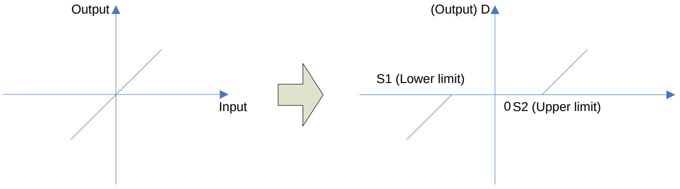
  S2: Upper limit value of the deadband

  S3:Input value to be controlled by the deadband

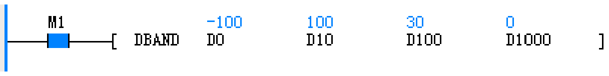
  D: Number of the start soft element that stores the output value that has been controlled by the deadband

- Function description

  You can judge whether the input value specified in S3 is within the range of the deadband specified by S1 and S2 so as to control and save the value in D. In this case, when S3 < S1, D = S3-S1; when S3 > S2, D = S3-S2; When S1<=S3<=S2, D=0.



- Application instance



LD M1

DBAND D0 D10 D100 D1000

Performing the deadband control of D0－D10 on the content of D100, and storing the obtained result in D1000 when M1=ON.In this case, D0 (-100)<D100(30)<D10(100), and D1000=0.

6.18.3 ZONE: Zone control instruction

LAD: ZONE (S1) (S2) (S3) (D)	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: ZONE *(S1) (S2) (S3) (D)*	**Step length**	9

Operand	Type	Applicable soft element	Indexing

S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S3	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

- Operand description

  S1: Negative deviation value that is added to the input value

  S2: Positive deviation value that is added to the input value

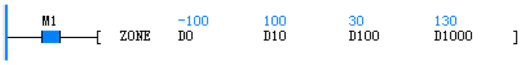
  S3: Input value to controlled by the zone control

  D: Number of the start soft element that stores the output value that hasbeen controlled byzone

- Function description

  You can judge the input value specified in S3 plus the deviation value specified by S1 or S2 so as to control and save the values in D. In this case, when S3<0, D=S3+S1; when S3>0, D=S3+S2; when S3 =0, D=0.

- Application instance

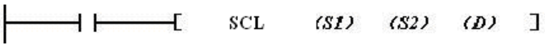


  LD M1

  ZONE D0 D10 D100 D1000

  Performing the zone control of D0 - D10 on the content of D100, and storing the obtained result in D1000 when M1=ON.In this case, D100(30)>0,D1000=D100(30)+D10(100), and D1000=130.

6.18.4 SCL: Coordinatesetting instruction

LAD: 	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: SCL *(S1) (S2) (S3) (D)*	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S2	INT								D				V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

- Operand description

  S1: Input value on which coordinates are set, or number of the soft element that stores the inputted value.

  S2: Number of the start table conversion soft element that is used to set the coordinates.

  D: Number of the soft element that stores the output value on which coordinates have been set.

- Function description

  1. Setting the coordinates on the input values specified by S1 according to the specified conversion characteristics, and storing the obtained results in the soft element numbers specified by D.

  2. The conversion for setting the coordinates is performed based on the data table stored in soft elements, of which the start soft element is specified by S2. However, when the outputted data is not an integer value, the data is outputted after rounding off the first decimal place.

  3. The coordinates are set through the conversion table.

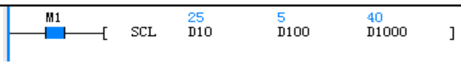
Points of the coordinates		S2
Point 1	X coordinate	S2+1
	Y coordinate	S2+2
Point 2	X coordinate	S2+3
	Y coordinate	S2+4
…	…	…
Point n (End)	X coordinate	S2+2n-1
	Y coordinate	S2+2n

- Note

  1. The X data of the data table needs to be arranged in ascending order. If only part of the data is not arranged in ascending order and the detection is started from button, and the operation before this part is still executed;

  2. S1 must be within the range set by the data table;

- Application instance

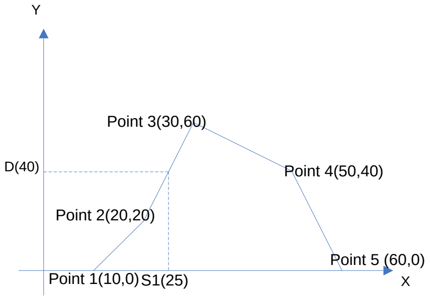


LD M1

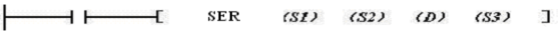
SCL D10 D100 D1000

Setting the coordinates on the content of D10, and storing the obtained results in D1000 when M1=ON.

Points of the coordinates		D100	5
Point 1	X coordinate	D101	10
	Y coordinate	D102	0
Point 2	X coordinate	D103	20
	Y coordinate	D104	20
Point 3	X coordinate	D105	30
	Y coordinate	D106	60
Point 4	X coordinate	D107	50
	Y coordinate	D108	40
Point 5	X coordinate	D109	60
	Y coordinate	D110	0



6.18.5 SER: Data search instruction

LAD: 	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: SER *(S1) (S2) (S3) (D)*	**Step length**	**9**

Operand	Type	Applicable soft element												Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√
S3	INT	Constant							D				V	R	

- Operand description

  S1: Number of the start soft element to be searched.

  S2: Reference value or number of the soft element that stores the target value for search.

  D: Number of the start soft element that stores the number of the pieces of same data, the position of the maximum value, and the position of the minimum value.

  S3: Number of units of data to be searched (1 ≤ S3 ≤ 256).

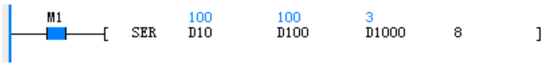
- Function description

  1. Searching S3 data starting from S1, searching the same data as S2, and storing the obtained results in D-D+4.

  2. When the same data exists, the number of the same data, and positions of the initial value/final value, min. and max. value are stored in 5 soft elements starting from D.

  3. When the same data does not exist, the first 3 soft elements store 0, and the other two are the same as above.

- Application instance



LD M1

SER D0 D10 D100 D1000 8

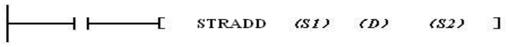
Searchingthe contents of 8 unit starting from D10, and storing the search results in5 units starting from D1000 when M1=ON.

The element value to be searched S1	value	The element value to be compared S2	Data position	Search result D	Value
D10	100	100	0	D1000	3
D11	78		1	D1001	0
D12	92		2	D1002	7
D13	100		3	D1003	5

D14	110		4	D1004	6
D15	-20		5		
D16	145		6		
D17	100		7		

6.19 Character string instructions

6.19.1 STRADD: String combination instruction

<table>
<tr><td colspan="8" rowspan="2">LAD:

 </td><td colspan="2">Applicable model</td><td colspan="6">VC3 VC5</td></tr>
<tr><td colspan="2">Influenced flag bit</td><td colspan="6">Zero flag, carry flag, and borrow flag</td></tr>
<tr><td colspan="8">IL: STRADD <i>(S1)</i> <i>(S2) (D)</i></td><td colspan="2">Step length</td><td colspan="6">7</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S1</td><td>INT</td><td>String</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>R</td><td>√</td></tr>
<tr><td>S2</td><td>INT</td><td>String</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>INT</td><td></td><td></td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td></td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>R</td><td>√</td></tr>
</table>

- Operand description

  S1: First string unit

  S2: Second string unit

  D: String storage unit after connection

- Function description

  1. Connecting the string units starting from S1 and S2, and storing them in the soft elements starting from D when the energy flow is valid;

  2. The combination of strings indicates that the first character of the S2 stringis connected to the end character of the S1 string, and the end identifier of the S1string is ignored.

  3. The valid data of a string unit indicates the data starting from the specified soft element for the string unit to the first data in which "00H" is detected.

  4. When the number of connected characters is an odd number, '00H' is added to the high byte of the last character soft element. When it is even, '0000H' is added to the next element of the last character soft element.

- Note

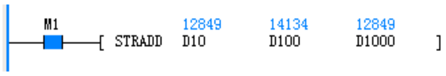
  1. When S1 and S2 specify a character string, a maximum of 32 characters are allowed. The comma and the double quotation mark represent the separators in the host computer software, so the character cannot be recognized by the host computer software;

  2. If '00H' is stored in both S1 and S2, then '0000H' can be added directly to D;

  3. When the soft element addresses of the string unit of S1 and D or S2 and D overlaps, the system reports the "The value of the Instruction operand is illegal" error.

  4. When '00H' does not appear in the corresponding soft element range of the string unit starting from S1 or S2, the system reports the "The element number of the instruction operand is out of range" error.

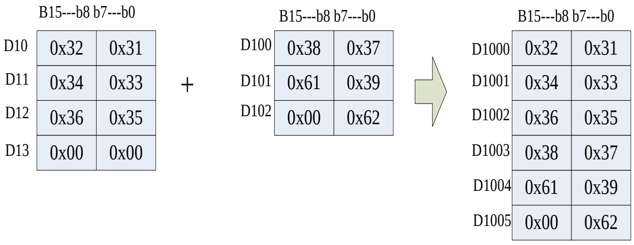
- Application instance



LD M1

STRADD D10 D100 D1000

Connecting the string unit starting from D10 and the string unit starting from D100, and storing the obtained result in the unit starting from D1000 when M1=ON.



	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x36	0x35
D13	0x00	0x00

	B15---b8	b7---b0
D100	0x38	0x37
D101	0x61	0x39
D102	0x00	0x62

	B15---b8	b7---b0
D1000	0x32	0x31
D1001	0x34	0x33
D1002	0x36	0x35
D1003	0x38	0x37
D1004	0x61	0x39
D1005	0x00	0x62

6.19.2 STRLEN: Instruction for detecting the string length

LAD:	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: STRLEN *(S) (D)*	**Step length**	**5**

Operand	Type	Applicable soft element													Indexing
S	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

- Operand description

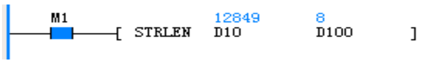
  S: The string unit

  D:The length of the string unit
- Function description

  1. Detecting the length of the S string, and storing the obtained value in D when the energy flow is valid.

  2. The valid data of a string unit indicates the data starting from the specified soft element for the string unit to the first data in which "00H" is detected.
- Note

  When '00H' does not appear in the corresponding soft element range of the string unit starting from S, the system reports the "The element number of the instruction operand is out of range" error.
- Application instance



LD M1

STRLEN D10 D100

Detecting the length of the string unit starting from D10, and storing the obtained result in D100 when M1=ON.

6.19.3 STRRIGHT: Instruction for reading a string from right

LAD: STRRIGHT (S1) (D) (S2)	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: STRRIGHT *(S1) (D) (S2)*	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√
S2	INT	Constant							D				V	R	

- Operand description

  S1: The string unit

  D: Storing the extracted string unit

  S2: Number of the extracted characters
- Function description

  1. Extracting S2 characters starting from the last valid character of the S1 string (not counting '00H'), and storing them in the soft element starting from D when the energy flow is valid;

  2. Storing "00H" in D soft element when S2 is equal to zero;

  3. Adding '00H'to the high byte of the soft element that holds the last character when the number of extracted characters is an odd number. Adding '0000H' to the next element of the soft element that holds the last character when it is even.

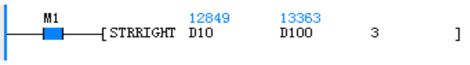
  4. The valid data of a string unit indicates the data starting from the specified soft element for the string unit to the first data in which "00H" is detected.
- Note

  1. When '00H' does not appear in the corresponding soft element range of the string unit starting from S1, the system reports the "The element number of the instruction operand is out of range" error.

  2. S2 is greater than or equal to 0.

3. S2 needs to be less than or equal to the character number of the S1 string.

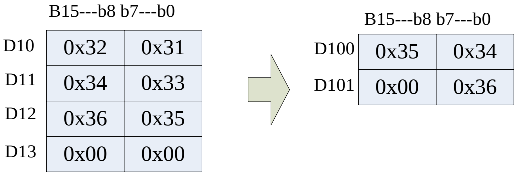
- Application instance



LD M1

STRRIGHT D10 D100 3

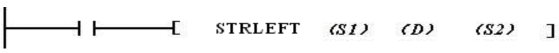
Extracting 3 characters from the right side of the string unit starting from D10, and storing them in the unit starting from D100 when M1=ON.



	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x36	0x35
D13	0x00	0x00

	B15---b8	b7---b0
D100	0x35	0x34
D101	0x00	0x36

6.19.4 STRLEFT: Instruction for reading a string from left

LAD: 	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: STRLEFT *(S1) (D) (S2)*	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√
S2	INT	Constant							D				V	R	√

- Operand description

S1: The string unit

D: Storing the extracted string unit

S2: Number of the extracted characters

- Function description

1. Extracting S2 characters rightward starting from the left side of the S1 string, and storing them in the soft element starting from D when the energy flow is valid;

2. Storing "00H" in D soft element when S2 is equal to zero;

3. Adding '00H'to the high byte of the soft element that holds the last character when the number of extracted characters is an odd number. Adding '0000H' to the next element of the soft element that holds the last character when it is even.

4. The valid data of a string unit indicates the data starting from the specified soft element for the string unit to the first data in which "00H" is detected.

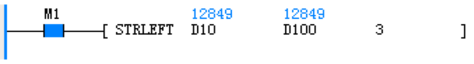
- Note

1. When '00H' does not appear in the corresponding soft element range of the string unit starting from S1, the system reports the "The element number of the instruction operand is out of range" error.

2. S2 is greater than or equal to 0.

3. S2 needs to be less than or equal to the character number of the S1 string.

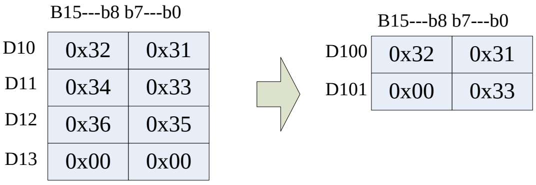
- Application instance



LD M1

STRLEFT D10 D100 3

Extracting 3 characters from the left side of the string unit starting from D10, and storing them in the unit starting from D100 when M1=ON.



	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x36	0x35
D13	0x00	0x00

	B15---b8	b7---b0
D100	0x32	0x31
D101	0x00	0x33

6.19.5 STRMIDR: Instruction for reading any characters of a string

LAD: STRMIDR (S1) (D) (S2)	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: STRMIDR *(S1) (D) (S2)*	**Step length**	**7**

Operand	Type	Applicable soft element												Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√
S2	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√

- Operand description

S1: The string unit

D:The extracted string unit

S2: Start position of the characters to be extracted

S2+1: Number of characters to be extracted (n)

- Function description

1. Extracting n character data of the S1 string starting from the S2 character, and storing them in the soft element starting from D when the energy flow is valid;

2. Adding '00H'to the high byte of the soft element that holds the last character when the number of extracted characters is an odd number. Adding '0000H' to the next element of the soft element that holds the last character when it is even.

3. The valid data of a string unit indicates the data starting from the specified soft element for the string unit to the first data in which "00H" is detected.

4. Performing no action when n is 0.

5. Extractingall character data of the S1 string, and storing them in the soft element starting from D when n is -1.

- Note

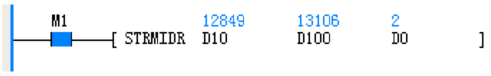
1. S2 needs to be less than or equal to the character number of the S1 string.

2. n is greater than 2.

3. S2 is greater than or equal to 1.

4. When '00H' does not appear in the corresponding soft element range of the string unit starting from S1, the system reports the "The element number of the instruction operand is out of range" error.

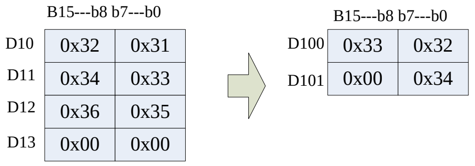
- Application instance



LD M1

STRMIDR D10 D100 D0

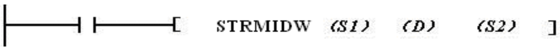
Reading D1 (D1=3) characters from D0 (D0=2) of the string unit starting from D10, and storing them in the unit starting from D100 when M1=ON.



	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x36	0x35
D13	0x00	0x00

	B15---b8	b7---b0
D100	0x33	0x32
D101	0x00	0x34

6.19.6 STRMIDW: Instruction for replacing any characters of a string

LAD:  	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: STRMIDW *(S1) (D) (S2)*	**Step length**	**7**

Operand	Type	Applicable soft element												Indexing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√
S2	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√

- Operand description

S1: The string unit to be used

D: The string unit to be replaced

S2: Start positionof the replacement

S2+1: Number of characters to be replaced (n)

- Function description

1. Replacing n character data starting from the S2 character of the D string with n characters data of S1 string when the energy flow is valid;

2. The valid data of a string unit indicates the data starting from the specified soft element for the string unit to the first data in which "00H" is detected.

3. Performing no action when n is 0.

4. When n is -1, the contents to the final character data specified by S1 is stored in the soft elements following the soft element specified by D.

- Note

1. S2 needs to be less than or equal to the character number of the S1 string.

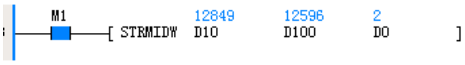
2. n is greater than 2.

3. S2 is greater than or equal to 1.

4. When the number of characters for replacement exceeds the last character of the string unit starting from D, the extra characters are not saved.

5. When '00H' does not appear in the corresponding soft element range of the string unit starting from S1 and D, the system reports the "The element number of the instruction operand is out of range" error.

- Application instance



LD M1

STRMIDW D10 D100 D0

Replacing D1 (D1=3) characters after the D0 (D0=2) characters of the string unit starting from D100 with the first D1 (D1=3) characters of the string unit starting from D10 when M1=ON.

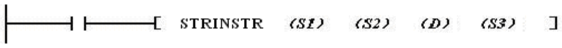
	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x00	0x35

	B15---b8	b7---b0
D100	0x35	0x34
D101	0x37	0x36
D102	0x39	0x38
D103	0x00	0x61

⇨

	B15---b8	b7---b0
D100	0x31	0x34
D101	0x33	0x32
D102	0x39	0x38
D103	0x00	0x61

6.19.7 STRINSTR: String search instruction

LAD: 	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: STRINSTR *(S1) (S2) (D) (S3)*	**Step length**	**9**

Operand	Type	Applicable soft element												Indexing
S1	INT	String						D	SD	C	T	V	R	√
S2	INT							D	SD	C	T	V	R	√
D	INT							D	SD	C	T	V	R	√
S3	INT	Constant						D				V	R	

- Operand description

S1: The string unit to be searched

S2: Search source

D: Search results

S3: Start positionof searching

- Function description

1. Searching the same string as the S1 string starting from the the S3 character of the S2 string, and storing the string position information of search results in D when the energy flow is valid.

2. Storing "0" in D when there is no consistent string.

3. No operation is performed when the position S3 where the search is started is a negative number or "0".

4. The valid data of a string unit indicates the data starting from the specified soft element for the string unit to the first data in which "00H" is detected.

- Note

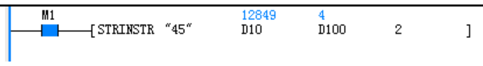
1. When '00H' does not appear in the corresponding soft element range of the string unit starting from S1 or S2, the system reports the "The element number of the instruction operand is out of range" error.

2. S3 is less than or equal to the number of characters in the S2 string.

3. When S1 specifies a character string, a maximum of 32 characters are allowed. The comma and the double quotation mark represent the separators in the host computer software, so the character cannot be recognized by the host computer software.

4. When S1 is an empty character string ('00H'), the position of the S2 character string '00H' is detected (if S2 is an even number of characters, it is the position of the first '00H').

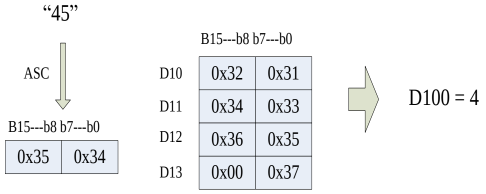
- Application instance


M1
STRINSTR "45"
12849
D10
4
D100
2
]

LD M1

STRINSTR "45" D10 D100 2

Searching the same string as the "45" starting fromthe second character of the character string unit starting from D10, and storing the search results in D100 when M1=ON.


"45"
ASC
B15---b8 b7---b0
0x35
0x34
B15---b8 b7---b0
D10
0x32
0x31
D11
0x34
0x33
D12
0x36
0x35
D13
0x00
0x37
D100 = 4

6.19.8 STRMOV: String transmission instruction

<table>
<tr><td colspan="8" rowspan="2">LAD:
STRMOV (S) (D)]</td><td colspan="2">Applicable model</td><td colspan="6">VC3 VC5</td></tr>
<tr><td colspan="2">Influenced flag bit</td><td colspan="6">Zero flag, carry flag, and borrow flag</td></tr>
<tr><td colspan="8">IL: STRMOV <i>(S)</i> <i>(D)</i></td><td colspan="2">Step length</td><td colspan="6">5</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S</td><td>INT</td><td>String</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>INT</td><td></td><td></td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td></td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>R</td><td>√</td></tr>
</table>

- Operand description

  S: Source string unit

  D: Destination unit

- Function description

  1. Transferring all data of the S string unit, including '00H', to the element unit starting from D.

  2. The valid data of a string unit indicates the data starting from the specified soft element for the string unit to the first data in which "00H" is detected.

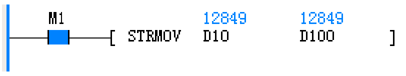
- Note

  1. When '00H' does not appear in the corresponding soft element range of the string unit starting from S, the system reports the "The element number of the instruction operand is out of range" error.

  2. When the number of characters of the S string unit is even, '00H' is stored in LSB, and both MSB and LSB of the corresponding position in D stores '00H'.

  3. When S1 specifies a character string, a maximum of 32 characters are allowed. The comma and the double quotation mark represent the separators in the host computer software, so the character cannot be recognized by the host computer software.

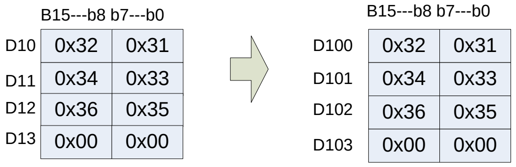
- Application instance


M1
STRMOV
12849
D10
12849
D100
]

LD M1

STRMOV D10 D100

Transferring the character string data starting from D10 to the unit starting from D100 when M1=ON.


B15---b8 b7---b0
D10
0x32
0x31
D11
0x34
0x33
D12
0x36
0x35
D13
0x00
0x00
B15---b8 b7---b0
D100
0x32
0x31
D101
0x34
0x33
D102
0x36
0x35
D103
0x00
0x00

6.20 Extension file register instructions

6.20.1 LOADR: Instruction for reading data from an extension file register

<table>
<tr><td colspan="8" rowspan="2">LAD:
LOADR (S1) (S2)]</td><td colspan="2">Applicable model</td><td colspan="5"></td></tr>
<tr><td colspan="2">Influenced flag bit</td><td colspan="5">Zero flag, carry flag, and borrow flag</td></tr>
<tr><td colspan="8">IL: LOADR <i>(S1)</i> <i>(S2)</i></td><td colspan="2">Step length</td><td colspan="5">5</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S1</td><td>INT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>R</td><td>√</td></tr>
</table>

S2	INT	Constant							D					

- Operand description

S1: Soft element unit of the extension register for storing data

S2: Number of data that has been read (1 ≤ S2 ≤ 1024)

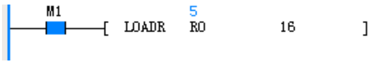
- Function description

Reading out S2 data from S1 of the extension file register stored in the storage box to the soft elements starting from S1 of the extension register in the PLC;

- Note

1. No operation is executed when S2 is specified as 0.
2. When the storage box is not connected, if you execute this instruction,the system reports that there is no memory card.
3. When S2=1024, the execution time of the instruction is about 80ms. You need to set the watchdog time correctly in the practical applications.

- Application instance



LD M1

LOADR R0 16

Reading 16 data starting from R0 of the memory card R backup area, and storing these data in the 16 soft element units starting from R0 when M1=ON.

6.20.2 SAVER: Instruction for writing data to an extension file register

LAD: SAVER (S1) (S2) (D)	**Applicable model**	
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: SAVER (S1) (S2) (D)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT												R		√
S2	INT	Constant													
D	INT								D						√

- Operand description

S1: Soft element unit of the extension register for storing data(only the start address of soft elements in the segment can be specified)

S2: Number of data that has been written per operation cycle (1 ≤ S2 ≤ 2048)

D: Storing number of data that has been written

- Function description

1. Writing data from S1 - S1+2047 in the extension register to the same unit in the storage box by 2048/S2 (if 20148 is not divisible by S2, the quotient is increased by 1) operation cycle.
2. Storing number of data that has been written in D during the writing process.
3. Setting the instruction execution end flag SM189 when the execution of the instruction execution ends.

- Note

1. When S2=2048, the execution time of the instruction is about several hundredms. You need to set the watchdog time correctly.
2. You need to execute the INITER or INITR instruction to initialize the operated segments before driving the SAVER instruction. If the data written to the extension file register do not match those of the extension register, the system reports that the operation on the memory card is incorrect.
3. When a value of 0 is specified in S2, it is executed according to S2 = 2048.

4. Data in S1 need to be number of the start soft elementin the extension register segment. Number of the start soft elementin each segment is as follows.

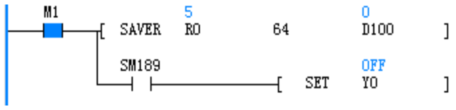
Segment number	Number of the start soft element	Range of writing to the memory card R backup area	Segment number	Number of the start soft element	Range of writing to the memory card R backup area

0	R0	R0–R2047	8	R16384	R16384–R18431
1	R2048	R2048–R4095	9	R18432	R18432–R20479
2	R4096	R4096–R6143	10	R20480	R20480–R22527
3	R6144	R6144–R8191	11	R22528	R22528–R24575
4	R8192	R8192–R10239	12	R24576	R24576–R26623
5	R10240	R10240–R12287	13	R26624	R26624–R28671
6	R12288	R12288–R14335	14	R28672	R28672–R30719
7	R14336	R14336–R16383	15	R30720	R30720–R32767

5. When no memory is connected, the system reports that there is no memory card.

6. When hardware writing protection is performed for the memory card, the system reports a memory card operation error.

- Application instance



LD M1

SAVER R0 64 D100

LD SM189

SET Y0

Storing 2048 soft element data starting from R0 in 2048 units starting from ER0, and storing the number of units that are already stored in D100 when M1=ON. When the execution of the instruction is completed, SM189=ON, and Y0=ON.

6.20.3 INITR: Instruction for initializing an extension register

LAD: INITR (S1) (S2)	**Applicable model**	
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: INITR *(S1) (S2)*	**Step length**	**5**

Operand	Type	Applicable soft element													Indexing
S1	INT													R	√
S2	INT	Constant													

- Operand description

S1: Units of the extension register and extension file register to be initialized (only the address of the start soft element in the segment can be specified)

S2: Segments of the extension register and extension file register to be initialized (S2=1)

- Function description

1. Initializing S2 extension registers and extension file registers starting from S1, and writing the initial value 0xFFFF.

2. The initialization is executed by segments.

- Note

1. S1 needs to specify number of the start soft element in the segment. Number of the start soft element in each segment is as follows.

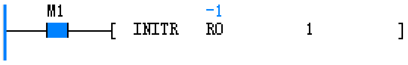
Segment number	Number of the start soft element	Range of writing to the memory card R backup area	Segment number	Number of the start soft element	Range of writing to the memory card R backup area
0	R0	R0–R2047	8	R16384	R16384–R18431
1	R2048	R2048–R4095	9	R18432	R18432–R20479
2	R4096	R4096–R6143	10	R20480	R20480–R22527
3	R6144	R6144–R8191	11	R22528	R22528–R24575
4	R8192	R8192–R10239	12	R24576	R24576–R26623
5	R10240	R10240–R12287	13	R26624	R26624–R28671
6	R12288	R12288–R14335	14	R28672	R28672–R30719
7	R14336	R14336–R16383	15	R30720	R30720–R32767

2. When the memory card is not used, the initialization of the extension file register is not performed.

3. If the memory card is connected,and hardware writing protection is performed for the memory card, the system reports a memory card operation error.

4. The instruction can only initialize one segment for each execution. When the memory card is used, the operation time of initializing each segment is about 100ms. In actual application, You need to set the watchdog time correctly.

- Application instance

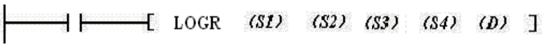


LD M1

INITR R0 1

Initializing the extension registers R – R2047 in segment 0 when M1=ON. In the case of using a memory card, the extension file registers ER – ER2047 are also initialized.

6.20.4 LOGR: Instruction for logging on an extension register

LAD: 	**Applicable model**	
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: LOGR *(S1) (S2) (S3) (S4) (D)*	**Step length**	**11**

Operand	Type	Applicable soft element													Indexing
S1	INT								D		C	T			√
S2	INT	Constant							D						
S3	INT													R	
S4	INT	Constant													
D	INT								D						√

- Operand description

S1: Object unit that performs the login

S2: Number ofloogined object units (1 – 1024)

S3: Address of the start soft element unit used in login

S4: Number of soft element segments used in login (1 – 16)

D: Number of logined data

- Function description

1. Continuously logining in S2 soft elements starting from S1 until S4 segments of the extension register and the extension file register starting starting from S3 are filled in when the energy flow is valid.

2. Performing login in each execution cycle.

3. Storing the number of logined data in D.

- Note

1. When using the memory card, you need to initialize the segments to be logged. If the data logged on the extension file register is inconsistent with the data starting from S1, the system reports that the operation on the memory card is incorrect. In this case, you can initialize the segments to be logged one by one through the INITR or INITER command, or initialize all ER elements through the menu clear instruction of the background memory card (you need to select the user program, global variables, data block, and system block simultaneously). To avoid data loss, you need to backup the memory card file content through the background software before initialization.

2. S3 needs to specify number of the start soft element in the segment. Number of the start soft element in each segment is as follows.

Segment number	Number of the start soft element	Range of writing to the memory card R backup area	Segment number	Number of the start soft element	Range of writing to the memory card R backup area
0	R0	R0–R2047	8	R16384	R16384–R18431
1	R2048	R2048–R4095	9	R18432	R18432–R20479
2	R4096	R4096–R6143	10	R20480	R20480–R22527
3	R6144	R6144–R8191	11	R22528	R22528–R24575
4	R8192	R8192–R10239	12	R24576	R24576–R26623
5	R10240	R10240–R12287	13	R26624	R26624–R28671
6	R12288	R12288–R14335	14	R28672	R28672–R30719

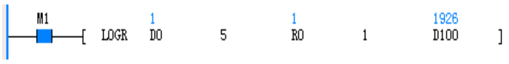
7	R14336	R14336–R16383	15	R30720	R30720–R32767

3. If there is memory hardware protection, the system reports that the operation on the memory card is incorrect.

4. Login data format

S3–S3+S2-1	The storage address of the first logined data S3 – S3+S2-1	D=S2	Data write area 1926×S4	Login data storage area
S3+S2–S3+2S2-1	The storage address of the second logined data S3 – S3+S2-1	D=2S2		
S3+2S2–S3+3S2-1		D=3S2		
…………	…………	……		
S3+1926×S4-1 –S3+2048×S4-1	Writing intothe location management area When using the data write area of one word, the sequence is changed from ON to OFF from 0 bits of S3+1926×S4-1, and the next soft element S3+1926×S4 is used when S3+1926×S4-1 is all OFF.		Management area 122 × S4	

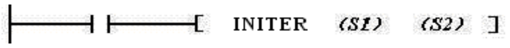
- Application instance



```
LD  M1
LOGR    D0  5   R0  1   D100
```

Logging the data of the D0–D4 unit in the R0–R2047 unit of the segment 0, and recording the number of logined data in D100 when M1=ON.

6.20.5 INITER: Instruction for initializing an extension file register

<table>
<tr><td colspan="7" rowspan="2">LAD:
 </td><td colspan="2">Applicable model</td><td colspan="7"></td></tr>
<tr><td colspan="2">Influenced flag bit</td><td colspan="7">Zero flag, carry flag, and borrow flag</td></tr>
<tr><td colspan="7">IL: INITER <i>(S1)</i> <i>(S2)</i></td><td colspan="2">Step length</td><td colspan="7">5</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>S1</td><td>INT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>R</td><td>√</td></tr>
<tr><td>S2</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

- Operand description

S1: The same extension register unit that share the same address with the extension file register unit to be initialized(only the start address of extension register in the segment can be specified)

S2: Segments of the extension register and extension file register to be initialized (S2=1)

- Function description

1. Initializing S2 segments of extension file registers starting from S1 in the storage box, and writing the value 0xFFFF into it.

2. The initialization is executed by segments.

- Note

1. S1 needs to specify number of the start soft element in the segment. Number of the start soft element in each segment is as follows.

Segment number	Number of the start soft element	Range of writing to the memory card R backup area	Segment number	Number of the start soft element	Range of writing to the memory card R backup area
0	R0	R0–R2047	8	R16384	R16384–R18431
1	R2048	R2048–R4095	9	R18432	R18432–R20479

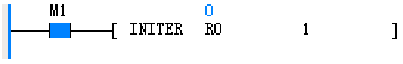
2	R4096	R4096–R6143	10	R20480	R20480–R22527
3	R6144	R6144–R8191	11	R22528	R22528–R24575
4	R8192	R8192–R10239	12	R24576	R24576–R26623
5	R10240	R10240–R12287	13	R26624	R26624–R28671
6	R12288	R12288–R14335	14	R28672	R28672–R30719
7	R14336	R14336–R16383	15	R30720	R30720–R32767

2. If there is memory hardware protection, the system reports that the operation on the memory card is incorrect.

3. When no storage box is connected, the system reports that there is no memory card.

4. The instruction can only initialize one segment for each execution, and the operation time of initializing each segment is about 100 ms. In actual application, You need to set the watchdog time correctly.

- Application instance



LD M1

INITER R0 1

Initializing the extension file registers ER0–ER2047 in segment 0 when M1=ON.

6.21 Positioning instructions

6.21.1 ZRN: Zero return instruction

LAD: ZRN (S1) (S2) (S3) (D)	**Applicable model**	VC1S VC1 VC2 VC3	
	Influenced flag bit		
IL: ZRN (*S1*) (*S2*) (*S3*) (*D*)	**Step length**	**11**	

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S3	BOOL		X	Y	M	S										
D	BOOL			Y												

- Operand description

  S1: Zero return speed, specifying the speed at which the zero return starts.

  32-bit instruction.

  S2: Crawling speed, the lower speed after the specified near-point dog signal (DOG) is turned ON.

  S3: DOG to be inputted in X element.

  When an element other than the input relay (X) is specified, the position offset of the zero point increases due to the influence of the PLC calculation cycle.

  D: Start address of the high-speed pulse output.

- Function description

  Taking Y0 as an example. When SM276 clearing signal is valid, and elements specified by the SM281 clearing signal are invalid, Y10 is the output port of the clearing signal. When elements specified by the SM281 clearing signal are invalid, Y (N) specified by SD175 is the output port of the clearing signal.

- Note

  1. Because the ZRN instruction is incapable of searching for DOG automatically, it is required to start the zero return operation further than the front end of DOG detection device.

- Time sequence diagram

2. During the zero return process, the value of the current value register decreases.

3. The min. output pulse frequency that can be outputted actually, is determined by the formula below:

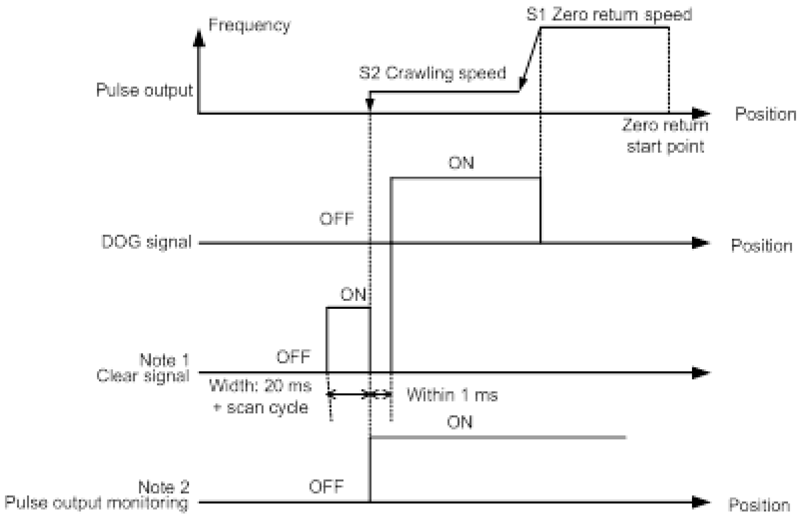
$$F_{\min\_acc} = \sqrt{\frac{F_{\max} \times 500}{T}}$$

In the above formula, $F_{\max}$ indicates the max. speed while T indicates the acceleration/deceleration time withthe unit of ms. The calculation result $F_{\min\_acc}$ is the limit value of min. output frequency.

4. For the pulse output frequency, the calculated frequency is output even if a value less than the calculated one is assigned. The frequency of the starting section of ACC and end part of DEC cannot be less than the above calculation result. If the max. speed is lower than the above calculation result, there is not pulse output.

5. Crawling speed needs to be larger than zero and less than one tenth of the max. speed.

6. For details, refer to Chapter 11 "User Guide for Positioning Function".


Frequency
Pulse output
S1 Zero return speed
S2 Crawling speed
Position
Zero return start point
ON
OFF
DOG signal
Position
ON
Note 1 Clear signal
OFF
Width: 20 ms + scan cycle
Within 1 ms
ON
Note 2 Pulse output monitoring
OFF
Position

Note1:Enabing the clear function when SM276 is set.

Note2:SM271 and SM291 are the pulses output monitors of Y0 and Y1.

6.21.2 PLSV:Variable speed pulse output instruction

LAD: ├─┤ ├──[PLSV *(S)* *(D1)* *(D2)*]	**Applicable model**	VC1S VC1 VC2 VC3
	Influenced flag bit	
IL: PLSV (*S*) (*D1*) (*D2*)	**Step length**	**8**

Operand	Type	Applicable soft element														Indexing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D1	BOOL			Y												
D2	BOOL			Y	M	S										

- Operand description

S: Output pulse frequency (Hz)

32-bit instruction.

D1: Start address of the high-speed pulse output.

D2: Start address of the rotating direction signal output. Corresponding to the positive/negative condition of ***S***, it acts as follows:

- When S is positive: D2 is ON.
- When S is negative: D2 is OFF.

- Function description

1. You can change the output pulse frequency (***S***) freely even in the high-speed pulse output state.

2. Because there is no acceleration or deceleration during the start/stop, if buffer is needed during the start or stop, it is recommended to use the RAMP instruction to change the value of the output pulse frequency (***S***).

3. During the process of the high-speed pulse output, when the energy flow driven by the instruction turns OFF, the output stops without deceleration.

4. If the high-speed pulse output monitoring is ON, the energy flow driven by the instruction is not driven by the instruction again after the energy flow turns OFF.

5. The direction is determined by the positive or negative nature of ***S***.

- Note

1. PLSY, PLS, and positioning instructions can output the high-speed pulses port. It is not allowed to use these instructions for high-speed pulse output on the same port at the same time.

2. For details, refer to Chapter 11 "User Guide for Positioning Function".

6.21.3 DRVI: Relative position control instruction

LAD: ├┤ ├─[DRVI (S1) (S2) (D1) (D2)]	**Applicable model**	VC1S VC1 VC2 VC3
	Influenced flag bit	
IL: DRVI (*S1*) (*S2*) (*D1*) (*D2*)	**Step length**	**11**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D1	BOOL			Y												
D2	BOOL			Y	M	S										

- Operand description

  S1: Number of output pulses (relatively specified)

  S2: Output pulse frequency (Hz).

  D1: Start address of the high-speed pulse output.

  D2: Start address of the rotating direction signal output. Corresponding to the positive/negative condition of ***S***, it actsas follows:

  • When S1 is positive: D2 is ON.

  • When S1 is negative: D2 is OFF.

- Function description

  1. ***S1*** corresponds to the following current value registers as a relative position.

SD162	Current position value in the positioning instructionoutputted through Y0 (MSB)
SD163	Current position value in the positioning instructionoutputted through Y0 (LSB)
SD182	Current position value in the positioning instructionoutputted through Y1 (MSB)
SD183	Current position value in the positioning instructionoutputted through Y1 (LSB)
SD202	Current position value in the positioning instructionoutputted through Y2 (MSB)
SD203	Current position value in the positioning instructionoutputted through Y2 (LSB)
SD222	Current position value in the positioning instructionoutputted through Y3 (MSB)
SD223	Current position value in the positioning instructionoutputted through Y3 (LSB)
SD242	Current position value in the positioning instructionoutputted through Y4 (MSB)
SD243	Current position value in the positioning instructionoutputted through Y4 (LSB)
SD262	Current position value in the positioning instructionoutputted through Y5 (MSB)
SD263	Current position value in the positioning instructionoutputted through Y5 (LSB)
SD282	Current position value in the positioning instructionoutputted through Y6 (MSB)
SD283	Current position value in the positioning instructionoutputted through Y6 (LSB)
SD302	Current position value in the positioning instructionoutputted through Y7 (MSB)
SD303	Current position value in the positioning instructionoutputted through Y7 (LSB)

  2. During the reverse rotation, the value of current value register decreases.

  3. The rotating direction is determined by the positive or negative nature of ***S1***.

  4. During the execution of the instruction, even if the contents of the operands are changed, these changes cannot be displayed in the current operation and can only take effect at the next execution of instruction.

  5. During the execution of the instruction, the output decelerates to stop when the energy flow driven by the instruction turns OFF. The execution completion flag SM does not act at the moment.

  6. If the high-speed pulse output flag is ON, the energy flow driven by the instruction is not driven by the instruction again after the energy flow turns OFF.

- Note

  1. The min. output pulse frequency that can be outputted actually, is determined by the formula below:

$$F_{\min\_acc} = \sqrt{\frac{F_{\max} \times 500}{T}}$$

  In the above formula, $F_{\max}$ indicates the max. speed while T indicates the acceleration/deceleration time withthe unit of ms. The calculation result $F_{\min\_acc}$ is the limit value of min. output frequency.

  2. For the pulse output frequency, the calculated frequency is output even if a value less than the calculated one is assigned. The frequency of the starting section of ACC and end part of DEC cannot be less than the above calculation result. If the max. speed is lower than the above calculation result, there is not pulse output.

  3. Crawling speed needs to be larger than zero and less than one tenth of the max. speed.

  4. For details, refer to Chapter 11 "User Guide for Positioning Function".

6.21.4 DRVA: Absolute position control instruction

LAD: |—| |—[DRVA (S1) (S2) (D1) (D2)]	**Applicable model**	VC1S VC1 VC2 VC3
	Influenced flag bit	
IL: DRVA (*S1*) (*S2*) (*D1*) (*D2*)	**Step length**	**11**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D1	BOOL			Y												
D2	BOOL			Y	M	S										

- Operand description

  S1: Target position (absolutelyspecified)

  S2: Output pulse frequency (Hz)

  32-bit instruction.

  D1: Start address of the high-speed pulse output. For VC1S, only Y0 or Y1 can be designated.

  The PLC output need to adopt transistor output mode.

  D2: Start address of the rotating direction signal output.

  If the pulse is transmitted in forward direction, ***D2*** is ON, otherwise ***D2*** is OFF.

- Function description

  1. ***S1*** corresponds to the following current value registers as a relative position.

SD162	Current position value in the positioning instructionoutputted through Y0 (MSB)
SD163	Current position value in the positioning instructionoutputted through Y0 (LSB)
SD182	Current position value in the positioning instructionoutputted through Y1 (MSB)
SD183	Current position value in the positioning instructionoutputted through Y1 (LSB)
SD202	Current position value in the positioning instructionoutputted through Y2 (MSB)
SD203	Current position value in the positioning instructionoutputted through Y2 (LSB)
SD222	Current position value in the positioning instructionoutputted through Y3 (MSB)
SD223	Current position value in the positioning instructionoutputted through Y3 (LSB)
SD242	Current position value in the positioning instructionoutputted through Y4 (MSB)
SD243	Current position value in the positioning instructionoutputted through Y4 (LSB)
SD262	Current position value in the positioning instructionoutputted through Y5 (MSB)
SD263	Current position value in the positioning instructionoutputted through Y5 (LSB)
SD282	Current position value in the positioning instructionoutputted through Y6 (MSB)
SD283	Current position value in the positioning instructionoutputted through Y6 (LSB)
SD302	Current position value in the positioning instructionoutputted through Y7 (MSB)
SD303	Current position value in the positioning instructionoutputted through Y7 (LSB)

  2. During the reverse rotation, the value of current value register decreases.

  3. The rotating direction is determined by the positive or negative nature of ***S1***.

  4. During the execution of the instruction, even if the contents of the operands are changed, these changes cannot be displayed in the current operation and can only take effect at the next execution of instruction.

  5. During the execution of the instruction, the output decelerates to stop when the energy flow driven by the instruction turns OFF. The execution completion flag SM does not actat the moment.

  6. If the high-speed pulse output flag is ON, the energy flow driven by the instruction is not driven by the instruction again after the energy flow turns OFF.

- Note

  1. The min. output pulse frequency that can be outputted actually, is determined by the formula below:

$$F_{\min\_acc}=\sqrt{\frac{F_{\max}\times 500}{T}}$$

  In the above formula, $F_{\max}$ indicates the max. speed while T indicates the acceleration/deceleration time with the unit of ms. The calculation result $F_{\min\_acc}$ is the limit value of min. output frequency.

  2. For the pulse output frequency, the calculated frequency is output even if a value less than the calculated one is assigned. The frequency of the starting section of ACC and end part of DEC cannot be less than the above calculation result. If the max. speed is lower than the above calculation result, there is not pulse output.

  3. Crawling speed needs to be larger than zero and less than one tenth of the max. speed.

  4. For details, refer to Chapter 11 "User Guide for Positioning Function".

6.21.5 DSZR: Instruction for zero return with DOG

LAD: ├──┤ ├──[DSZR *(S1)* *(S2)* *(D1)* *(D2)*]	**Applicable model**	VC1 VC2 VC3	
	Influenced flag bit		
IL: DSZR *(S1)* *(S2)* *(D1)* *(D2)*	**Step length**	**9**	

Operand	Type	Applicable soft element													Indexing
S1	BOOL		X	Y	M	S									
S2	BOOL		X												
D1	BOOL			Y											
D2	BOOL			Y	M	S									

- Operand description

  S1: Specifying the soft element number of DOG. When an input soft element is designated, the position offset of the zero point increases due to the influence of the PLC calculation cycle.

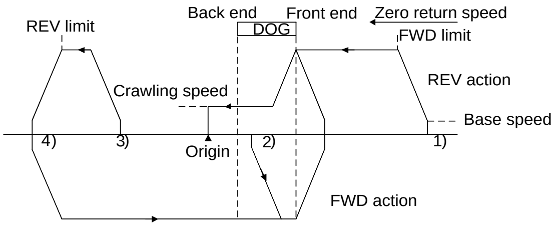
  S2: Specifying the soft element number of the input zero signal. Range: X0 – X7.

  D1: Specifying the pulse number of the output pulses.

  D2: Specifying the output object number of the rotating direction signal.

- Function description

  The ZRN instruction that permits the use of DOG signal and zero signal. There is FWD and REV limits. The ZRN action depends on the ZRN position. A clearing signal is sent after the instruction ends.



1. When the starting position is before passing DOG:

   1)Starting the ZRN action through executing the ZRN instruction.

   2) Moving toward the ZRN direction atzero return speed.

   3) Starting to decelerate to the crawling speed once the front end of DOG is detected.

   4) Stopping once the first zero signal is detected again after detecting the back end of DOG.

2. When the starting position is within DOG:

   1)Starting the ZRN action through executing the ZRN instruction.

   2) Moving toward the opposite direction ofthe ZRN direction atzero return speed.

   3) Decelerating to stop afterdetecting the front end of DOG.(leaving DOG)

   4) Moving toward the ZRN direction at zero return speed. (entering DOG again)

   5) Starting to decelerate to the crawling speed once the front end of DOG is detected.

   6) Stopping once the first zero signal is detected after detecting the back end of DOG.

3. When the starting position is in DOG OFF (after passing DOG):

   1)Starting the ZRN action through executing the ZRN instruction.

   2) Moving toward the ZRN direction atzero return speed.

   3)Decelerating to stop after detecting the reverse limit.

   4) Moving toward the opposite direction ofthe ZRN direction at zero return speed.

   5) Decelerating to stop after detecting the front end of DOG.(detecting (leave) DOG)

   6) Moving toward the ZRN direction at zero return speed.

   7) Starting to decelerate to the crawling speed once the front end of DOG is detected.

   8) Stopping once the first zero signal is detected after detecting the back end of DOG.

4. When the starting position is in limit switch position (FWD or REV limit):

   1)Starting the ZRN action through executing the ZRN instruction.

   2) Moving toward the opposite direction ofthe ZRN direction at zero return speed.

   3) Decelerating to stop after detecting the front end of DOG.(detecting (leave) DOG)

4) Moving toward the ZRN direction at zero return speed. (entering DOG again)

5) Starting to decelerate to the crawling speed once the front end of DOG is detected.

6) Stopping once the first zero signal is detected after detecting the back end of DOG.

- Note

1. PLSY, PLS, and positioning instructions can output the high-speed pulses port. It is not allowed to use these instructions for high-speed pulse output on the same port at the same time.

2. The min. output pulse frequency that can be outputted actually, is determined by the formula below:

$$F_{\min\_acc}=\sqrt{\frac{F_{\max}\times 500}{T}}$$

In the above formula, $F_{\max}$ indicates the max. speed, while T indicates the acceleration/deceleration time,, withthe unit of ms. The calculation result $F_{\min\_acc}$ is the limit value of min. output frequency.

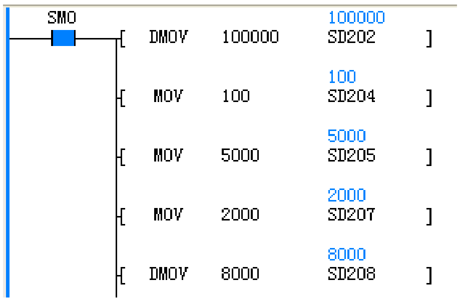
3. For the pulse output frequency, the calculated frequency is output even if a value less than the calculated one is assigned. The frequency of the starting section of ACC and end part of DEC cannot be less than the above calculation result. If the max. speed is lower than the above calculation result, there is not pulse output.

4. Crawling speed needs to be larger than zero and less than one tenth of the max. speed.

5. For details, refer to Chapter 11 "User Guide for Positioning Function".

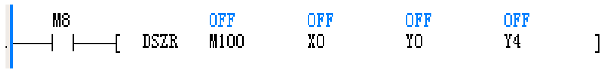
Application instance

Parameters such as the max. speed, base speed, acceleration/deceleration time, zero return speed, and crawling speed can adopt default values or can be reset through the soft element assignment.



```
SM0                       100000
─┤├──┬─[ DMOV  100000   SD202   ]
     │                  100
     ├─[ MOV   100      SD204   ]
     │                  5000
     ├─[ MOV   5000     SD205   ]
     │                  2000
     ├─[ MOV   2000     SD207   ]
     │                  8000
     └─[ DMOV  8000     SD208   ]
```

Executingthe DSZR instruction:



```
M8                OFF    OFF   OFF   OFF
─┤├──┤├──[ DSZR   M100   X0    Y0    Y4    ]
```

6.21.6 DVIT: Interrupt positioning instruction

LAD: ├─┤ ├──[DVIT *(S1)* *(S2)* *(D1)* *(D2)*]	**Applicable model**	**VC1**
	Influenced flag bit	
IL: DVIT *(S1)* *(S2)* *(D1)* *(D2)*	**Step length**	**11**

Operand	Type	Applicable soft element														Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D1	BOOL			Y												
D2	BOOL			Y	M	S										

- Operand description

S1: Specifying the number of output pulses after the interrupt (relative address).

S2: Specifying the output pulse frequency.

D1:Specifying the output number of the output pulses.

D2: Specifying the output object number of the rotation direction signal.

- Function description

1. When the interrupt is generated, the pulse is output at the specified frequency and the specified number of pulses.SM284 is valid for interrupt input function; SD176 is the designation of interrupt input function; SM283 is the logic inversion soft element of the interrupt input signal. The logic inversion determines whether the interrupt soft element generates an interrupt by ON or OFF.

2. The designation method of the interrupt input is as below: SM284 is set to ON.

3. Designating a input number (X0–X7) as the interrupt input in SD176, or designating the user interrupt

instruction soft elements, in which the LSB of SD176 correspond to the interrupt input used by the pulse output Y0, and the MSB corresponds to the interrupt input used by the pulse output Y1.

<table>
<tr><th>Set value</th><th>Setting content</th></tr>
<tr><td>0</td><td>Designating X0 as the interrupt input signal</td></tr>
<tr><td>1</td><td>Designating X1 as the interrupt input signal</td></tr>
<tr><td>……</td><td>……</td></tr>
<tr><td>7</td><td>Designating X7 as the interrupt input signal</td></tr>
<tr><td>8×1</td><td>Designating the user interrupt instruction soft element×1 as the interrupt input signal
<table>
<tr><th>Pulse output soft element</th><th>User interrupt instruction soft element</th></tr>
<tr><td>Y0</td><td>SM285</td></tr>
<tr><td>Y1</td><td>SM305</td></tr>
<tr><td>Y2</td><td>SM325</td></tr>
</table></td></tr>
</table>

- Note

1. PLSY, PLS, and positioning instructions can output the high-speed pulses port. It is not allowed to use these instructions for high-speed pulse output on the same port at the same time.

2. The min. output pulse frequency that can be outputted actually, is determined by the formula below:

$$F_{\min\_acc} = \sqrt{\frac{F_{\max} \times 500}{T}}$$

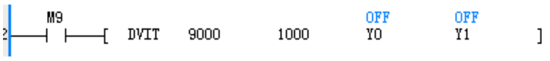
In the above formula, $F_{\max}$ indicates the max. speed, while T indicates the acceleration/deceleration time, withthe unit of ms. The calculation result $F_{\min\_acc}$ is the limit value of min. output frequency.

3. For the pulse output frequency, the calculated frequency is output even if a value less than the calculated one is assigned. The frequency of the starting section of ACC and end part of DEC cannot be less than the above calculation result. If the max. speed is lower than the above calculation result, there is not pulse output.

4. When the number of output pulses is less than the number of pulses required for deceleration, the frequency can be decelerated.

5. For details, refer to Chapter 11 "User Guide for Positioning Function".

- Application instance



6.21.7 STOPDV: Pulse output stop instruction

LAD: STOPDV (S1) (S2) (S3) (D)	**Applicable model**	**VC3**
	Influenced flag bit	
IL: STOPDV (S1) (S2) (S3) (D)	**Step length**	**12**

Operand	Type	Applicable soft element													Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S3	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	BOOL			Y											

- Operand description

S1: Number of output pulses after the instruction is executed (relative address).

S2: Base speed during the deceleration.

S3: Time for deceleration from the original output speed to the base speed.

D: Number of the output point corresponding to the high-speed pulses.

- Function description

1. Executing the STOPDV instruction. It can started during the execution process of the PLSY, PLS, and positioning instructions, and stops the output action of the specified axis.

2. When the drive energy flow of the instruction is ON, the pulse output stops after running the specified number of pulses. When the specified number of pulses is 0, the instruction stops its output actionimmediately. When the specified number of pulses is greater than 0, the instruction continues the original output action first, then

decelerates to the base speed, and stops the output action when the base speed is reached.

3. The base speed and acceleration/deceleration time are also set in the special data register of the output axis. The execution of the instruction does not change the setting of the special data register. The base speed and acceleration/deceleration time during the execution of the instruction are executed according to the set instruction operand instead of using the configuration in the special data register.

4. The direction signal of the output axis does not need to be specified, and the direction signals specified in the original PLSY, PLS, and positioning instructions are automatically recognized. The ON/OFF state of the direction signal is not changed during the execution of the instruction.

- Note

1. The min. output pulse frequency that can be outputted actually, is determined by the formula below:

$$F_{\min\_acc} = \sqrt{\frac{F_{\max} \times 500}{T}}$$

In the above formula, $F_{\max}$ indicates the max. speed, set via SD85 and SD86 while T indicates the acceleration/deceleration time, set via SD87, withthe unit of ms. The calculation result $F_{\min\_acc}$ is the limit value of min. output frequency.

2. For the pulse output frequency ***S2***, the calculated frequency is output even if a value less than the calculated one is assigned.The frequency of the starting section of ACC and end part of DEC cannot be less than the above calculation result. If the max. speed is lower than the above calculation result, there is not pulse output.

3. When the number of output pulses is less than the number of pulses required for deceleration, the frequency can be decelerated.

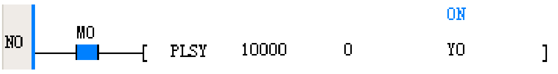
4. The output point number D corresponding to the high-speed pulses can specify Y0, Y1, Y2, Y3, Y4, Y5, Y6, and Y7.

5. When the instruction is executed, if the output axis is already in the stop state, no operation is performed; if the output axis is executing the LIN, CW, or CCW instructions, no operation is performed.

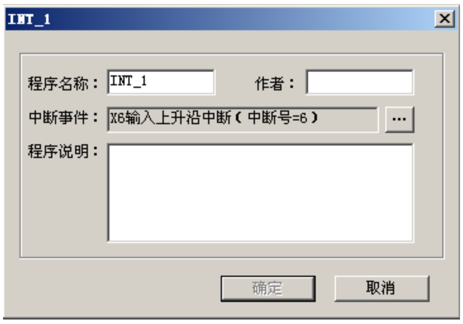
6. For details, refer to Chapter 11 "User Guide for Positioning Function".

- Application instance

In the main program, you can try out the PLSY instruction to drive Y0. The energy flow is controlled by the M0 element:



Setting the interrupt source for the interrupt subprogram, for example:



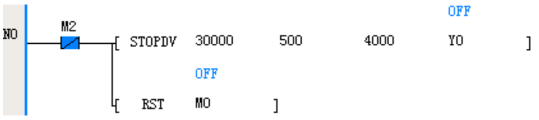
程序名称：INT_1

作者：

中断事件：X6输入上升沿中断（中断号=6）

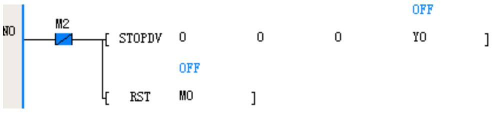
程序说明：

Adding the following statement to the interrupt subprogram:



In the above instruction, when X6 is set to ON, Y0 outputs some pulses and then decelerates to stop. If X6 is set to ON until Y0 stops completely, 30,000 pulses are outputted, which is not affected by the scan cycle.

For example, operand 1 in the STOPDV instruction in INT_1, is 0, the diagram is as shown below:



When the interrupt source event occurs (hereby referring to the rising edge of X6), Y0 stops immediately and is notaffected by the scan cycle.

Note that when the STOPDV instruction is called, it is necessary to cut off the energy flow of the high-speed instruction executed for Y0 in the main function to prevent the high-speed output from being restarted by scanning the instruction in the main function after Y0 is stopped.

6.21.8 LIN: Linear trajectory interpolation instruction

<table>
<tr><td colspan="3" rowspan="2">LAD:
LIN (S) (D1) (D2) (D3) (D4)</td><td>Applicable model</td><td colspan="2">VC3</td></tr>
<tr><td>Influenced flag bit</td><td colspan="2"></td></tr>
<tr><td colspan="3">IL: LIN (S) (D1) (D2) (D3) (D4)</td><td>Step length</td><td colspan="2">12</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="3">Applicable soft element</td><td>Indexing</td></tr>
</table>

S	DINT								D						
D1	BOOL			Y											
D2	BOOL			Y											
D3	BOOL			Y											
D4	BOOL			Y											

- Operand description

  S: The start address of the parameter table storage area.

  D1: The output point number corresponding to the X-axis pulse signal (or positive pulse signal). Only Y0 can be specified.

  D2: The output point number corresponding to the X-axis direction signal (or negative pulse signal). Only Y1 can be specified.

  D3: The output point number corresponding to the Y-axis pulse signal (or positive pulse signal). Only Y2 can be specified.

  D4: The output point number corresponding to the Y-axis direction signal (or negative pulse signal). Only Y3 can be specified.

- Function description

  1. Moving to the target position along a straight trajectory at the specified vector speed.

  2. Parameter table definition

<table>
<tr><th>D element</th><th colspan="3">Content</th></tr>
<tr><td>S</td><td colspan="3">Reserved</td></tr>
<tr><td rowspan="6">S+1</td><td colspan="3">Action mode and output logic relationship (configuration code in decimal)</td></tr>
<tr><td>Configuration</td><td>Mode</td><td>Output logic</td></tr>
<tr><td>00</td><td>Incremental type</td><td>Pulse+direction (Forward: ON/reverse: OFF)</td></tr>
<tr><td>01</td><td>Incremental type</td><td>Forward pulse+reverse puls</td></tr>
<tr><td>10</td><td>Absolute value type</td><td>Pulse+direction (Forward: ON/reverse: OFF)</td></tr>
<tr><td>11</td><td>Absolute value type</td><td>Forward pulse+reverse puls</td></tr>
<tr><td>S+2</td><td colspan="3">Resultant speed initial speed Fmin (Hz) (MSB)</td></tr>
<tr><td>S+3</td><td colspan="3">Resultant speed initial speed Fmin (Hz) (LSB)</td></tr>
<tr><td>S+4</td><td colspan="3">Resultant speed max. speed Fmax (Hz) (MSB)</td></tr>
<tr><td>S+5</td><td colspan="3">Resultant speed max. speed Fmax (Hz) (LSB)</td></tr>
<tr><td>S+6</td><td colspan="3">Acceleration/deceleration time T (ms) (MSB)</td></tr>
<tr><td>S+7</td><td colspan="3">Acceleration/deceleration time T (ms) (LSB)</td></tr>
<tr><td>S+8</td><td colspan="3">X-axis target position (moving distance) (MSB)</td></tr>
<tr><td>S+9</td><td colspan="3">X-axis target position (moving distance) (LSB)</td></tr>
<tr><td>S+10</td><td colspan="3">Y-axis target position (moving distance) (MSB)</td></tr>
<tr><td>S+11</td><td colspan="3">Y-axis target position (moving distance) (LSB)</td></tr>
</table>

  Among which:

  (1) In the incremental mode, the trajectory target adopts the relative address, which refers to the moving distance from the current position to the X and Y axes during the target period.

  (2) In the absolute value mode, the trajectory target adopts an absolute address, which refers to the absolute position coordinates of the target position on the X and Y axes.

- Note

  1. The output point numbers D1 and D3 corresponding to the two output axis pulse signals (or forward pulse signals) in the instruction must be used in groups. When the output group can only designatedas Y0 and Y2, Y1 and Y3 are used in combination with Y0 and Y2 respectively to provide the direction signal or negative pulse output signal.

  2. The output group (Y0 and Y2) can be specified as "Pulse + direction" mode or "Positive pulse + negative pulse" mode, the max. speed of the single axis is 200k while the resultant speedis a max.of 200K.

  3. The setting range of the moving distance for each axis is -8388608-+8388607 pulses.

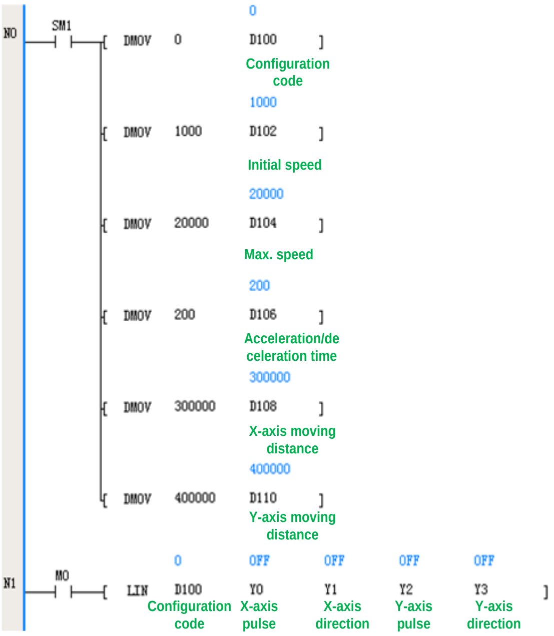
  4. It is not allowed to use multiple PLSY, PLS, or positioning instructions on the same high-speed pulse output port at the same time.

  5. The range of acceleration and deceleration time is 5-5000ms.

  6. Only trapezoidal acceleration/deceleration is supported, and the deceleration time cannot be set alone.

  7. The completion interrupt only supports one (Y0Y1) interrupt.

- Application instance


N0
SM1
DMOV 0 D100
0
Configuration code
DMOV 1000 D102
1000
Initial speed
DMOV 20000 D104
20000
Max. speed
DMOV 200 D106
200
Acceleration/deceleration time
DMOV 300000 D108
300000
X-axis moving distance
DMOV 400000 D110
400000
Y-axis moving distance
N1
M0
LIN D100 Y0 Y1 Y2 Y3
0 OFF OFF OFF OFF
Configuration code
X-axis pulse
X-axis direction
Y-axis pulse
Y-axis direction

6.21.9 CW: Clockwise arc trajectory interpolation

<table>
<tr><td rowspan="2">LAD:
├─┤ ├──[CW (S) (D1) (D2) (D3) (D4)]</td><td>Applicable model</td><td>VC3</td></tr>
<tr><td>Influenced flag bit</td><td></td></tr>
<tr><td>IL: CW (S) (D1) (D2) (D3) (D4)</td><td>Step length</td><td>12</td></tr>
</table>

Operand	Type	Applicable soft element													Indexing
S	DINT								D						
D1	BOOL			Y											
D2	BOOL			Y											
D3	BOOL			Y											
D4	BOOL			Y											

- Operand description

  S: The start address of the parameter table storage area.

  D1: The output point number corresponding to the X-axis pulse signal (or positive pulse signal). Only Y0 can be specified

  D2: The output point number corresponding to the X-axis direction signal (or negative pulse signal). Only Y1 can be specified.

  D3: The output point number corresponding to the Y-axis pulse signal (or positive pulse signal). Only Y2 can be specified.

  D4: The output point number corresponding to the Y-axis direction signal (or negative pulse signal). Only Y3 can be specified.

- Function description

  1. Moving to the target position along the arc trajectory in the clockwise direction at the specified line speed.

  2. Parameter table definition

<table>
<tr><th>D element</th><th>Content</th></tr>
<tr><td>S</td><td>Arc formation method<table><tr><th>Configuration</th><th>Mode</th></tr><tr><td>0</td><td>Center position designation</td></tr><tr><td>1</td><td>Passing position designation</td></tr></table></td></tr>
<tr><td>S+1</td><td>Action mode and output logic relationship (configuration code in decimal)<table><tr><th>Configuration</th><th>Mode</th><th>Output logic</th></tr><tr><td>00</td><td>Incremental type</td><td>Pulse+direction (Forward: ON/reverse: OFF)</td></tr><tr><td>01</td><td>Incremental type</td><td>Forward pulse+reverse puls</td></tr><tr><td>10</td><td>Absolute value type</td><td>Pulse+direction (Forward: ON/reverse: OFF)</td></tr><tr><td>11</td><td>Absolute value type</td><td>Forward pulse+reverse puls</td></tr></table></td></tr>
<tr><td>S+2</td><td>Resultant speed initial speed (MSB)</td></tr>
<tr><td>S+3</td><td>Resultant speed initial speed (LSB)</td></tr>
<tr><td>S+4</td><td>Resultant speed (MSB)</td></tr>
<tr><td>S+5</td><td>Resultant speed (LSB)</td></tr>
<tr><td>S+6</td><td>Acceleration/deceleration time T (ms) (MSB)</td></tr>
<tr><td>S+7</td><td>Acceleration/deceleration time T (ms) (LSB)</td></tr>
<tr><td>S+8</td><td>X-axis moving distance (target position) (MSB)</td></tr>
<tr><td>S+9</td><td>X-axis moving distance (target position) (LSB)</td></tr>
<tr><td>S+10</td><td>Y-axis moving distance (target position) (MSB)</td></tr>
<tr><td>S+11</td><td>Y-axis moving distance (target position) (LSB)</td></tr>
<tr><td>S+12</td><td>X-axis coordinate of the circle center position/passing point (MSB)</td></tr>
<tr><td>S+13</td><td>X-axis coordinate of the circle center position/passing point (LSB)</td></tr>
<tr><td>S+14</td><td>Y-axis coordinate of the circle center position/passing point (MSB)</td></tr>
<tr><td>S+15</td><td>Y-axis coordinate of the circle center position/passing point (LSB)</td></tr>
</table>

Among which:

(1) In the incremental mode, the trajectory target adopts the relative address, which refers to the moving distance from the current position to the X and Y axes during the target period.

(2) In the absolute value mode, the trajectory target adopts an absolute address, which refers to the absolute position coordinates of the target position on the X and Y axes.

- Note

  1. The output point numbers D1 and D3 corresponding to the two output axis pulse signals (or forward pulse signals) in the instruction must be used in groups. When the output group can only designatedas Y0 and Y2, Y1 and Y3 are used in combination with Y0 and Y2 respectively to provide the direction signal or negative pulse output signal.

  2. The output group (Y0 and Y2) can be specified as "Pulse + direction" mode or "Positive pulse+ negative pulse" mode, and the max. speed is 200k.

  3. The setting range of the moving distance for each axis is -8388608-8388607 pulses.

4. It is not allowed to use multiple PLSY, PLS, or positioning instructions on the same high-speed pulse output port at the same time.

5. The coordinates of the passing point specified in the passing position mode refer to the passing position of the entire circle, but the user-defined trajectory path may not pass through this point.

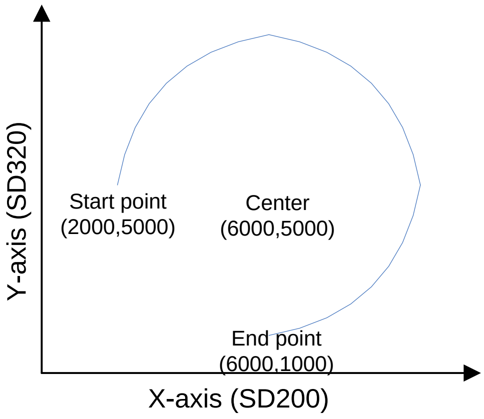
6. The range of acceleration and deceleration time is 5-5000ms.

7. Only trapezoidal acceleration/deceleration is supported, and the deceleration time cannot be set alone.

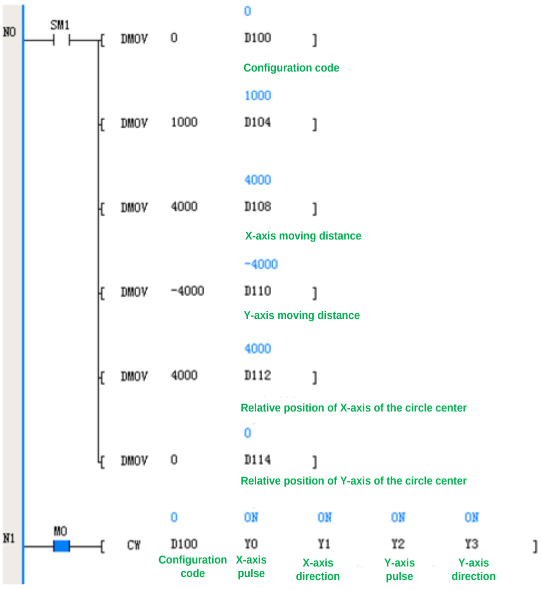
8. The completion interrupt only supports one (Y0Y1) interrupt.

- Application instance

Assuming the current position value of the SD element is (2000, 5000), and you want to draw the arc shown in the following figure:


Y-axis (SD320)
Start point
(2000,5000)
Center
(6000,5000)
End point
(6000,1000)
X-axis (SD200)

You can adopt the incremental type, the displacement of the end point to the start point is (4000, -4000), and the displacement of the center to the start point is (4000, 0). You can program like this:



```
N0  SM1
    ┤ ├──┬[ DMOV   0      D100  ]
         ├[ DMOV   1000   D104  ]
         ├[ DMOV   4000   D108  ]
         ├[ DMOV   -4000  D110  ]
         ├[ DMOV   4000   D112  ]
         └[ DMOV   0      D114  ]
N1  M0
    ──────[ CW  D100  Y0  Y1  Y2  Y3 ]
```

0
Configuration code
1000
4000
X-axis moving distance
-4000
Y-axis moving distance
4000
Relative position of X-axis of the circle center
0
Relative position of Y-axis of the circle center
0
ON
ON
ON
ON
Configuration code
X-axis pulse
X-axis direction
Y-axis pulse
Y-axis direction

6.21.10 CCW: Counterclockwise arc trajectory interpolation instruction

LAD: CCW (S) (D1) (D2) (D3) (D4)	Applicable model	VC3
	Influenced flag bit	
IL: CCW (S) (D1) (D2) (D3) (D4)	Step length	12

Operand	Type	Applicable soft element													Indexing
S	DINT								D						
D1	BOOL			Y											
D2	BOOL			Y											
D3	BOOL			Y											
D4	BOOL			Y											

- Operand description

S: The start address of the parameter table storage area.

D1: The output point number corresponding to the X-axis pulse signal (or positive pulse signal). Only Y0 can be specified.

D2: The output point number corresponding to the X-axis direction signal (or negative pulse signal). Only Y1 can be specified.

D3: The output point number corresponding to the Y-axis pulse signal (or positive pulse signal). Only Y2 can be specified.

D4: The output point number corresponding to the Y-axis direction signal (or negative pulse signal). Only Y3 can be specified.

- Function description

1. Moving to the target position along the arc trajectory in the counter clock wise direction at the specified line speed.

2. Parameter table definition

<table>
<tr><th>D element</th><th colspan="3">Content</th></tr>
<tr><td rowspan="4">S</td><td colspan="3">Arc formation method</td></tr>
<tr><td>Configuration</td><td colspan="2">Mode</td></tr>
<tr><td>0</td><td colspan="2">Center position designation</td></tr>
<tr><td>1</td><td colspan="2">Passing position designation</td></tr>
<tr><td rowspan="6">S+1</td><td colspan="3">Action mode and output logic relationship (configuration code in decimal)</td></tr>
<tr><td>Configuration</td><td>Mode</td><td>Output logic</td></tr>
<tr><td>00</td><td>Incremental type</td><td>Pulse+direction (Forward: ON/reverse: OFF)</td></tr>
<tr><td>01</td><td>Incremental type</td><td>Forward pulse+reverse pulse</td></tr>
<tr><td>10</td><td>Absolute value type</td><td>Pulse+direction (Forward: ON/reverse: OFF)</td></tr>
<tr><td>11</td><td>Absolute value type</td><td>Forward pulse+reverse pulse</td></tr>
<tr><td>S+2</td><td colspan="3">Resultant speed initial speed (MSB)</td></tr>
<tr><td>S+3</td><td colspan="3">Resultant speed initial speed (LSB)</td></tr>
<tr><td>S+4</td><td colspan="3">Resultant speed (MSB)</td></tr>
<tr><td>S+5</td><td colspan="3">Resultant speed (LSB)</td></tr>
<tr><td>S+6</td><td colspan="3">Acceleration/deceleration time T (ms) (MSB)</td></tr>
<tr><td>S+7</td><td colspan="3">Acceleration/deceleration time T (ms) (LSB)</td></tr>
<tr><td>S+8</td><td colspan="3">X-axis moving distance (target position) (MSB)</td></tr>
<tr><td>S+9</td><td colspan="3">X-axis moving distance (target position) (LSB)</td></tr>
<tr><td>S+10</td><td colspan="3">Y-axis moving distance (target position) (MSB)</td></tr>
<tr><td>S+11</td><td colspan="3">Y-axis moving distance (target position) (LSB)</td></tr>
<tr><td>S+12</td><td colspan="3">X-axis coordinate of the circle center position/passing point (MSB)</td></tr>
<tr><td>S+13</td><td colspan="3">X-axis coordinate of the circle center position/passing point (LSB)</td></tr>
<tr><td>S+14</td><td colspan="3">Y-axis coordinate of the circle center position/passing point (MSB)</td></tr>
<tr><td>S+15</td><td colspan="3">Y-axis coordinate of the circle center position/passing point (LSB)</td></tr>
</table>

Among which:

(1) In the incremental mode, the trajectory target adopts the relative address, which refers to the moving distance from the current position to the X and Y axes during the target period.

(2) In the absolute value mode, the trajectory target adopts an absolute address, which refers to the absolute position coordinates of the target position on the X and Y axes.

- Note

1. The output point numbers D1 and D3 corresponding to the two output axis pulse signals (or forward pulse signals) in the instruction must be used in groups. When the output group can only designatedas Y0 and Y2, Y1 and Y3 are used in combination with Y0 and Y2 respectively to provide the direction signal or negative pulse output signal.

2. The output group (Y0 and Y2) can be specified as "Pulse + direction" mode or "Positive pulse+ negative pulse" mode, and the max. speed is 200k.

3. The setting range of the moving distance for each axis is -8388608-8388607 pulses.

4. It is not allowed to use multiple PLSY, PLS, or positioning instructions on the same high-speed pulse output port at the same time.

5. The coordinates of the passing point specified in the passing position mode refer to the passing position of the entire circle, and the user-defined trajectory path may not pass through this point.

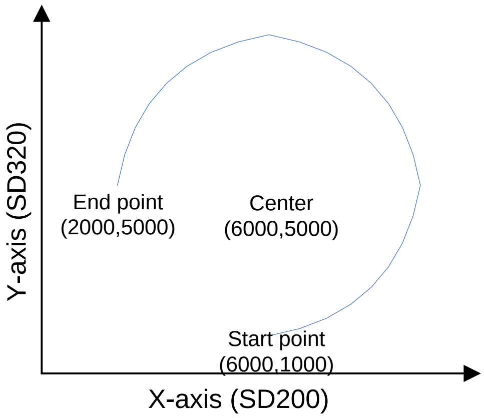
6. The range of acceleration and deceleration time is 5-5000ms.

7. Only trapezoidal acceleration/deceleration is supported, and the deceleration time cannot be set alone.

8. The completion interrupt only supports one (Y0Y1) interrupt.

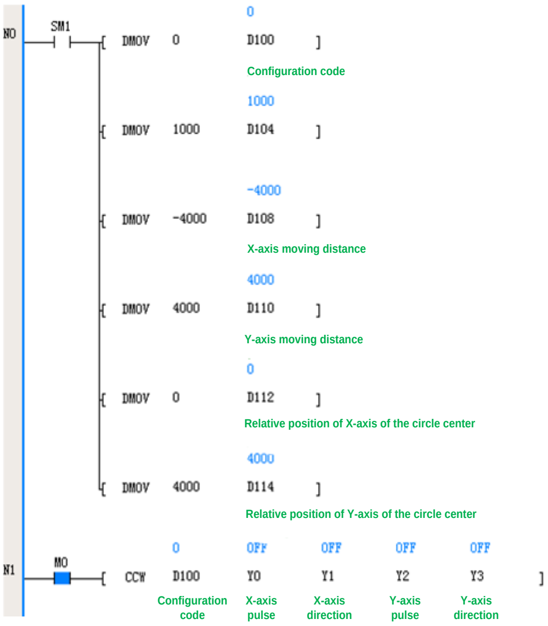
- Application instance

Assuming the current position value of the SD element is (6000, 1000), and you want to draw the arc shown in the following figure:


Y-axis (SD320)
End point
(2000,5000)
Center
(6000,5000)
Start point
(6000,1000)
X-axis (SD200)

You can adopt the incremental type, the displacement of the end point to the start point is (4000, -4000), and the displacement of the center to the start point is (0, 4000).

You can program like this:



```
N0   SM1
     | |----[ DMOV   0      D100  ]
           [ DMOV   1000   D104  ]
           [ DMOV   -4000  D108  ]
           [ DMOV   4000   D110  ]
           [ DMOV   0      D112  ]
           [ DMOV   4000   D114  ]
N1   M0
     ---[ CCW   D100   Y0   Y1   Y2   Y3   ]
```

0
Configuration code
1000
-4000
X-axis moving distance
4000
Y-axis moving distance
0
Relative position of X-axis of the circle center
4000
Relative position of Y-axis of the circle center
0
OFF
OFF
OFF
OFF
Configuration code
X-axis pulse
X-axis direction
Y-axis pulse
Y-axis direction

6.21.11 MOVELINK: Synchronous control instruction

LAD: MOVELINK *(S1) (S2) (S3) (S4) (S5) (S6)*	**Applicable model**		
	Influenced flag bit		
IL: MOVELINK (*S1*) (*S2*) (*S3*) (*S4*) (*S5*) (*S6*)	**Step length**	**17**	

Operand	Type	Applicable soft element													Indexing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S3	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		
S4	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		
S5	BOOL			Y											
S6	BOOL			Y							C				

- Operand description

  S1: Total number of pulses generated by the slave axis when it follows the spindle axis.

  S2: Total number of pulses generated by the spindle axis when it is followed by the slave axis.

  S3: Number of pulses generated by the spindle axis before reaching the constant speed synchronization

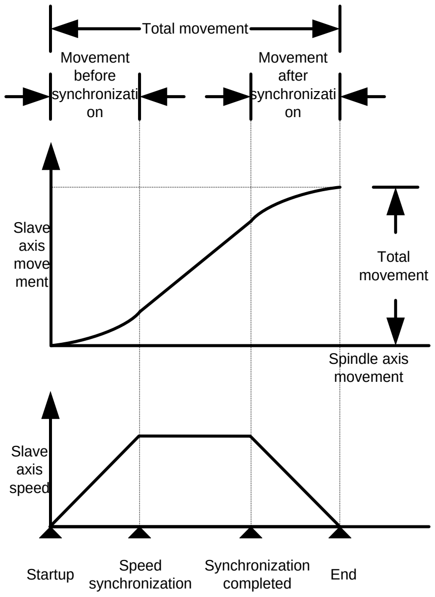
  S4: Number of pulses generated by the spindle axisafter ending the constant speed synchronization

  S5: Output point number corresponding to the slave axis pulse signal. Range: Y4 and Y5.

  S6: Output point number corresponding to the spindle axis pulse signalor high-speed counter number. Range: Y0–Y7, C236–C263.

- Function description

  1. This instruction implements a simple synchronization function for two-axis motion. The slave axis measures and follows the speed and position of the spindle axis. The spindle axis can be used as a high-speed output or input port of the module.



2. As shown in the figure above, after this instruction is started, the slave axis starts from the stop state and moves by following the spindle axis after a period of acceleration and deceleration. When the spindle axis outputs the specified number of pulses ***S3***, the speed of the slave axis is the same as that of the spindle axis, and the constant speed synchronization starts. When there are S4 pulses to be generated by the spindle axis before it reaches the destination position, the constant speed synchronization is stopped, and the slave axis starts to decelerate the speed. When the spindle axis reaches the target position, the slave axis stops.

3. When the spindle axis is defined as an output axis, operand S6 is set to an output point number Y0–Y7;

when defined as an input axis, operand ***S6*** is set toC236–C263.

4. When the energy flow of the instruction is turned on, the slave axis starts running. This instruction supports the calling mode and subprogram, and the interrupt program calling methodin the main program.

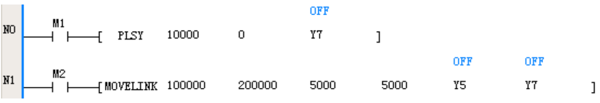
5. The direction signal of the output axis does not need to be specified, and the ON/OFF state of the direction signal is not changed during the following process.

- Note

1. It is not allowed to use multiple PLSY, PLS, or positioning instructions on the same high-speed pulse output port at the same time.

2. The output point number D corresponding to the high-speed pulses can be specified as: Y0, Y2, Y4, Y5, Y6, Y7. Y1 and Y3 are used in combination with Y0 and Y2 respectively to provide the direction signal or negative pulse output signal.

- Application instance



6.21.12 GEARBOX: Electronic gear instruction

LAD: GEARBOX *(D1)* *(S1)* *(D2)* *(S2)*	**Applicable model**	
	Influenced flag bit	
IL: GEARBOX (*D1*) (*S1*) (*D2*) (*S2*)	**Step length**	**9**

Operand	Type	Applicable soft element													Indexing
D1	BOOL			Y											
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D2	BOOL			Y											
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√

- Operand description

  D1: The output point number corresponding to the spindle axis signal. Range: Y0–Y7.

  S1: This operand together with ***S2*** determines the ratio of the electronic gear.

  D2: The output point number corresponding to the slave axis pulse signal. Range: Y4 and Y5.

  S2: This operand together with ***S1*** determines the ratio of the electronic gear.

  The range of the electronic gear ratio (***S1***/***S2***): 1/10000–10000.

- Function description

  1. Enabling the slave axis to follow the spindle axis according to the electronic gear ratio (S1/S2). When the spindle axis sends N pulses, the slave axis sends N × S1/S2 pulses. When the spindle axis outputs pulses at a frequency of F, the slave axis outputs pulses at a frequency of F × S1/S2.

  2. When the energy flow of the instruction is turned on, the slave axis starts running.

  3. The direction signal of the output axis does not need to be specified, and it does not change its ON/OFF state of the direction signalduring the follow-up process.

  4. When the electronic gear ratio is less than 0, that is, one value of S1, S2 is positive and the other is negative, and the relevant SD elements of the slave axis are decreased progressively with the pulse output.

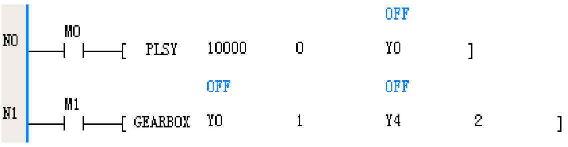
- Note

  1. It is not allowed to use multiple PLSY, PLS, or positioning instructions on the same high-speed pulse output port at the same time.

  2. If the electronic gear ratio is outside the specified range, there is no pulse output.

  3. The output point number D corresponding to the high-speed pulses can be specified as: Y0, Y2, Y4, Y5, Y6, and Y7. Y1 and Y3 are used in combination with Y0 and Y2 respectively to provide the direction signal or negative pulse output signal.

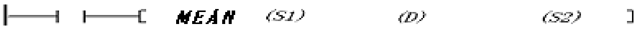
- Application instance



```
                                      OFF
N0   M0
   --| |----[ PLSY    10000     0     Y0        ]
                      OFF             OFF
N1   M1
   --| |----[ GEARBOX Y0        1     Y4     2        ]
```

6.22 Data processing instructions

6.22.1 MEAN: Mean instruction

LAD: 	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: MEAN (*S1*) (*D*) (*S2*)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T		R	√
S2	INT	Constant							D					R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C			R	√

- Operand description

  S1: Number of the start word element that stores data on which the mean operation is to be performed

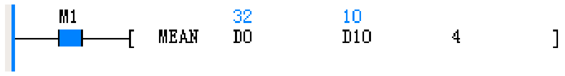
  S2: Number of pieces of data on which the mean operation is to be performed (1–64)

  D: Number of the word element that stores the obtained mean value

- Function description

  1. Storing the mean of S2 16-bit data starting from S1 in D, and rounding off the remainder.

- Application instance



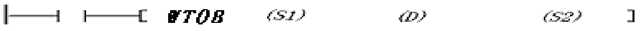
LD M1

MEAN D0 D10 4

Taking the average value of four unit

sof data starting from D0, and storing the obtained value in D10 when M1 = ON. When D0=32, D1=10, D2=15, and D3=-14, D10=10.

6.22.2 WTOB: Byte-unit data separation instruction

LAD: 	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: WTOB (*S1*) (*D*) (*S2*)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT								D	SD	C	T		R	√
S2	INT								D	SD	C	T		R	√
D	INT	Constant							D					R	√

- Operand description

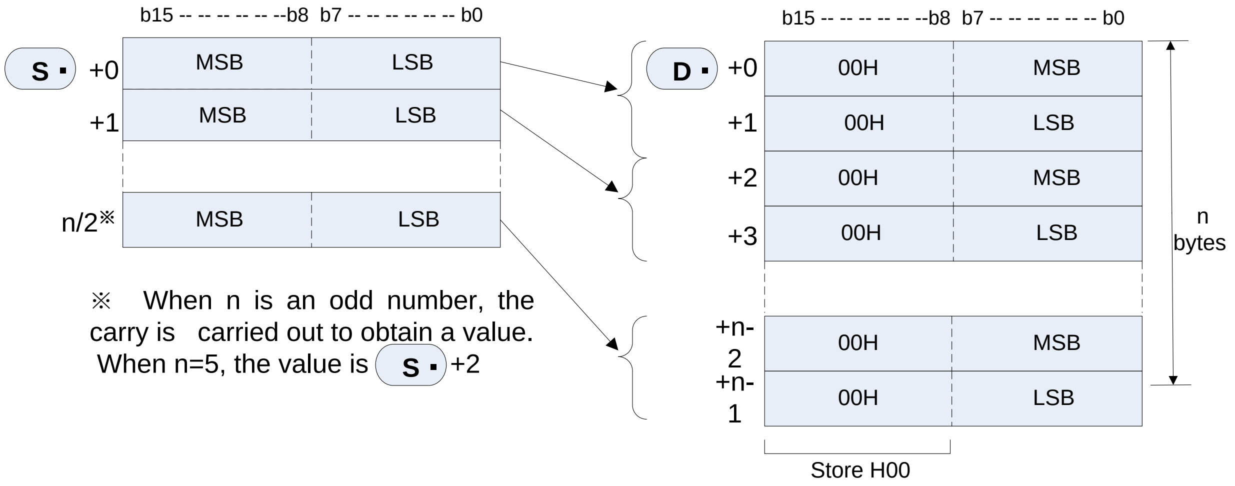
  S1: Number of the start soft element that stores data to be separated in byte unit

  S2: Number of byte data to be separated (S2 ≥ 0)

  D: Number of the start soft element that stores the result that has been separated in byte unit

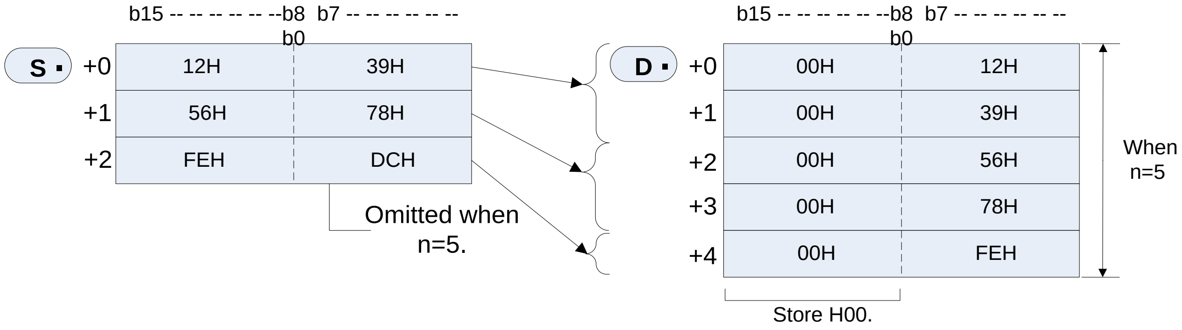
- Function description

  1. Separating the 16-bit data stored in S2/2 soft elements from S1 into S2 bytes, storing them in the low bytes of S2 soft elements starting from D, and clearing the high bytes.


b15 -- -- -- -- -- --b8 b7 -- -- -- -- -- -- b0
S ▪ +0
+1
n/2※
MSB
LSB
b15 -- -- -- -- -- --b8 b7 -- -- -- -- -- -- b0
D ▪ +0
+1
+2
+3
+n-2
+n-1
00H
MSB
LSB
n bytes
Store H00

※ When n is an odd number, the carry is carried out to obtain a value. When n=5, the value is S ▪ +2

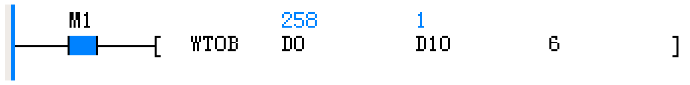
2. When S2 is odd, only the high byte (8 bits) is the object data in the last data of the separated source. For example, when n=5, the data of the low bytes of S﹣S+2 are stored in D﹣D+4.


b15 -- -- -- -- -- --b8 b7 -- -- -- -- -- -- b0
S ▪ +0
+1
+2
12H
39H
56H
78H
FEH
DCH
Omitted when n=5.
b15 -- -- -- -- -- --b8 b7 -- -- -- -- -- -- b0
D ▪ +0
+1
+2
+3
+4
00H
12H
39H
56H
78H
FEH
When n=5
Store H00.

3. When S2=0, the instruction is not executed.

4. The source and destination operands cannot overlap.

- Application instance


M1
WTOB
258
D0
1
D10
6

LD M1

WTOB D0 D10 6

Separating three units of data starting from D0 into six units of data according to the high and low bytes, and storing the obtained data in six units starting from D10 when M1=ON. When D0=0x102, D1=0x304, and D2=0x506, D10=0x01, D11=0x02, D12=0x03, D13=0x04, D14=0x05, and D15=0x06.

6.22.3 BTOW: Byte-unit data combination instruction

LAD: BTOW (S1) (D) (S2)	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: BTOW (S1) (D) (S2)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT								D	SD	C	T		R	√
S2	INT								D	SD	C	T		R	√
D	INT	Constant							D					R	√

- Operand description

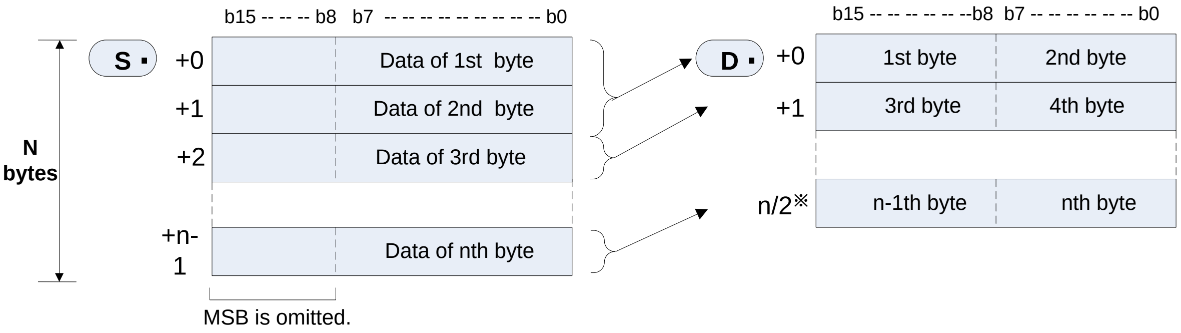
S1: Number of the start soft element stores data to be combined in byte unit

S2: Number of byte data to be combined (S2 ⩾ 0)

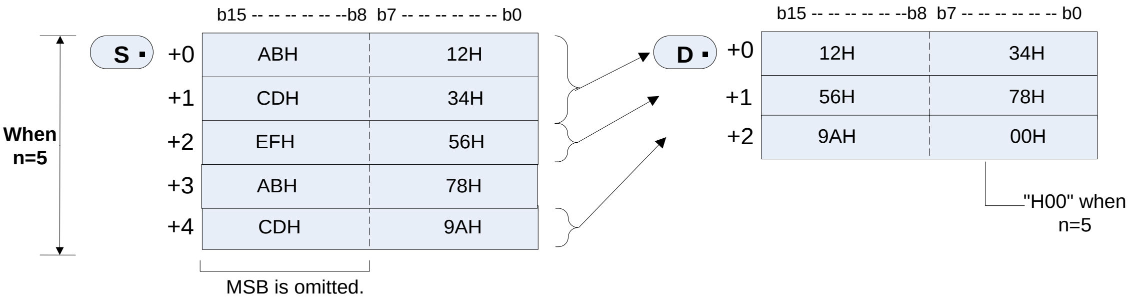
D: Number of the start soft element that stores the result that has been combined in byte unit

- Function description

1. Combining the least significant bytes (8 bits) of S2 16-bit data starting from S1 into 16-bit data, and storing these data in S2/2 soft elements starting from D. The most significant bytes (8 bits after S1) of the 16-bit data in the data source are ignored.


b15 -- -- -- b8 b7 -- -- -- -- -- -- -- -- -- b0
S ▪
+0
+1
+2
+n-1
N bytes
Data of 1st byte
Data of 2nd byte
Data of 3rd byte
Data of nth byte
MSB is omitted.
b15 -- -- -- -- -- --b8 b7 -- -- -- -- -- -- b0
D ▪
+0
+1
n/2※
1st byte
2nd byte
3rd byte
4th byte
n-1th byte
nth byte

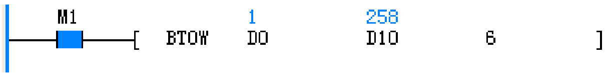
2. When S2 is odd, the least significant byte that is combined finally is cleared.


b15 -- -- -- -- -- --b8 b7 -- -- -- -- -- -- b0
S ▪
When n=5
+0 ABH 12H
+1 CDH 34H
+2 EFH 56H
+3 ABH 78H
+4 CDH 9AH
MSB is omitted.
b15 -- -- -- -- -- --b8 b7 -- -- -- -- -- -- b0
D ▪
+0 12H 34H
+1 56H 78H
+2 9AH 00H
"H00" when n=5

3. When S2=0, the instruction is not executed.

4. The source and destination operands cannot overlap.

- Application instance


M1
BTOW
1
D0
258
D10
6

LD M1

BTOW D0 D10 6

Combining six unitsof data starting from D0 into three units of data, and storing the obtained data in three units starting from D10 when M1=ON. When D0=0x01, D1=0x02, D2=0x03, D3=0x04, D4=0x05, and D5=0x06, D10=0x102, D11=0x304, and D12=0x506.

6.22.4 UNI: Instruction for combining 4bits of 16-bit data

LAD:	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: UNI (*S1*) (*D*) (*S2*)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT								D	SD	C	T		R	√
S2	INT								D	SD	C	T		R	√
D	INT	Constant							D					R	√

- Operand description

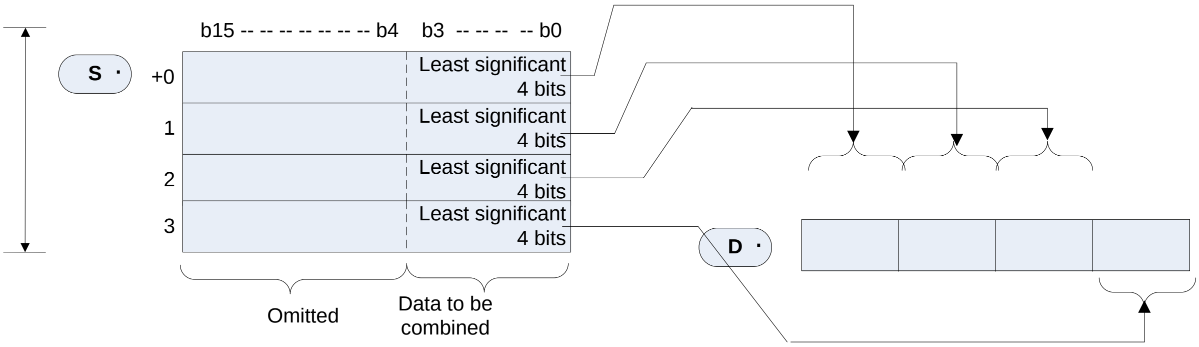
S1: Number of the start soft element that stores data to be combined

S2: Number of the combined data (when the range of ***S2*** is 0—4, and S2=0, there is no processing)

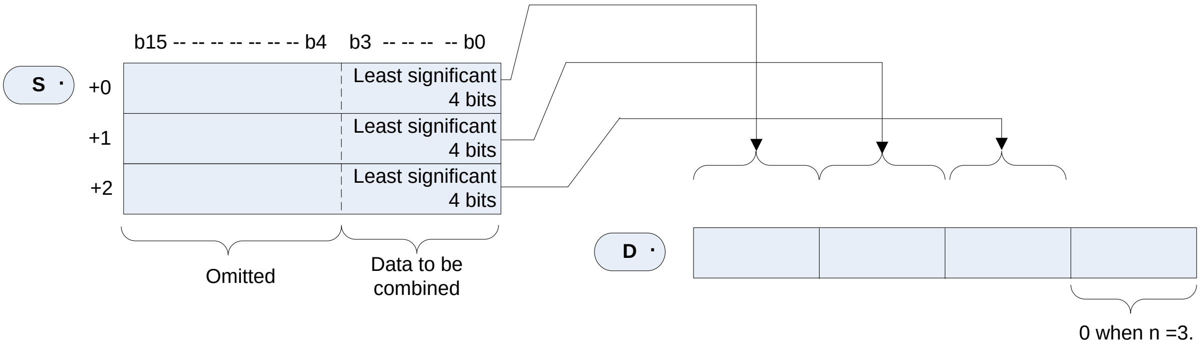
D: Number of the soft element that stores the combined data

- Function description

1. Storing S2 16-bit data starting from S1 in S2 soft elements starting from D.


b15 -- -- -- -- -- -- -- b4
b3 -- -- -- -- b0
S ·
+0
1
2
3
Least significant 4 bits
Least significant 4 bits
Least significant 4 bits
Least significant 4 bits
D ·
Omitted
Data to be combined

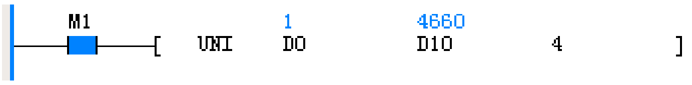
2. When S2 is ranging from 1 to 3, the ones of the LSB {4 × (4-S2)} of D are zero.


b15 -- -- -- -- -- -- -- b4
b3 -- -- -- -- b0
S ·
+0
+1
+2
Least significant 4 bits
Least significant 4 bits
Least significant 4 bits
D ·
Omitted
Data to be combined
0 when n =3.

3. When the range of ***S2*** is 1-4, and ***S2***=0, the instruction is not executed.

4. The source and destination operands cannot overlap.

- Application instance



```
 M1              1          4660
─┤├──[  UNI     D0          D10        4        ]
```

LD M1

UNI D0 D10 4

Combining the lower four bits of four units of data starting from D0, and storing them in D10. When D0=0x01, D1=0x02, D2=0x03, and D3=0x04, D10=0x1234.

6.22.5 DIS: Instruction for separating 4bitsof 16-bit data

LAD:	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: DIS (*S1*) (*D*) (*S2*)	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT								D	SD	C	T		R	√
S2	INT								D	SD	C	T		R	√
D	INT	Constant							D					R	√

- Operand description

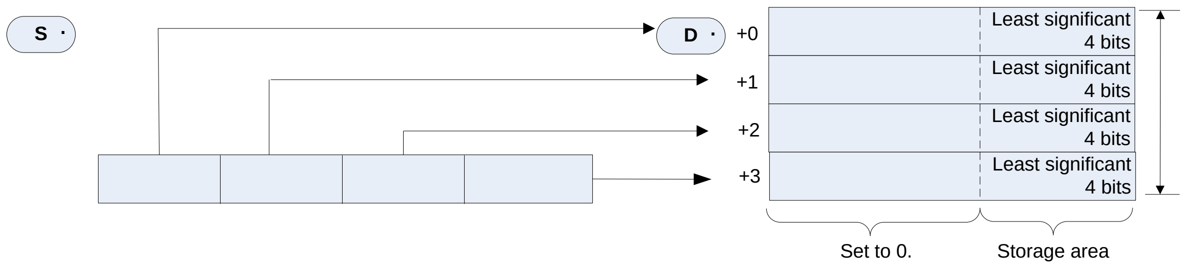
S1: Number of the start soft element that stores data to be separated

S2: Number of the separated data (when the range of ***S2*** is 0—4, and ***S2***=0, there is no processing)

D: Number of the soft element that stores the separated data

- Function description

1. Storing S2 16-bit data starting from S1 in S2 soft elements starting from D.

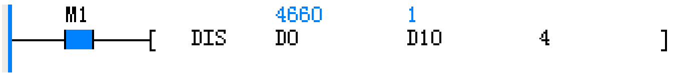


2. The valid range of S2 is 1—4, and the rest of the data do not execute the instruction.

3. The upper 12 bits of S2 soft elements starting from D are cleared.

4. The source and destination operands cannot overlap.

- Application instance



LD M1

DIS D0 D10 4

Separating every 4 bits of the data in the D0 unit, and storing these data in four units starting from D10 when M1=ON. When D0=0x1234, D10=0x01, D11=0x02, D12=0x03, and D13=0x04.

6.22.6 ANS: Signal alarm set instruction

LAD:	**Applicable model**	VC3 VC5
	Influenced flag bit	**Zero flag, carry flag, and borrow flag**
IL: ANS *(S1)* *(S2)* *(D)*	**Step length**	**7**

Operand	Type	Applicable soft element													Indexing
S1	INT											T			√
S2	INT	Constant							D					R	√
D	BOOL			S											√

- Operand description

S1: Timing timer number for judging time, only applicable for 100ms timer with the range of T0-T209

S2: Data for judging time (1—32767)

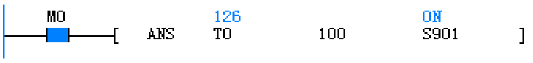
D: Set signal alarm soft elements. Range: S900—S999

- Function description

1. Setting D when the energy flow duration is greater than S2; resetting the timer S1 and not setting D when the energy flow duration of the instruction is less than S2; resetting S1 when the energy flow is invalid, S1 reset.

Address No.	Name	function
SM400	Enable the signal alarm	When SM400 is set to ON, the following SM401 and SD401 work
SM401	Signal alarm acts	SM401 is set to ON when there is any action in state S900-S999
SD401	Mini.number of the On state	Storing the mini.number of actions in S900-S999

- Application instance

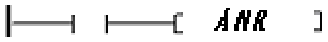


LD M0

ANS T0 100 S901

Setting S901 to on when the energy flow is valid, and not be disconnected within 10 seconds.

6.22.7 ANR: Signal alarm reset instruction

<table>
<tr><td colspan="3" rowspan="2">LAD:
</td><td>Applicable model</td><td>VC3 VC5</td></tr>
<tr><td>Influenced flag bit</td><td>Zero flag, carry flag, and borrow flag</td></tr>
<tr><td colspan="3">IL: ANR</td><td>Step length</td><td>1</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="2">Applicable soft element</td><td>Indexing</td></tr>
</table>

- Operand description

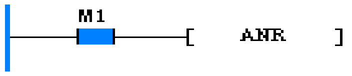
  No operand.

- Function description

  1. Resetting the running state of the signal alarms S900-S999 when the energy flow is valid; resetting the one with the mini. number if there are multiple state actions;resetting the one with the next mini. number, when the energy flow is valid again.

Address No.	Name	function
SM400	Enable the signal alarm is valid	When SM400 is set to ON, the following SM401 and SD401 work
SM401	Signal alarm acts	SM401 is set to ON when there is any action in state S900-S999
SD401	Mini.number of the on state	Storing the mini.number of actions in S900-S999

- Application instance



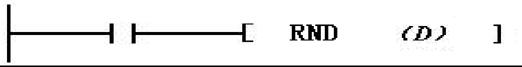
LD M1

ANR

When the energy flow is valid, if there are multiple Ss set by ANS, the one with the mini. number is reset.

6.23 Other instructions

6.23.1 RND: Instruction for generating random numbers

<table>
<tr><td colspan="8" rowspan="2">LAD:
</td><td colspan="3">Applicable model</td><td colspan="6">VC2 VC3 VC5</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="6">Zero flag bit</td></tr>
<tr><td colspan="8">IL: RND <i>(D)</i></td><td colspan="3">Step length</td><td colspan="6">3</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable soft element</td><td>Indexing</td></tr>
<tr><td>D</td><td>INT</td><td></td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td></td><td>Z</td><td>R</td><td>√</td></tr>
</table>

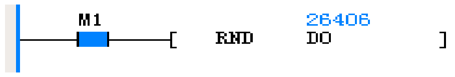
- Operand description

  D: Number of the start soft element that stores the random number

- Function description

  1. When a pseudo-random number ranging from 0 to 32767 is generated,its value is stored in ***D*** as a random number; if the generated random number is 0, the zero flag bit (SM270) is set.

- Application instance



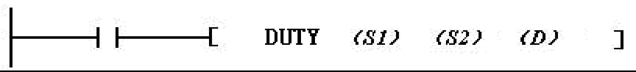
LD M1

RND D0

Generating a random number,storing it in D0, and D0=26406 when M1=ON.

6.23.2 DUTY: Instruction for generating timed pulses

LAD:



Applicable model	VC2 VC3 VC5
Influenced flag bit	
Step length	**7**

IL: DUTY *(S1) (S2) (D)*

Operand	Type	Applicable soft element															Indexing
S1	INT	Constant		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	BOOL	SM															

- Operand description

S1: ON scanning times

S2: OFF scanning times

D: Destination address of timed pulse output

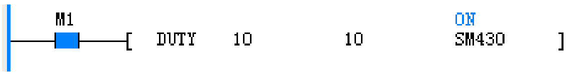
- Function description

1. The timed pulse output unit D changes with the ON and OFF scanning times (specified in S1 and S2, respectively).
2. SM unit. Range: SM430-SM434.

Destination address of timed pulse output	Soft elements for counting the number of scans
SM430	SD430
SM431	SD431
SM432	SD432
SM433	SD433
SM434	SD434

3. This instruction can be used for 5 times, but the same timing clock output destination address cannot be used in multiple DUTY instructions.

- Application instance



When M1=ON, 10 scans are ON, and 10 scans are OFF in SM430, and the count value of the scan times is saved in D430 at the same time.

- Note

When the operation starts at the rising edge of the instruction, the energy flow does not stop even if it is cut off. It stops in STOP or upon power outage.

Chapter 7 SFC Tutor

This chapter introduces the basic programming concepts, basic and complex programming methods, and note ofSFC.

Chapter 7 SFC Tutor 251

- 7.1 Introduction to SFC........252
  - 7.1.1 What is SFC........252
  - 7.1.2 What is SFC of VC series PLCs........252
  - 7.1.3 Basic concept of SFC........252
  - 7.1.4 Programming symbols and their usage........252
  - 7.1.5 SFC program structure........253
  - 7.1.6 Execution of SFC program........257
- 7.2 Relationship between SFC and LAD........258
  - 7.2.1 STL instruction and step states........258
  - 7.2.2 SET instruction........259
  - 7.2.3 RET instruction and SFC program segment........259
  - 7.2.4 OUT instruction and RST instruction........259
  - 7.2.5 SFC selection branch, parallel branch and merge........259
- 7.3 How to program With SFC........259
- 7.4 Notesin SFC programming........260
  - 7.4.1 Common programming errors........260
  - 7.4.2 Programming tricks........263
- 7.5 Examples of SFC programming........264
  - 7.5.1 Simple sequential structure........265
  - 7.5.2 Selection branch structure........266
  - 7.5.3 Parallel branch structure........269

7.1 Introduction to SFC

7.1.1 What is SFC

The Sequential Function Chart, or SFC, is a programming language that developed and got popular in recent years. SFC can turn a PLC programming project into a structured flow chart. By using the programming elements and language structure compliant with IEC61131-3 standard, it divides a complex systematic process into sequential multi-step procedures and transitions between the procedures, and thus implements sequential control.

The SFC programming is direct and sequential. Each procedure and conversion condition after decomposition is a relatively simple program process, which isideal for the sequential control application. These advantages explain why it has been widely used.

7.1.2 What is SFC of VC series PLCs

The SFC of VC series PLCs is a programming language used by VEICHI's VC series PLC products. In addition to the standard SFC functions, one or more LAD program blocks can be built in the SFC of VC series PLCs.

The program written with SFC of VC series PLCs can be converted into corresponding LAD or IL program.

The SFC of VC series PLCs can also support a maximum of 20 independent processes. These independent processes can be run independently, that is to say, the step state within each process are scanned and transferred separately by process. Jumping among the independent procedures is enabled.

7.1.3 Basic concept of SFC

The SFC has two basic concepts: step state and transfer. Other concepts, such as jump, branch and multiple independent processes, are derived from these two basic concepts.

- Stepstate

  1. Definition of step state

  A step state is actually an independent program, representing a working state or an operation in the sequence control process. Putting multiple step states together in a organic way can form a complete SFC program.

  2. Execution of step state

  In a SFC program, each step state is represented by a fixed S element.

  A step state is valid when it is being executed. For a valid step state, the state of its corresponding S element is ON, and the PLC scans and executes all of the instruction sequences within it. A step state is invalid when it is not executed. For an invalid step state, the state of its corresponding S elementis OFF, and the PLC does not scan and execute its instruction sequences.

- Transfer

  The sequence control process is actually a series of step state switching processes. A PLC that is executing a step state leaves the current step state, and enter and execute a new step state if certain logic conditions are met. This switching process is called the transfer of the step state.

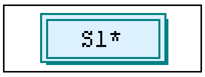
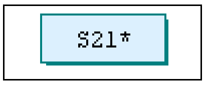
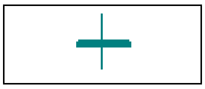
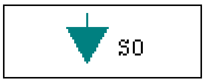
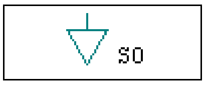
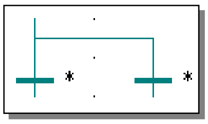
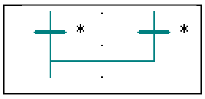
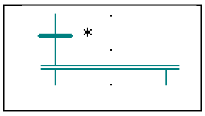
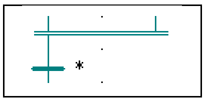
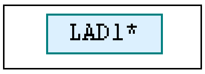
  The logical condition that triggers the transfer is called the step transfer condition.

7.1.4 Programming symbols and their usage

- Programming symbols

  The VC series PLC SFC programming language consists of the following basic programming symbols:

Table 7-1 Programming symbols

Symbol name	Symbol	Description
Initial step		An initial step state, numbered as Sn. The "n" cannot be repeated. The execution of a SFC program needs to start with an initial step symbol. The address range of S soft element corresponding to an initial step symbol is S0–S19
Normal step		A normal step state, numbered as Sn. The "n" cannot be repeated. The address range of S soft element corresponding to a normal step symbol is S20--S991
Transfer		A transfer. It can be built-in with a transfer condition (an embedded LAD). You can compile the transfer condition so that the state of S element connected with this transfer is set when the condition is met, and enters the next step state. The transfer symbol needs to be used between steps
Jump		A jump symbol, used after the transfer symbol. It can set the specified S element to ON when the transfer conditions are met. It is used to cycle or jump among the step state.
Reset		A reset symbol, used after the transfer symbol. It can set the specified S element to OFF when the transfer conditions are met. It is used to end the SFC program.
Selection branch		Multiple independent transfer conditions, used after a step symbol. When the transfer condition of one branch is met, the last step state ends, and the next step state of the corresponding branch starts.It is used to select one of multiple step branches. After selecting one branch, no other branches can be selected.
Selection merge		A merge of step branches, connected at the junction of the selection branches. When the transfer condition of one branch is met, the last step state ends, and the next step state takes effect.
Parallel branch		Connected after a step symbol, and the subsequent parallel branches share the same transfer condition. When the transfer condition is established, the subsequent parallel branches are validated and executed at the same time.
Parallel merge		A merge of parallel branches, connected at the junction of the parallel branches. When the multiple parallel step branches are executed and the transfer conditions are met, the next step state takes effect.
LAD block		The LAD block is used to represent LAD instructions other than the SFC flow. It can be used to start the initial step and other general operations.

- Usage of programming symbols

1. An initial step symbol can be used alone. It cannot be preceded by other symbols, and can be followed only by the transfer symbols.

2. A LAD block cannot be connected with other symbols.

3. A normal step symbol can only be connected with a transfer symbol. The normal step symbol cannot exist alone in the diagram.

4. The reset and jump symbols need to be preceded by transfer symbols and followed by nothing.

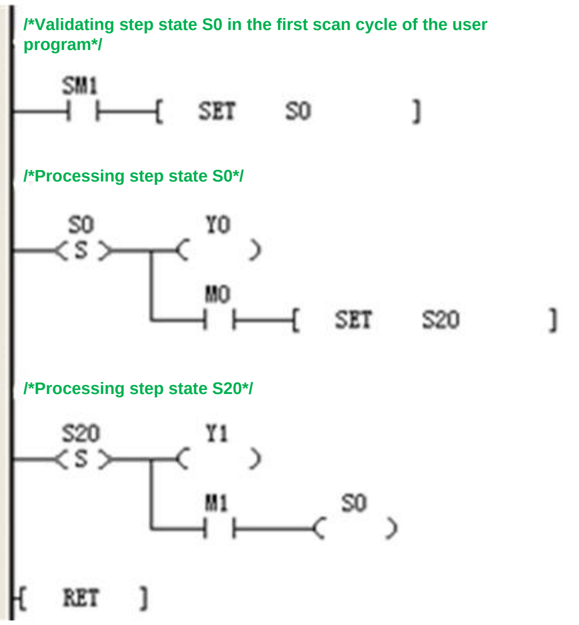
5. Neither a transfer symbol nor a jump symbol can exist alone in a program.

7.1.5 SFC program structure

The process structure of a SFC is classified into three types: simple sequential structure, selection branch structure and parallel branch structure. Besides, the jump structure is also a special form of the selection structure.

- Simple sequential structure

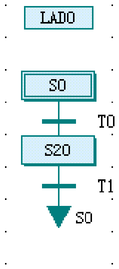
The following figure shows a simple structured SFC program and its LAD counterpart.



/*Validating step state S0 in the first scan cycle of the user program*/

/*Processing step state S0*/

/*Processing step state S20*/



In a simple structured SFC program, when the step transfer conditions are met, the program is sequentially transferred from the current step state to the next step state without any branch structure.At the last step, when the transfer conditions are met, the SFC program section either ends or transfers to the initial step state.

1. LAD block

The LAD block is used to start the SFC program section. To be specific, setting the S element of the initial step symbol to ON. In the preceding figure, the program uses the power-on startup mode.

The LAD block can also be used in other general program sections except the SFC program.

2. Initial step state

As shown in above figure, the initial step state is started by a LAD block. The range of S elements for initialstep stateis 0﹣19.

3. Normal step symbol

The normal step symbol is used for programming in sequential processes. The range of S elements for normalstep state is 20﹣1023.

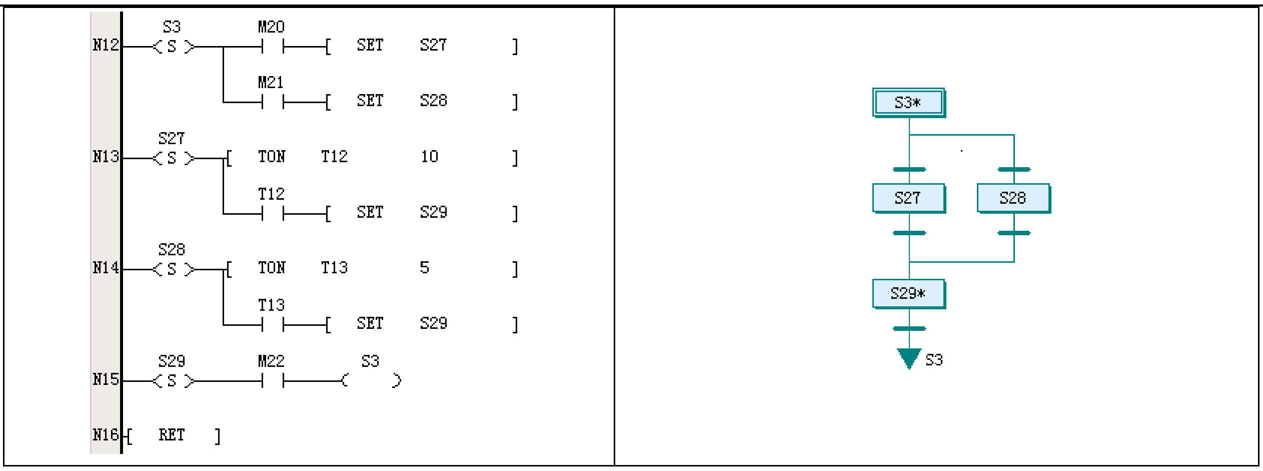
4. Transfer or reset

The last transfer symbol of the program shown in the above figure is connected with a jump symbol, which leads the program to jump to the initial step state. This is a cyclic program.

However, the last transfer symbol of the program can be connected with a reset symbol, which can reset the last step state. After the reset, the program ends, andwaits for the next round of execution.

- Selection branch structure

The following figure shows a selection branch structure, with LAD on the left and SFC on the right.



1. Selection branch

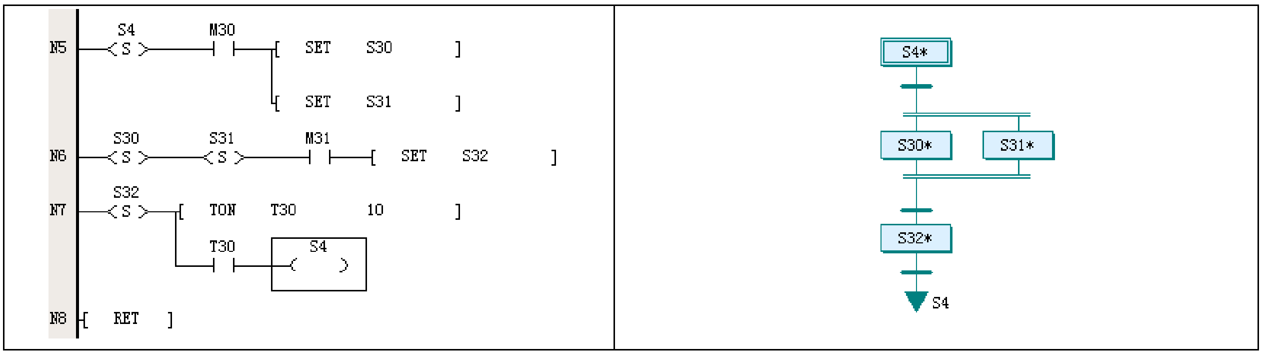
The step state of a branch is validated when its corresponding transfer conditions are met. You need to ensure that the transfer conditions of different branches are all exclusive, so as to make sure that each time only one branch is selected.As shown in the preceding figure, S27 and S28 in row N12 of LAD program are transferred from conditions M20 and M21 respectively. The conditions M20 and M21 must not be set at the same time in order to ensure that S27 and S28 are not be selected at the same time.

2. Selection merge

The selection merge is the structure where all selection branches merge to the same step state. The transfer conditions areset respectively. As shown in the preceding figure, the transfer condition in the step state S27 in row N13 is that time is up for T12,while that in the step state S28 in row N14 is that time is up for T13. However, the results are the same: step state S29 starts.

- Parallel branch structure

The following figure shows a parallel branch structure, with LAD on the left and SFC on the right.



1. Parallel branch

When the transfer conditions of the parallel branch structure are met, each step state connected to the parallel branch structure is simultaneously activated. This is also a common sequential control structure, which enables the PLC to process multiple procedures at the same time. As shownin row N5 program of the preceding figure, S30 and S31 are validated at the same time when transfer conditionM30 is met and M30 is set.

2. Parallel merge

When the transfer conditions of the parallel merge structure are met, each step state connected to the parallel merge structure becomes invalid at the same time. As shownin row N6 program of the preceding figure, when the program is running both S30 and S31 at the same time, and M31 is set, the program starts S32 and ends S30 and S31.

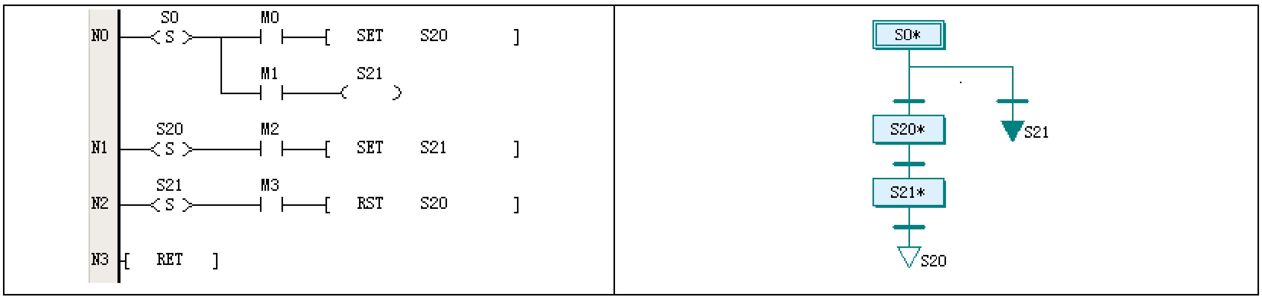
The transfer condition for the parallel mergeis that the transfer action can be executed only when all individual steps are finished before merging.

- Jump structure

The jump structure is often used for the following purposes: to omit certain step states, to cyclically return to the initial step state or the normal step state, andto jump to other processes.

1. Omitting certain step states

In a process, when sequential execution is not required under certain transfer conditions, the program can jumps directly to the needed step state and omits the unnecessary step states, as shown in the following figure, with LAD on and left and SFC on theright.



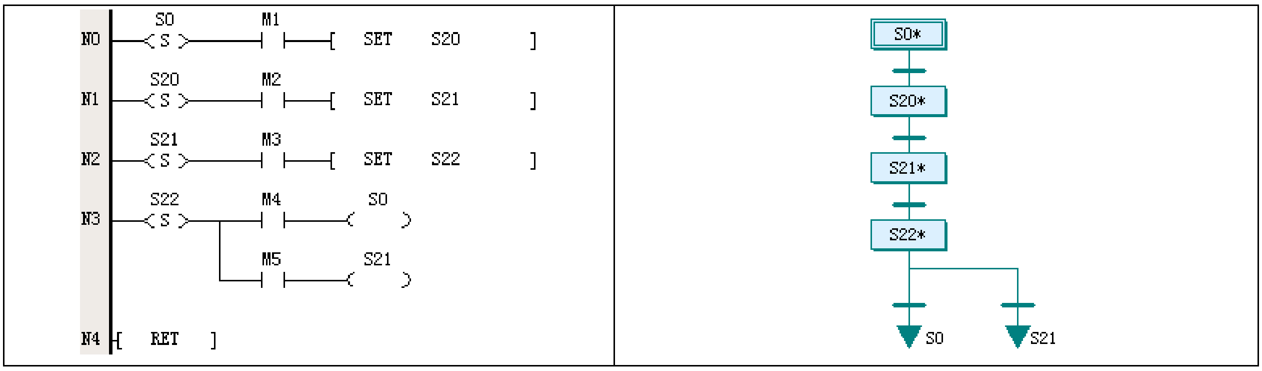
In the SFC program shown in the preceding figure, the jump symbol S21 is used as the jump, while the step state S20 is skipped. In fact, a branch structure is selected before the jump is performed.

In the LAD, the second branch in row N0 is a jump instruction, which adopts the form of an OUT coil instead of an form of the SET instruction that issequentially transferred. When running in the step state S0 and M1 is ON, the program jumps to the step state S21.

2. Cycling

In a process, when it is necessary to cycle a part or all of the step states under certain transfer conditions, you can use a jump symbol to implement the cycle function. At the end of this process, a part of the step states can be cycled if the program jumps to the previous normal step symbol, or all step states can be cycled if the program jumps to thei nitial step symbol.

The following figure shows a program that realizes the above two cycle structures at the same time, with LAD on the left and SFC on theright.



In the SFC, when the step state S22 is valid and one of the transfer conditions is met, the program jumps to S21 and re-run the step state S21. Under another transfer condition, the program jumps to the initial step state S0 and re-run allstep states.

In LAD, these two kinds of jumps are realized in row N3, where you can see the OUT coil of the jump instruction.

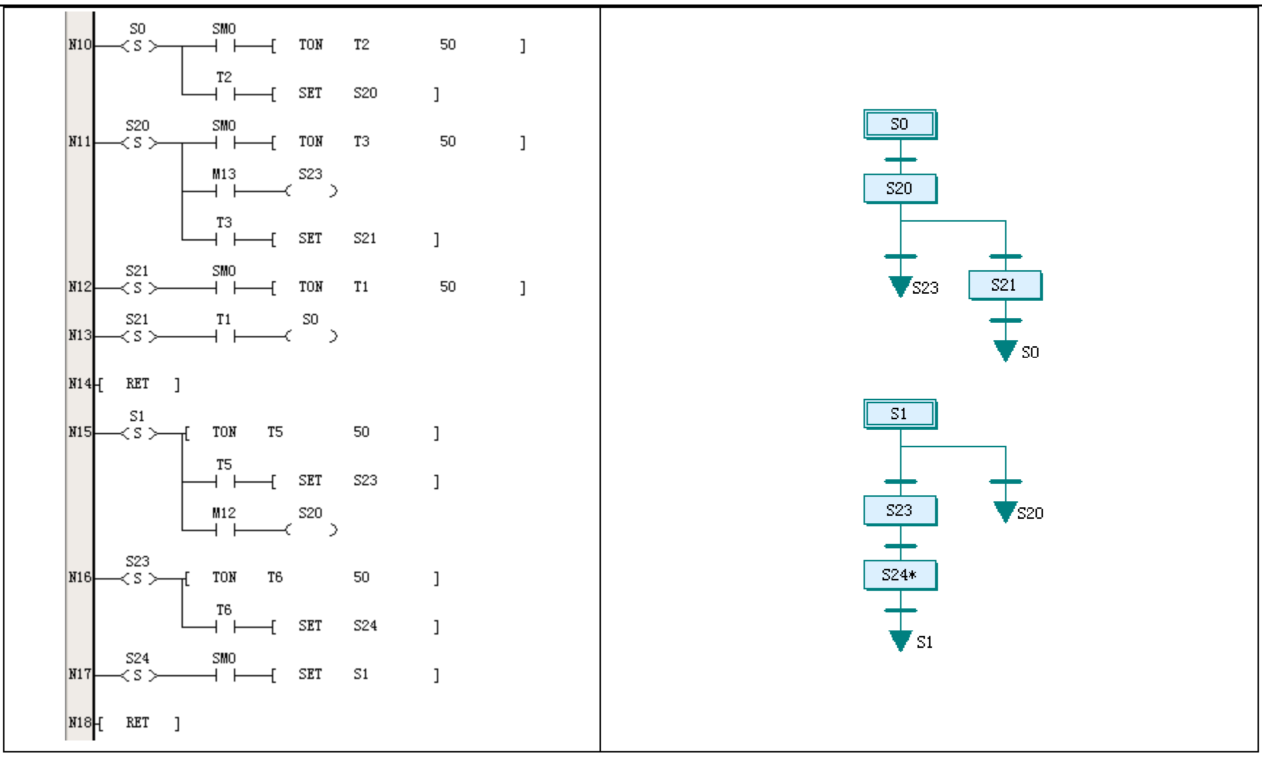
3. Jumping among different independent processes

The SFC of VC series PLCs support multiple independent processes at the same time, and jumping among these processes is enabled. You can set certain transfer conditions in an independent process, and when the conditions are met, it is directly transferred to another independent process, or it jumps to a random step state (initial ornormal) of another independent process.

Note

Jumping among multiple independent processes complicates the PLC program. You need to use it with prudence.

The following figure shows a program that realizes a jump from one independent process to another, with LAD on the left and SFC on the right.


N10 S0 S SM0 TON T2 50
T2 SET S20
N11 S20 S SM0 TON T3 50
M13 S23
T3 SET S21
N12 S21 S SM0 TON T1 50
N13 S21 S T1 S0
N14 RET
N15 S1 S TON T5 50
T5 SET S23
M12 S20
N16 S23 S TON T6 50
T6 SET S24
N17 S24 S SM0 SET S1
N18 RET
S0
S20
S23
S21
S0
S1
S23
S20
S24*
S1

In the SFC, when the step state S0 in the first process is valid, the program can jump to the step state S23 in the second process

under certain transfer conditions; while the step state S1 in the second process is valid, the program can also jump to the step state S20 in the first process under certain transfer conditions.

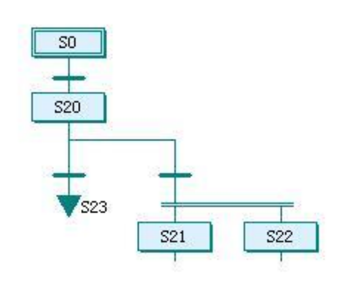
As shown in the preceding figure, the jump is based on a selection branch structure. When the program jumps to another process, all step states in the original process become invalid. As shown in the example, if the program jumps to the step state S23 in the second process from the step state S20 in the first process, the S20 is set to OFF and all step states (S0, S20, and S21) in thefirst process become invalid (OFF).

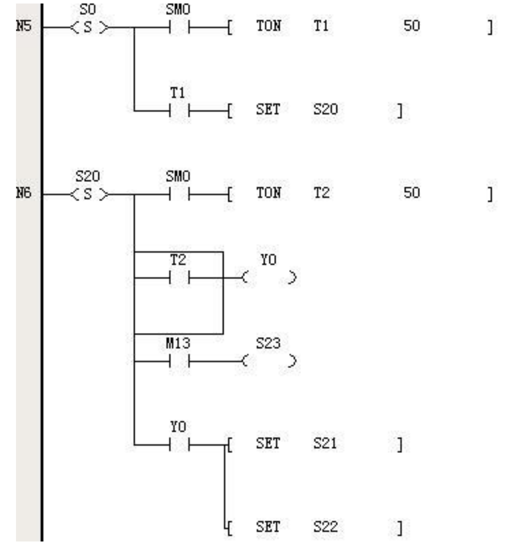
7.1.6 Execution of SFC program

The similarity between the execution of a SFC program and that of a LAD program is that they both carry out the continuously cyclic scanning from up to down and from left to right.

On the other hand, their difference lies in: in a SFC program, the step states' validity can change according to the sequence conditions, and only internal instruction sequences in the valid step states can be scanned and executed. While in the main program of a normal LAD, all programs are scanned and executed in each scan cycle.

As shown in the following figure, the program on the right is the LAD counterpart of the SFC program on the left. When the step state S20 is valid, the T2 timer is scanned and starts timing. The step states S21 and S22 are not be executed before T2 timer reaches the preset value, and S23 is not executed when M13 is OFF.





The state of S elements is switch between ON and OFF according to the transfer conditions, thus making the program transfer from one step state to another. When a S element changes from ON to OFF, the output elements of the corresponding step state are reset or cleared. For details, refer to section 5.3.1 "STL: SFC state loading instruction".

Note

1. The SFC program of VC series PLCs usually contains SFC and LAD blocks. The LAD blocks are used to handle operations outside the process, including starting the SFC. The LAD blocks are not controlled by the S elements, and the program rows of these LAD blocks scanned by the PLCs are executed in every scan cycle.

2. The state change of the S element affects the built-in instructions of the corresponding step states, and the switch-over between two step states takes some time, so it is necessary to observe certain rules on the operation of certain soft elements and usage of the instructions during the SFC programming.

7.2 Relationship between SFC and LAD

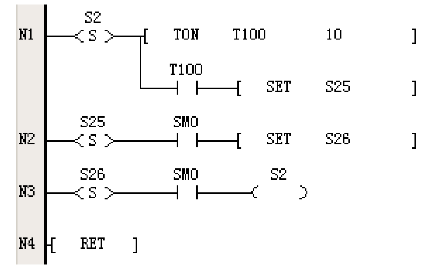
A SFC program can take the form of a LAD, which can help you understand the practical significance of the SFC program structure.

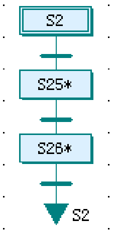
In a LAD, the SFC symbols are replaced with various corresponding SFC instructions, while the corresponding processes are represented by various structures.

7.2.1 STL instruction and step states

In a LAD, a step state is started by a STL instruction.All step states are represented by S elements.

The following diagram on the left is a LAD program of a simple sequential structure, and the diagram on the right is it scorresponding SFC program.





As shown in the LAD program, the step state S2 starts with a STL instruction, and the following TON instruction is the internal instruction sequence of S2. An internal instruction of the step state can be made up of multiple statements, which is actually a relatively complete program section, and almost consistent with the normal LAD counterpart.

The difference between an initial step state and a normal step state only lies in the range in which the S elements are selected.

For details about the STL instruction, refer to section 5.3.1 "STL: SFC state loadinginstruction". You need to note that when the step state changes from ON to OFF, the destination operands corresponding to the built-in instructions (OUT, TON, TOF, PWM, HCNT, PLSY, PLSR, DHSCS, SPD, DHSCI, DHSCR, DHSZ, DHST, DHSP, and BOUT)are cleared.

📖 Note

Because the PLCs run in continuous scan cycles, after a step state transfer, those built-in statements of the original step state are not be affected by the transfer of ON to OFF until the next scan. For details, refer to section 7.4.1 "Common programming errors".

7.2.2 SET instruction

As shown in the preceding diagram, the transfer symbols in the SFC program on the right are realized through the SFC state transfer instruction.

The transfer conditions consist of the NO contacts before the SET statements. The NO contacts are controlled by built-in instruction statements or through the external operations.

Setting the specified step state to be valid when the energy flow of the SFC state transfer instructionis valid, and setting the current valid step state to be invalid to complete the state transition.

7.2.3 RET instruction and SFC program segment

As shown in the preceding diagram, the SFC program on the right starts with a S2 initial step symbol, and returns to S2 after passing two ordinary step symbols. While in the LAD, the end of a segment of the SFC program needs to be marked with the RET instruction.

The RET instruction can be used only in the main program.

7.2.4 OUT instruction and RST instruction

As shown in the preceding diagram, the jump to S2 is realized in LAD by the N3 row, which uses an OUT instruction. The destination operand of the OUT(jump) instruction can be in any independent procedure.

If the reset symbol S26 is used, line N3 in the LAD is a RST instruction, which can reset the last step state S26.

7.2.5 SFC selection branch, parallel branch and merge

For details about LAD counter part of SFC selection branches, refer to selection branch structure of section 7.1.5 "SFC program structure".

For details about LAD counterpart of SFC parallel branches, refer to parallel branch structure of section 7.1.5 "SFC program structure".

7.3 How to program With SFC

1. Analyzing the work flow and determining the program structure

The structure of a SFC program is classified into three types: simple sequential structure, selection branch structure and parallel branch structure. Besides, the jump structure is also a special form of the selection branch structure.To program with SFC, the first step to do is to determine the structure of the flow. For example, a single object passing through a sequential flow is a simple sequential structure. Multiple objects with different parameters to be processed asynchronously needs a selection branch structure. While a cooperation of multiple independent mechanical elements may need a parallel branch structure.

2. Determining the major procedures and transfer conditions to draw a draft of the flow chart

After determining the structure, you need to figure out the major procedures and transfer conditions. By dividing the work flow into smaller operation stages, you can get the procedures. Ending each procedure with a transfer condition,and then you can get a draft of the work flow.

3. Making a SFC program according to a draft of the flow chart

Using the SFC programming language in Auto Studio to make a SFC program out of a draft of the flow chart. By now you have got an executable PLC program, but the program needs to be improved.

4. Making a list of I/O pointsand determining the object of each procedure and actual transfer conditions

Generally, the input points are mostly transfer conditions while the output points are mostly operation objects. In addition, with the list, you can further modify the SFC.

5. Inputting the steps and transfer conditions

In the SFC programming interface, you can right click a SFC symbol and select **Embedded LAD** item in the shortcut menu. Then you are able to open the built-in LAD editing workspace of this symbol, and input the LAD programs and transfer condition.

6. Programming the LAD program blocks

You need to remember to program some program blocks that provide general functions, such as start, stop and alarm functions of the sequence process. Such program blocks need to be placed in LAD blocks.

Note

The start and stop operations are crucial for personal and equipment safety. Considering the special features of SFC program, you need to make sure that all outputs that should be stopped are shut down when the PLCs are stopped.

7.4 Notesin SFC programming

The STL statement has some special characteristics, and the PLCs periodically scan the instruction statements by their display order.Because of these reasons, there are several important notes in SFC programming.

7.4.1 Common programming errors

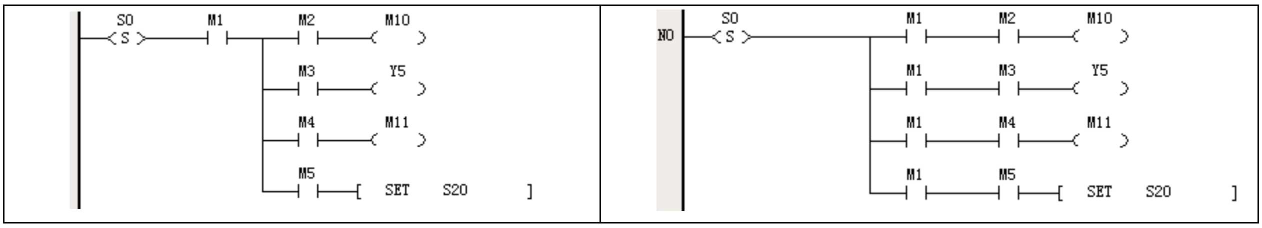
1. Reusing the stepstate symbols

In the same PLC program, each step state symbol used for sequential control programming corresponds to a unique S element and cannot be reused.

You need to note this requirement when editing a SFC program using the LAD editor.

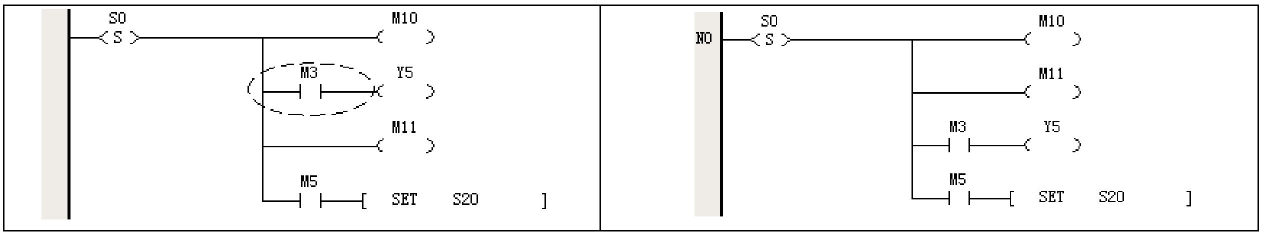
2. Setting branches after a transfer condition

Setting conditioned branches after a transfer condition is disabled in SFC programming, as shown in the left diagram below. Because M1 has become a transfer condition and cannot be branched. Instead, you can change it into the right diagram below, which can be compiled correctly.



3. Connecting output coils to internal bus after a NC or NO contact instruction

When a NO or NC contact instruction is used in a branch, the output coils in the subsequent branches are prohibited to connect the internal bus, as shown in the left diagram below. Instead, you can change the branch order into the one shown in the right diagram below, which can be compiled correctly.

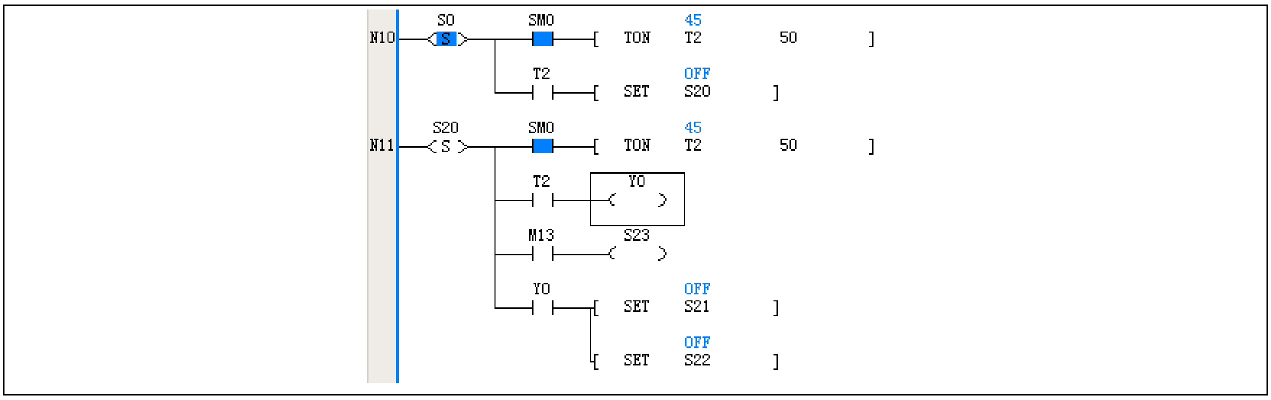


4. Reusing the same soft element in the neighboring step states

When the PLCs execute the program, the instructions are cyclically scanned by their display orders. When the program shifts from the previous step state to the next step state, the sequence of instructions in the previous step state just ends the scan, and the sequence of instructions in the next step state has also been turned on to form a control output.

Therefore, when the STL instruction is turned from OFF to OFF, certain internal elements of the instruction are reset (For details, refer to section 5.3.1 "STL: SFC state loading instruction"), but the reset can only be carried out during the next scan cycle. That is to say, at the moment of transferring the step state, the internal elements of the last step state retains their original data and states until the step state is scanned in the next cycle.

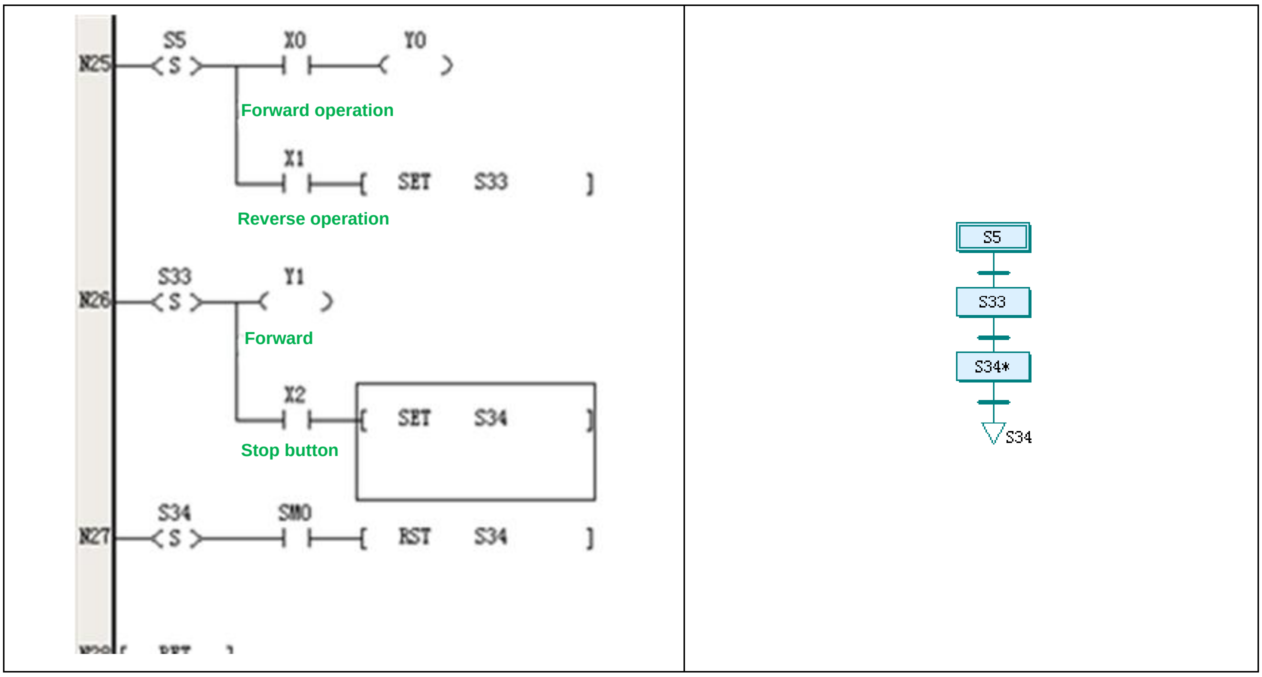
As shown in the following diagram, the two neighboring step states use the same timer: T2. When the step state is shifted from S0 to S20,the T2 element keeps its timing value and state, rendering the step state S20 unable to perform the timing operation as it is originally designed by the user. The program will jump directly to S21 and S22. Therefore, you need to note that reusing soft elements in a program is not prohibited, but it is best not to reuse them in the neighboring step states, otherwise unexpected results may occur.



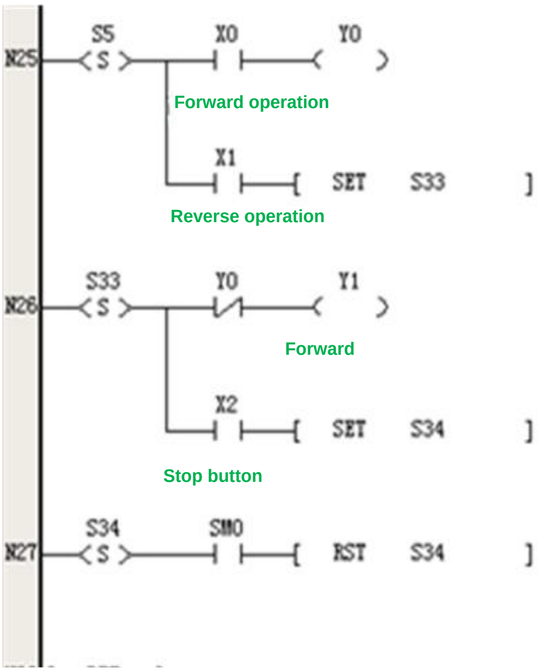
5. Failing to interlock the soft elements

During the SFC programming, certain soft elements may become contradictory to each other under some special transfer conditions of the step states. At this time, inter-locking isnecessary.

Taking the following forward and backward operation program as an example, where Y0 and Y1 are respectively forward and backward control output. X0 is forward operation, X1 is backward operation, and X2 is stop button. It is required that Y0 and Y1 are interlocked, that is to say, they cannot be ON at the same time.However, in this example, when the device runs forward, and X1 is turned ON to shift the step state from S5 to S33, Y0 and Y1 are both ON in one program scan cycle.



Therefore, you need to add the interlock statements to the program by adding a Y0 NC contact before the Y1 output coil, as shown in the following diagram.

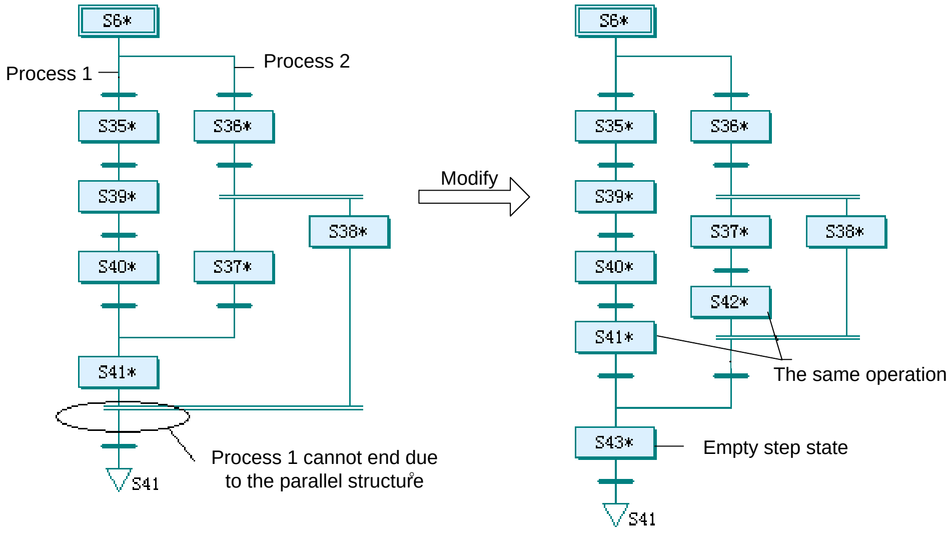

N25
S5
S
X0
Y0
Forward operation
X1
SET S33
Reverse operation
N26
S33
S
Y0
Y1
Forward
X2
SET S34
Stop button
N27
S34
S
SM0
RST S34

6. Confusing jumps with transfers

Jumps are mostly used between different processes and non-neigh boring step states, while transfers are used between the neighboring step states. It is prohibited to change an output coil into a SET instruction statement where a jump should be used, or change a SET instruction statement into an output coil where a transfer should be used.

7. Selecting a parallel branch structure as the branch transfer point, causing the failure to end the process

A branch is selected from multiple ones. If parallel branches are included, the process may never be ended, as shown in the following figure. In the program on the left,when process 1 runs to the step state S41, the transfer condition is a parallel branch, and therefore the system does not run process 2,causing the failure to implement the transfer condition, and thus causing a process error.


S6*
Process 1
Process 2
S35*
S36*
S39*
S38*
S40*
S37*
S41*
S41
Process 1 cannot end due to the parallel structure
Modify
S6*
S35*
S36*
S39*
S37*
S38*
S40*
S42*
S41*
The same operation
S43*
Empty step state
S41

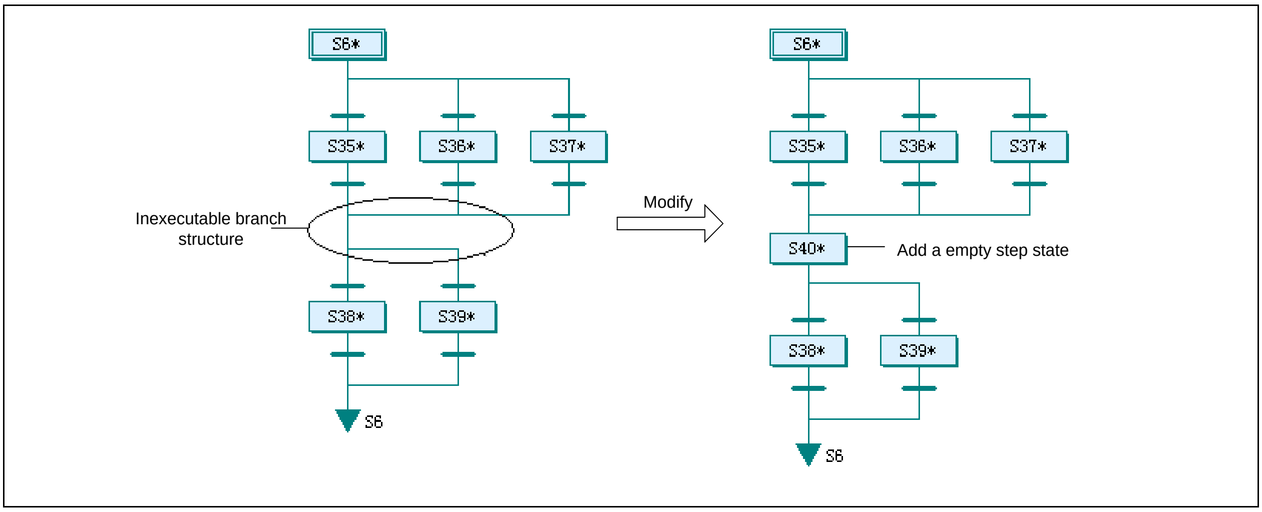
The modification method is shown in the program on the right, you need to add a step state S42 whose function is the same as S41. Then you need to add an empty step state S43 that serves as a programming structural element without actual function. The transfer conditions of S38, S41, and S43 need to be designed by the programmers, for example, all of them can adopt the transfer conditions of the original S41.

7.4.2 Programming tricks

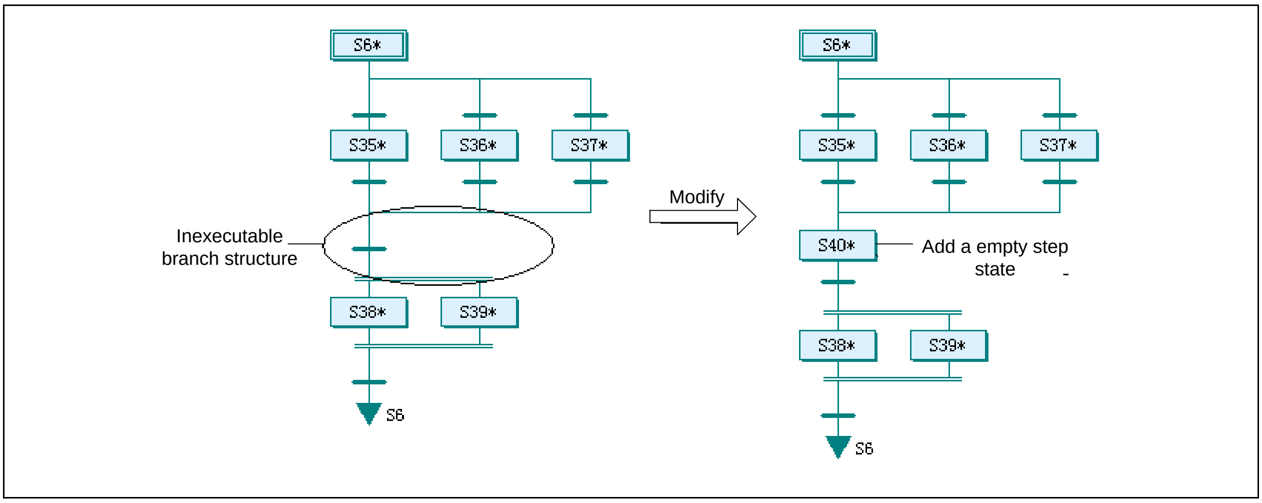
1. Making use of empty step states

You can use empty step states to deal with the branches with grammatical problems. The empty step states do not provide actual operation, but a necessary node in structure before the next transfer. You can see the following example.

In the left diagram below, the selection merge is connected immediately with another selection branch structure, which is prohibited. You can modify it by adding an empty step stateas shown in the right diagram.


S6*
S35*
S36*
S37*
Inexecutable branch structure
S38*
S39*
S6
Modify
S6*
S35*
S36*
S37*
S40*
Add a empty step state
S38*
S39*
S6

In the left diagram below, the selection merge is connected immediately with a parallel branch structure, which is prohibited too. You can also modify it by adding an empty step state as shown in the right diagram.

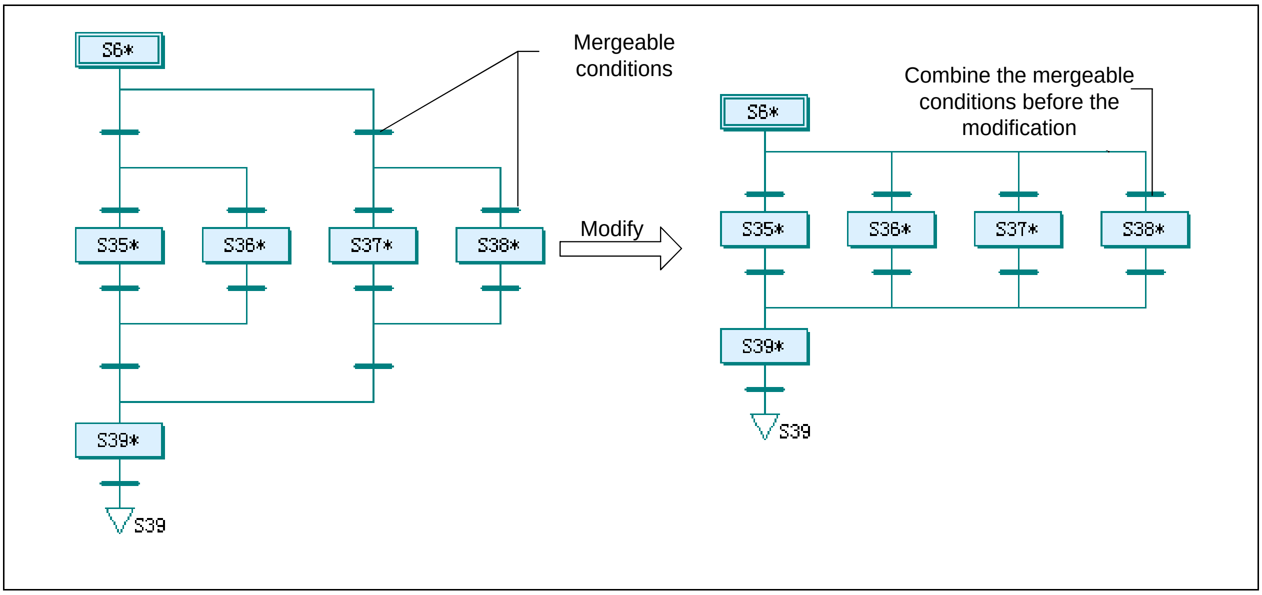

S6*
S35*
S36*
S37*
Inexecutable branch structure
S38*
S39*
S6
Modify
S6*
S35*
S36*
S37*
S40*
Add a empty step state
S38*
S39*
S6

You can address other tricky structures, such as parallel merge connected with parallel branches, or parallel branches connected with selection branches, by adding an empty step state.

2. Merging branches and transfer conditions

Some branches that seem complicated are actually caused by improper designand can be merged or simplified appropriately.

As shown in the following diagram, the designer set a selection branch first, and then set two selection branches. In fact, only one selection branch of four branches can achieve the same, and the two levels of transfer symbols of the original design can be merged into one.


S6*
Mergeable conditions
S35*
S36*
S37*
S38*
S39*
S39
Modify
Combine the mergeable conditions before the modification
S6*
S35*
S36*
S37*
S38*
S39*
S39

3. Using the function for saving data at power outage

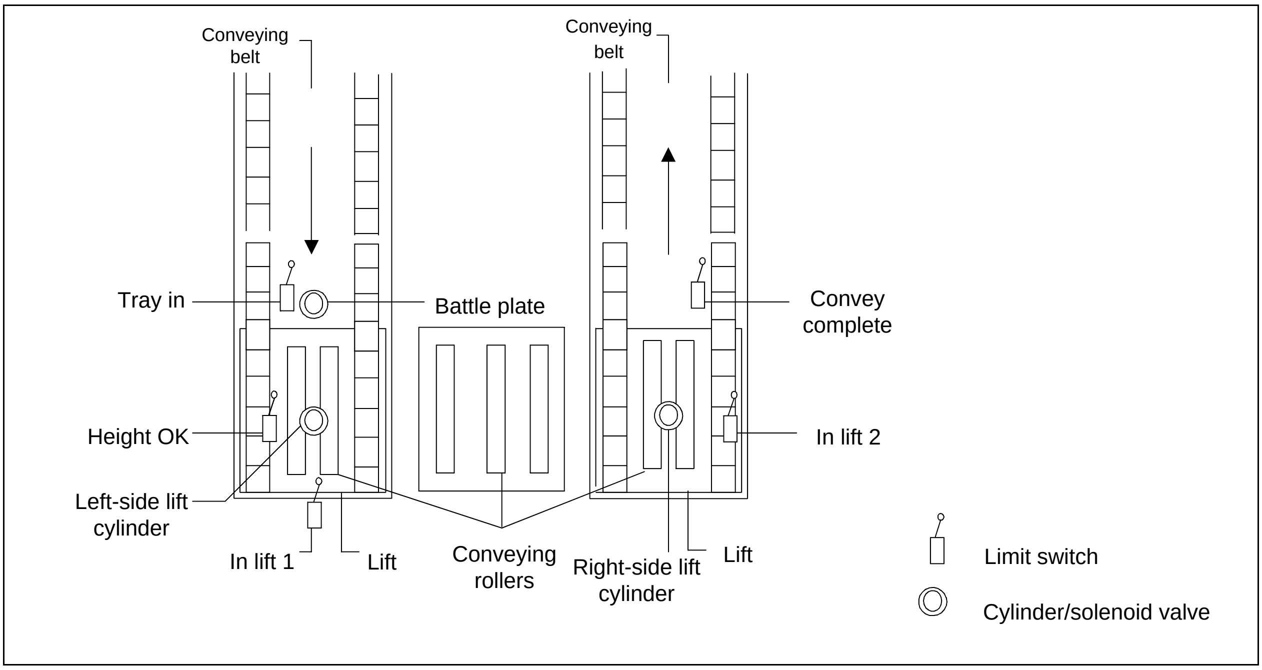
The values of S elements can be kept by the function for saving data at power outage. In this way, after the power on, the program can rerun from the step state when the power outage occurred.

7.5 Examples of SFC programming

The examples in this section are just illustrations of SFC programming, with simplified operations and conditions. The device configuration is conceptual and for study only. You cannot apply the example programs to actual use.

7.5.1 Simple sequential structure

The following example is a workpiece tray lifting and conveying machine. This machine uses cylinder lifting devices and conveying rollers to convey the workpiece tray from one conveying belt to another. The following figure shows a top view of the machine.



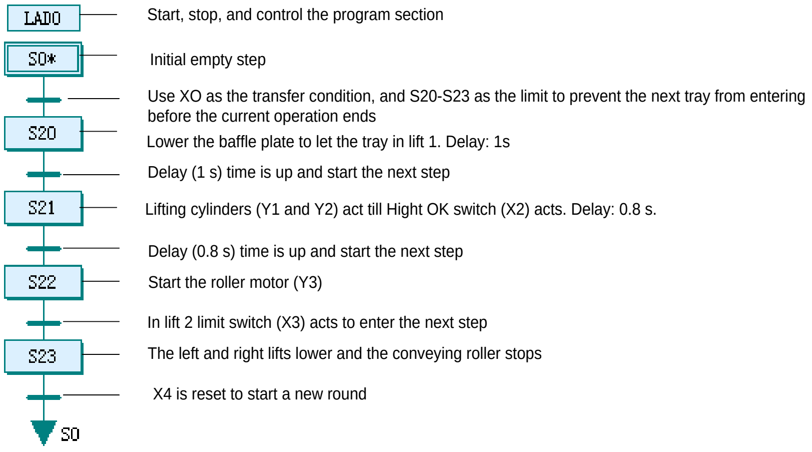
After the machine is started, the workpiece tray is conveyed along the convey or belt at the left sideto the entrance of the machine and triggers the "Tray in" limit switch. If no other tray is occupying the machine, the "Baffle plate" lowers down to let the work piecetray enter the machine. When the tray has completely entered the left lift, it triggers the "In lift 1" limit switch, the lift raises the tray until the "Height OK" limit switch is triggered. The rollers then act to convey the tray to the lift on the rightside until the "In lift 2" limit switch is triggered. The lift then lowers to put the tray to the conveying belt on the right.When the "Convey complete" limit switch is reset, a complete lifting and conveying process is over and the machine is ready for the next round.

The I/O points are listed in the following table.

SN	Address	Monitored objects		SN	Address	Monitored objects
1	X0	Tray in limit switch		8	Y0	Cylinder solenoid valve for the baffle plate
2	X1	In lift 1 limit switch		9	Y1	Cylinder solenoid valve for the left lift
3	X2	Height OK limit switch		10	Y2	Cylinder solenoid valve for the right lift
4	X3	In lift 2 limit switch		11	Y3	Conveying roller motor contactor
5	X4	Convey complete switch		12	Y4	Motor contactor for the left conveying belt
6	X5	Start switch		13	Y5	Motor contactor for the right conveying belt
7	X6	Auxiliary signal of emergency switch				

This is a simple sequential process. Each tray is conveyed in several successive procedures, with no other choices or parallel procedures, and no parallel flow. Writing the program with SFC would be faster and clearer than the conventional logic design method.

The following diagrams are the SFC program and its LAD counterpart.


LAD0
S0*
S20
S21
S22
S23
S0

Start, stop, and control the program section

Initial empty step

Use XO as the transfer condition, and S20-S23 as the limit to prevent the next tray from entering before the current operation ends

Lower the baffle plate to let the tray in lift 1. Delay: 1s

Delay (1 s) time is up and start the next step

Lifting cylinders (Y1 and Y2) act till Hight OK switch (X2) acts. Delay: 0.8 s.

Delay (0.8 s) time is up and start the next step

Start the roller motor (Y3)

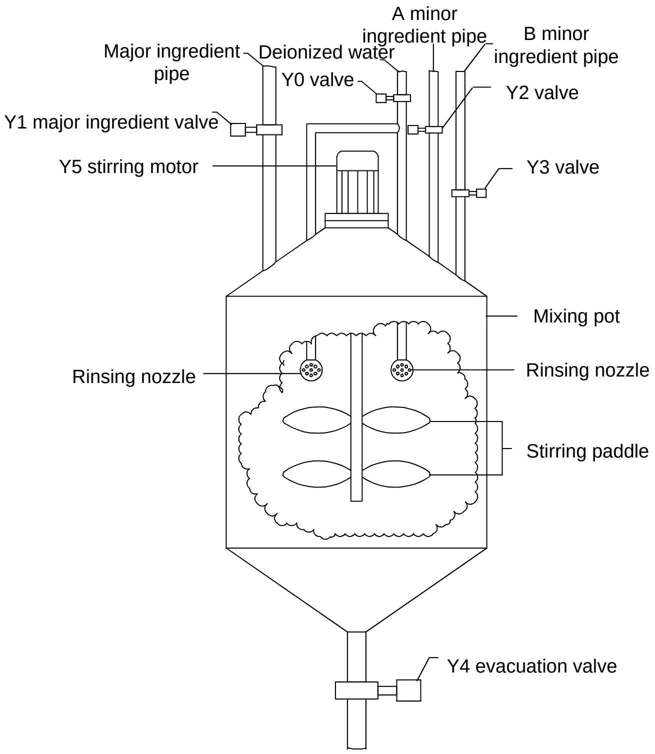
In lift 2 limit switch (X3) acts to enter the next step

The left and right lifts lower and the conveying roller stops

X4 is reset to start a new round

7.5.2 Selection branch structure

The following example is a material mixing process flow. Through this flow, two kinds of products, namely A and B, are produced. The following figure shows the illustration of the manufacturing device.


A minor ingredient pipe
B minor ingredient pipe
Major ingredient pipe
Deionized water
Y0 valve
Y2 valve
Y1 major ingredient valve
Y5 stirring motor
Y3 valve
Mixing pot
Rinsing nozzle
Rinsing nozzle
Stirring paddle
Y4 evacuation valve

To start the operation, the first step is to select the product type, A or B through the touch screen for the next batch of products, and then start production. The second step is to add the major ingredients until the weight of the added ingredient reaches 2000kg. The third step is to add minor ingredients, namely adding A minor ingredients for type A product or B minor ingredients for type B product until the added A or B minor ingredient reaches 500kg respectively. The forth step is to mix the ingredients for 20 minutes. The fifth step is to evacuate the materials until the remaining material is less than 20kg and the delay time is up. Once these steps are complete, the machine is ready for the next round.

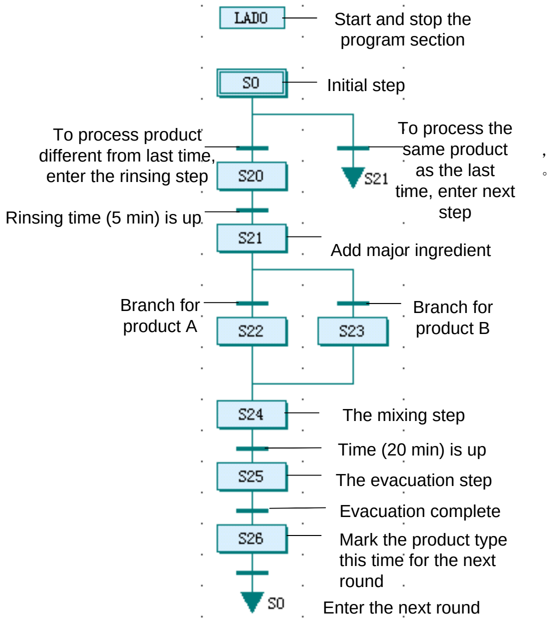
If it is the first time for the machine to start production, or the product type of the previous batch is different from what is going to be produced, you need to open the deionized water valve and evacuation valve to rinse the machine for 5 minutes before adding the major ingredients.

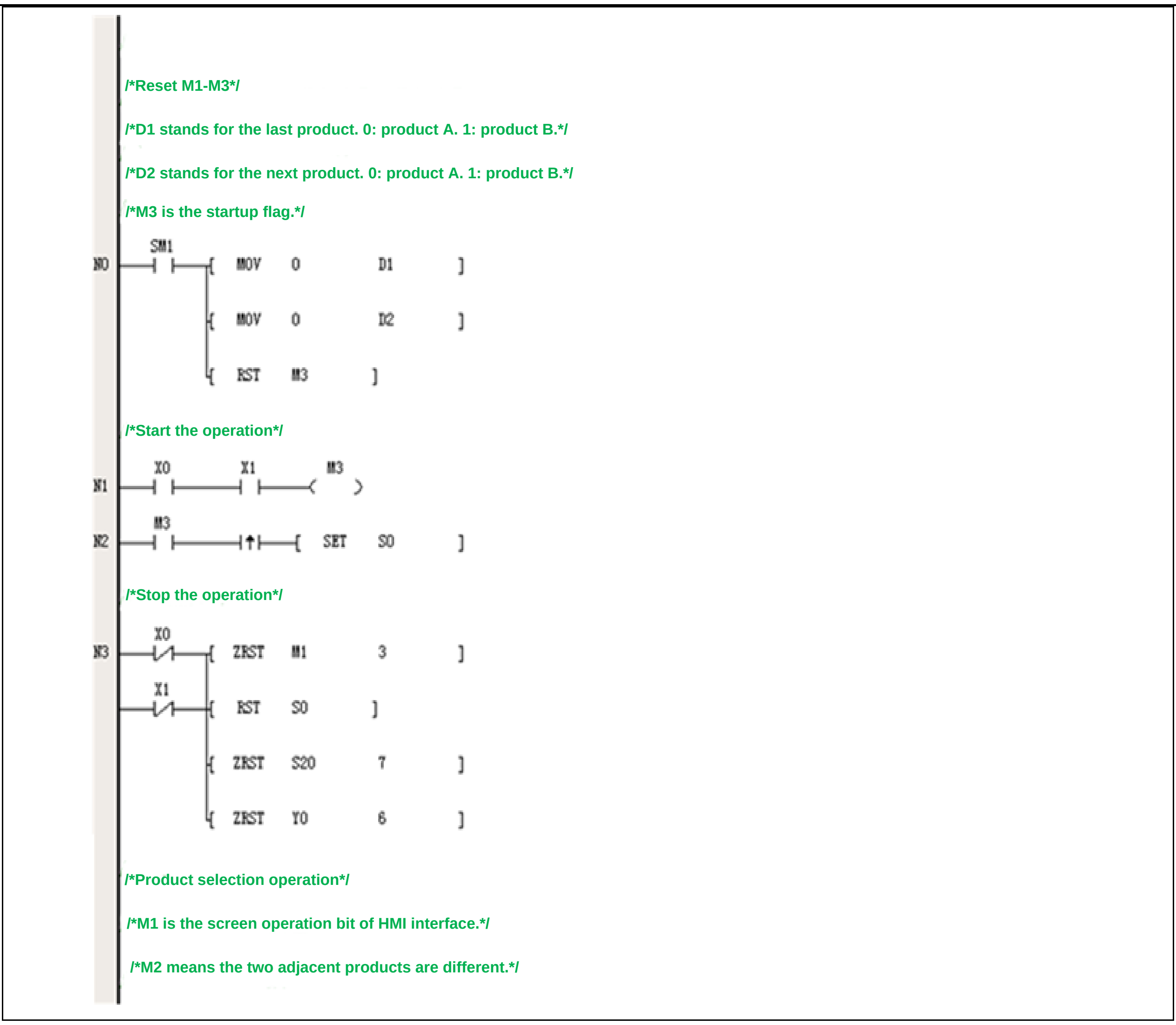
The I/O points are listed in the following table.

SN	Address	Monitored objects		SN	Address	Monitored objects
1	X0	Deionized water valve open		10	X11	Evacuation valve open
2	X1	Deionized water valve closed		11	X12	Evacuation valve closed
3	X2	Major ingredient valve open		12	Y0	Solenoid valve for deionized water
4	X3	Major ingredient valve closed		13	Y1	Solenoid valve for major ingredient
5	X4	Minor ingredient A valve open		14	Y2	Solenoid valve for minor ingredient A
6	X5	Minor ingredient A valve closed		15	Y3	Solenoid valve for minor ingredient B
7	X6	Minor ingredient B valve open		16	Y4	Solenoid valve for evacuation
8	X7	Minor ingredient B valve closed		17	Y5	Mixing motor contactor
9	X10	Mixing motor running				

Obviously this is a selection branch structured flow. You can select only one type of product, A or B, in a round of product production. You can switch the type of product only when a round of production is completed.Meanwhile, there is a structure for selection and jump in the process, namely the rinsing step.

The following diagrams are the SFC program and its LAD counter part.


LAD0
Start and stop the program section
S0
Initial step
To process product different from last time, enter the rinsing step
To process the same product as the last time, enter next step
S20
S21
Rinsing time (5 min) is up
S21
Add major ingredient
Branch for product A
Branch for product B
S22
S23
S24
The mixing step
Time (20 min) is up
S25
The evacuation step
Evacuation complete
S26
Mark the product type this time for the next round
S0
Enter the next round



/*Reset M1-M3*/

/*D1 stands for the last product. 0: product A. 1: product B.*/

/*D2 stands for the next product. 0: product A. 1: product B.*/

/*M3 is the startup flag.*/

```
     SM1
N0 ──┤ ├──┬─[ MOV   0     D1    ]
          ├─[ MOV   0     D2    ]
          └─[ RST   M3    ]
```

/*Start the operation*/

```
     X0        X1        M3
N1 ──┤ ├───────┤ ├───────( )
     M3
N2 ──┤ ├───────┤↑├──[ SET   S0    ]
```

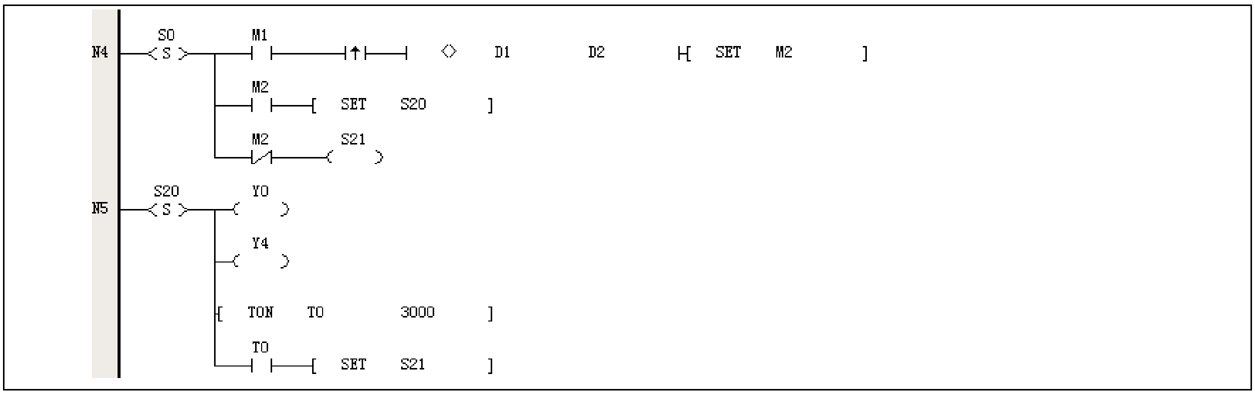
/*Stop the operation*/

```
     X0
N3 ──┤/├──┬─[ ZRST  M1    3     ]
     X1   │
   ──┤/├──┼─[ RST   S0    ]
          ├─[ ZRST  S20   7     ]
          └─[ ZRST  Y0    6     ]
```

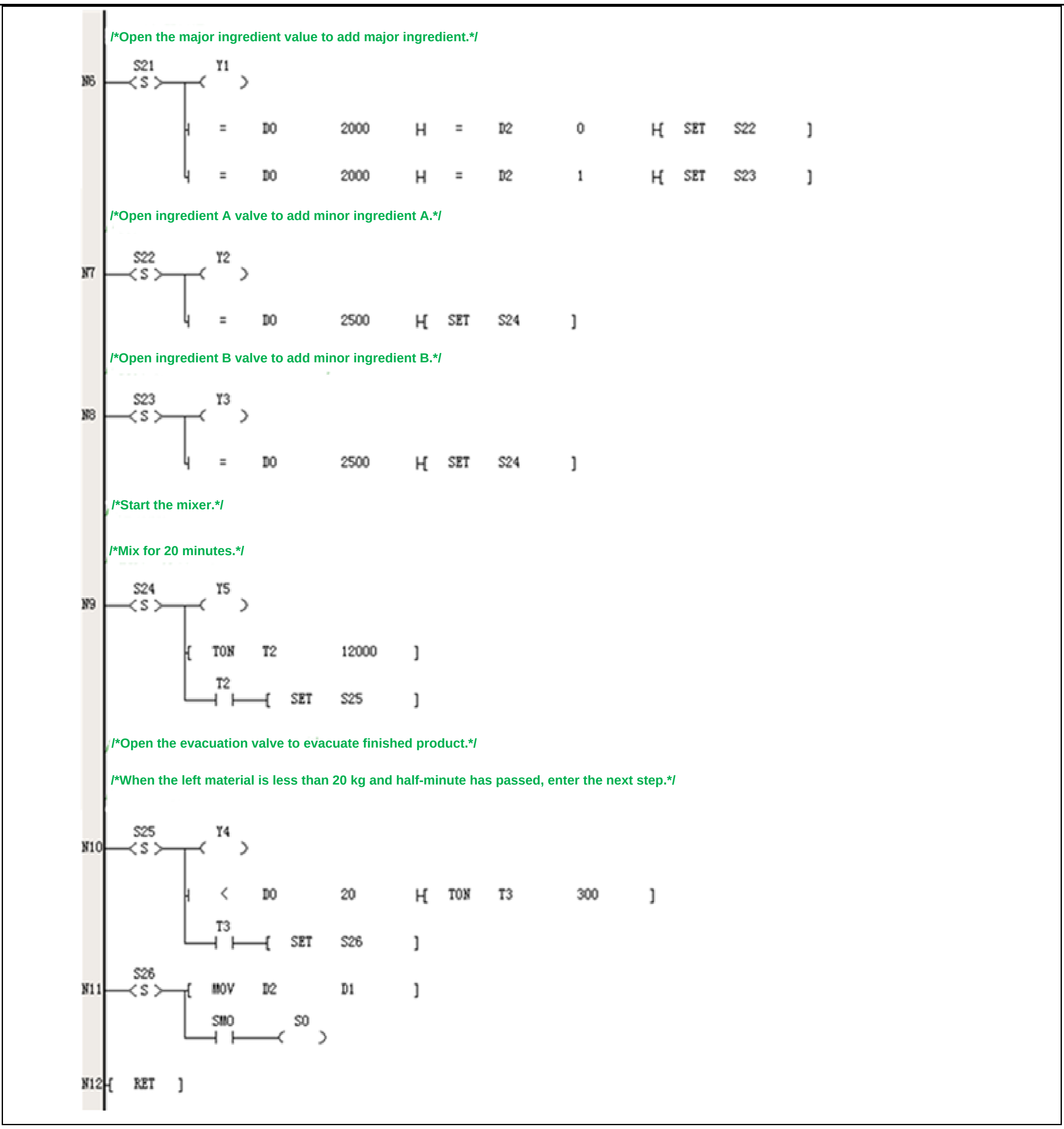
/*Product selection operation*/

/*M1 is the screen operation bit of HMI interface.*/

/*M2 means the two adjacent products are different.*/



```
     S0        M1
N4 ──<S>──┬────┤ ├─────────┤↑├───┤ <>    D1     D2    ├[ SET   M2    ]
          │    M2
          ├────┤ ├───[ SET   S20   ]
          │    M2        S21
          └────┤/├───────( )
     S20       Y0
N5 ──<S>──┬────( )
          │    Y4
          ├────( )
          ├─[ TON   T0    3000  ]
          │    T0
          └────┤ ├───[ SET   S21   ]
```



/*Open the major ingredient value to add major ingredient.*/

/*Open ingredient A valve to add minor ingredient A.*/

/*Open ingredient B valve to add minor ingredient B.*/

/*Start the mixer.*/

/*Mix for 20 minutes.*/

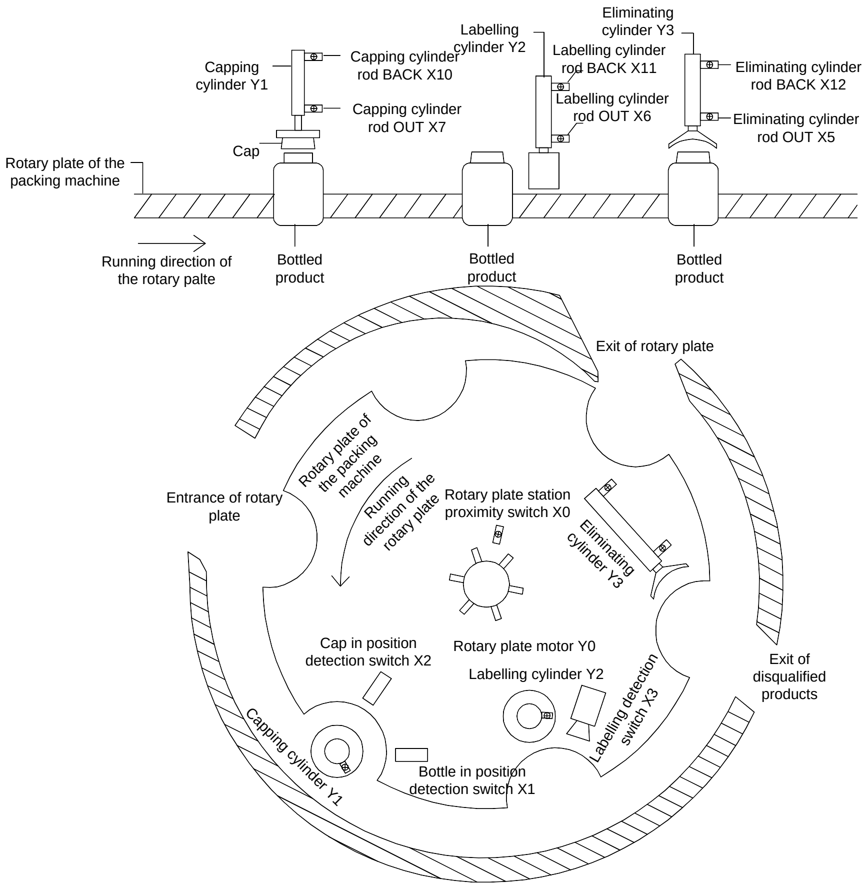
/*Open the evacuation valve to evacuate finished product.*/

/*When the left material is less than 20 kg and half-minute has passed, enter the next step.*/

N6
S21
S
Y1
= D0 2000
= D2 0
SET S22
= D0 2000
= D2 1
SET S23
N7
S22
S
Y2
= D0 2500
SET S24
N8
S23
S
Y3
= D0 2500
SET S24
N9
S24
S
Y5
TON T2 12000
T2
SET S25
N10
S25
S
Y4
< D0 20
TON T3 300
T3
SET S26
N11
S26
S
MOV D2 D1
SM0
S0
N12
RET

7.5.3 Parallel branch structure

The following example is a bottle packager. This packager seals the bottles and sticks labels to them. Meanwhile, it examines the bottle caps and labels, so that the flawed products are eliminated in the third procedure, while the qualified products continue to the next work flow.If no bottle is sent from the last work flow, the packager does not conduct any sealing or labeling. These three procedures are carried out at the same time, and each bottle moves from one position to another each time the rotary plate rotates.The following figure shows the illustration of the packager.


Eliminating cylinder Y3
Labelling cylinder Y2
Capping cylinder Y1
Capping cylinder rod BACK X10
Labelling cylinder rod BACK X11
Eliminating cylinder rod BACK X12
Capping cylinder rod OUT X7
Labelling cylinder rod OUT X6
Eliminating cylinder rod OUT X5
Cap
Rotary plate of the packing machine
Running direction of the rotary palte
Bottled product
Bottled product
Bottled product
Exit of rotary plate
Rotary plate of the packing machine
Running direction of the rotary plate
Entrance of rotary plate
Rotary plate station proximity switch X0
Eliminating cylinder Y3
Cap in position detection switch X2
Rotary plate motor Y0
Labelling cylinder Y2
Labelling detection switch X3
Exit of disqualified products
Capping cylinder Y1
Bottle in position detection switch X1

During the operation, the rotary plate rotates one step each time, which is detected by the X0 limit switch. The rotary plate stays at each step long enough for all the three procedures, driven by cylinders, finished. The cylinder rod OUT signal and cylinder rod BACK signal are monitored respectively.

The I/O points are listed in the following table.

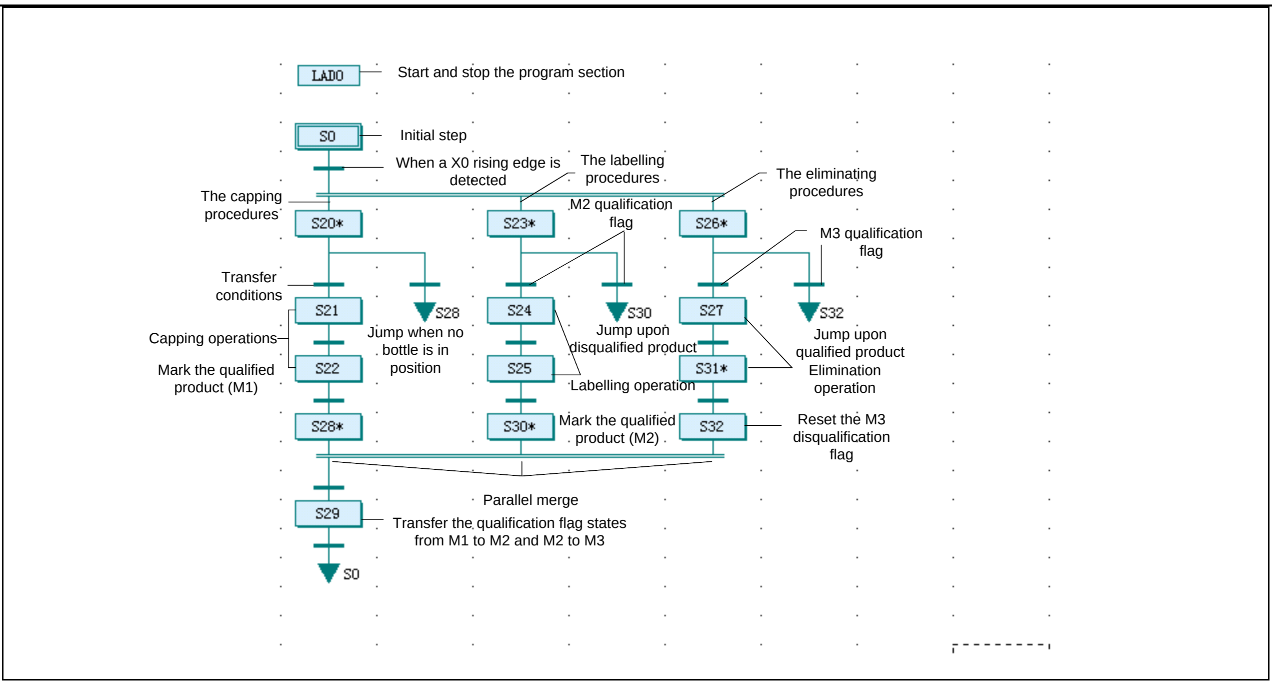
SN	Address	Monitored objects	SN	Address	Monitored objects
1	X0	Rotary plate station proximity switch	8	X10	Capping cylinder rod BACK
2	X1	Bottle in position detection switch	9	X11	Labelling cylinder rod BACK
3	X2	Cap in position detection switch	10	X12	Eliminating cylinder rod BACK
4	X3	Labelling detection switch	11	Y0	Rotary plate motor
5	X5	Eliminating cylinder rod OUT	12	Y1	Capping cylinder
6	X6	Labelling cylinder rod OUT	13	Y2	Labelling cylinder
7	X7	Capping cylinder rod OUT	14	Y3	Eliminating cylinder

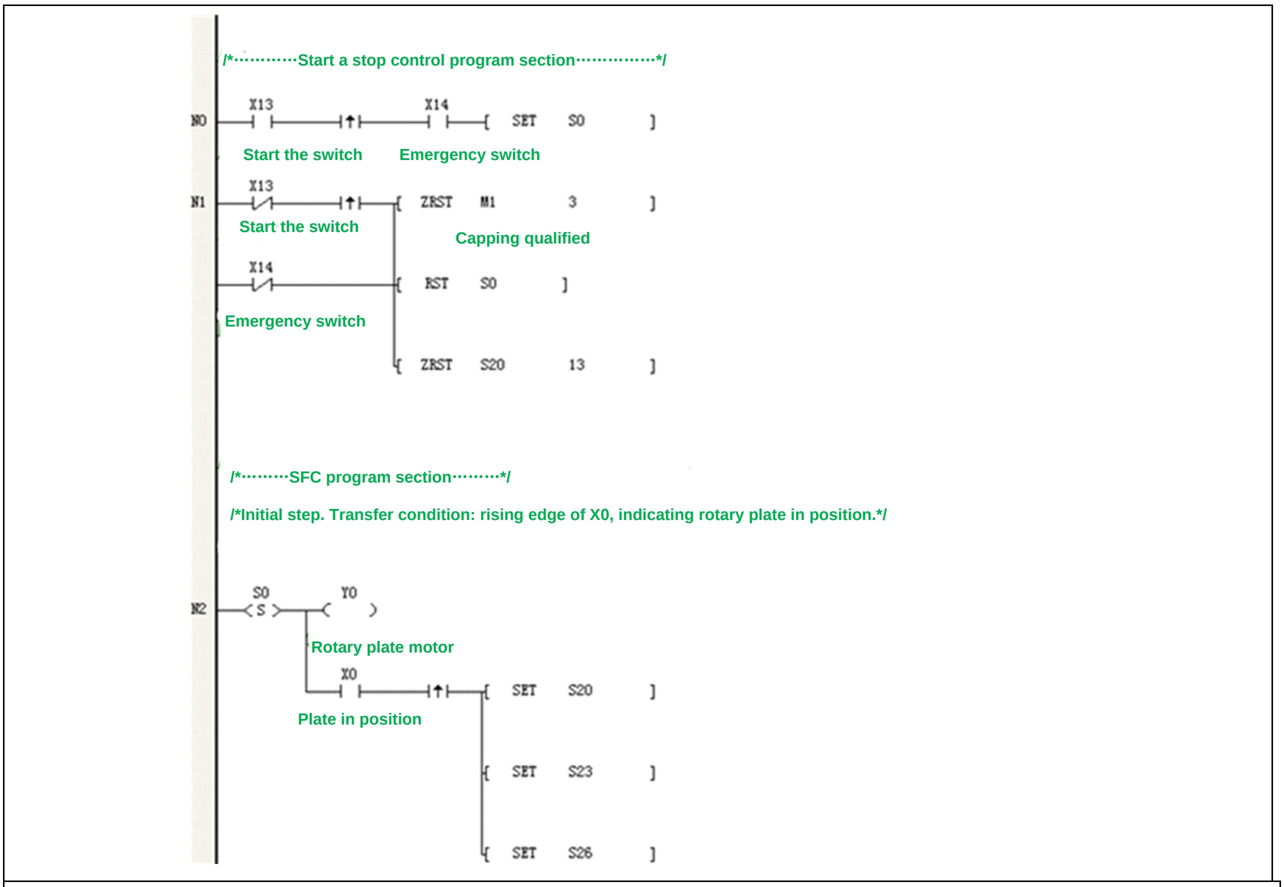
It is obvious that this is a parallel branch structured flow. After the rotating of the rotary plate, operations are performed in three stations. After all the operations are performed in the three stations, the device performs the subsequent operations.The following diagrams are the SFC program and its LAD counter part.

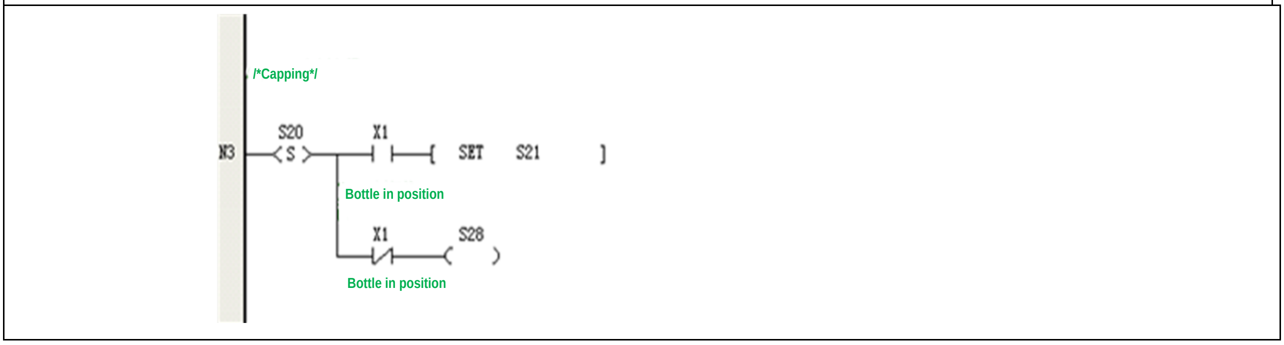
In the program, M1-M3 are the qualification flags for the procedures of capping, labeling and eliminating respectively.

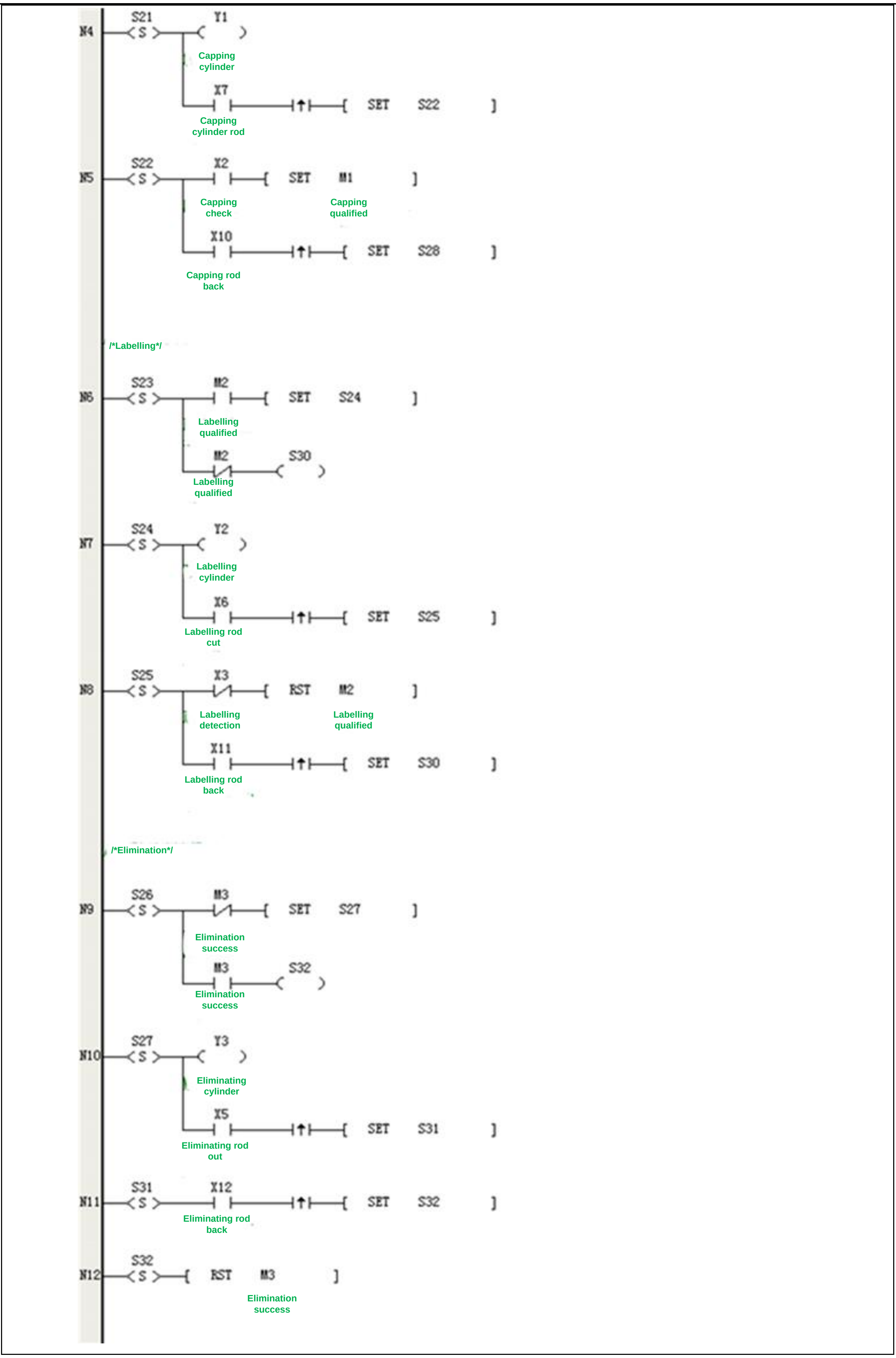
When the capping procedure runs to S22, X2 checks whether the capping is qualified or not. If yes, the corresponding qualification flag M1 is set. When the labeling procedure runs to S25, X3 checks whether the labeling is qualified or not. If not, M2 is reset. After all the procedures are complete, at step S29, the M2 state is transferred to M3, and M1 state is transferred to M2.

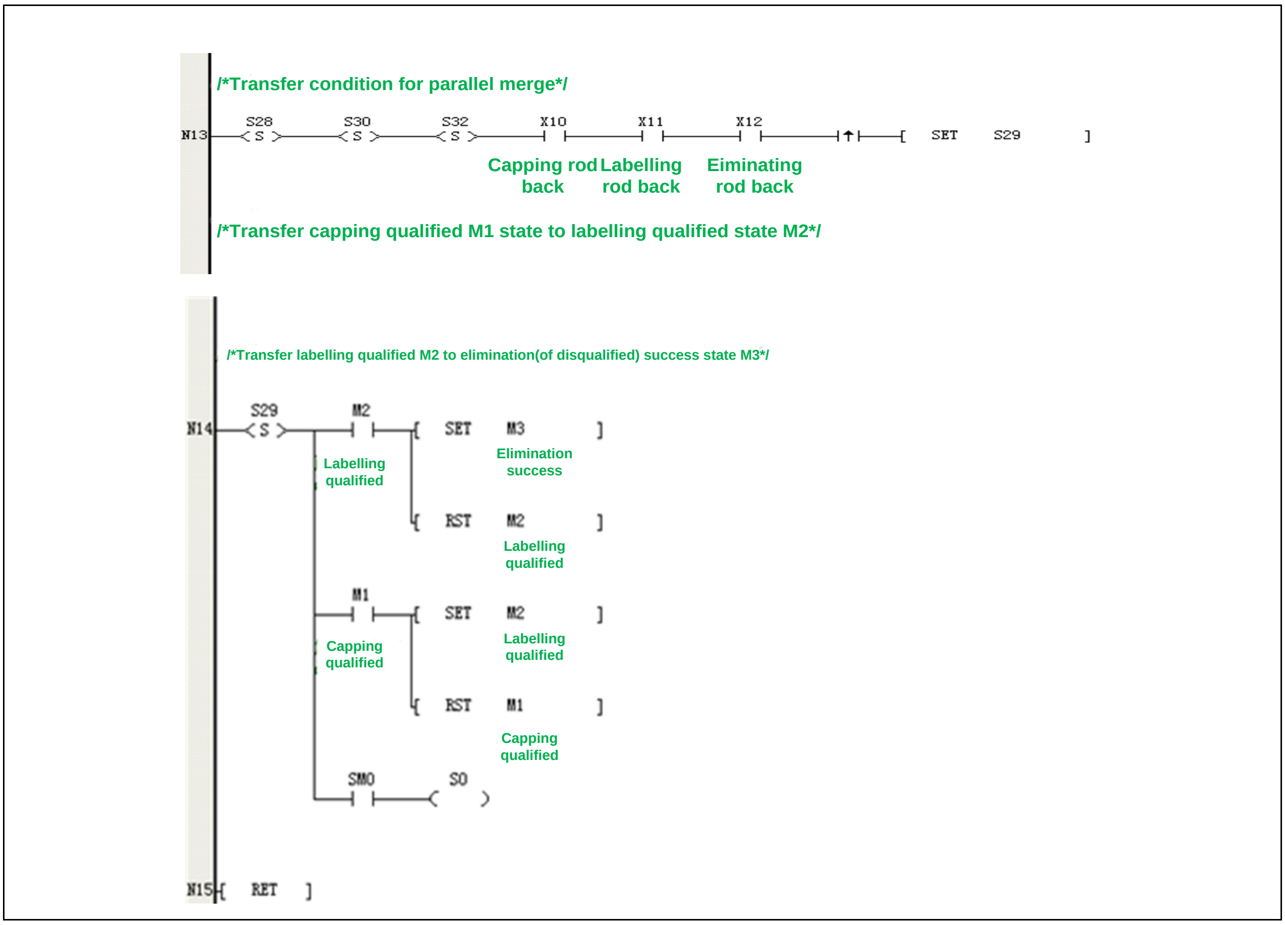
The capping procedure acts according to X1 state. If X1 indicates that no bottle is in position, the capping does not proceed. The labeling procedure acts according to M2 state. If M2 is OFF, it indicates that the bottle in position is disqualified, and the labeling does not proceed. The eliminating procedure acts according to M3. If M3 is ON, it indicates that the bottle is qualified, and the elimination does not proceed, and vice versa. In both cases, M3 is reset in S32 to prepare for the next procedure.


LAD0
Start and stop the program section
S0
Initial step
When a X0 rising edge is detected
The labelling procedures
The eliminating procedures
The capping procedures
S20*
S23*
M2 qualification flag
S26*
M3 qualification flag
Transfer conditions
S21
S28
S24
S30
S27
S32
Capping operations
Jump when no bottle is in position
Jump upon disqualified product
Jump upon qualified product
Elimination operation
S22
S25
S31*
Mark the qualified product (M1)
Labelling operation
S28*
S30*
Mark the qualified product (M2)
S32
Reset the M3 disqualification flag
Parallel merge
S29
Transfer the qualification flag states from M1 to M2 and M2 to M3
S0


/*············Start a stop control program section···············*/
N0
X13
X14
SET S0
Start the switch
Emergency switch
N1
X13
ZRST M1 3
Start the switch
Capping qualified
X14
RST S0
Emergency switch
ZRST S20 13
/*·········SFC program section·········*/
/*Initial step. Transfer condition: rising edge of X0, indicating rotary plate in position.*/
N2
S0
S
Y0
Rotary plate motor
X0
SET S20
Plate in position
SET S23
SET S26


/*Capping*/
N3
S20
S
X1
SET S21
Bottle in position
X1
S28
Bottle in position


N4
S21
S
Y1
Capping cylinder
X7
SET S22
Capping cylinder rod
N5
S22
S
X2
SET M1
Capping check
Capping qualified
X10
SET S28
Capping rod back
/*Labelling*/
N6
S23
S
M2
SET S24
Labelling qualified
M2
S30
Labelling qualified
N7
S24
S
Y2
Labelling cylinder
X6
SET S25
Labelling rod cut
N8
S25
S
X3
RST M2
Labelling detection
Labelling qualified
X11
SET S30
Labelling rod back
/*Elimination*/
N9
S26
S
M3
SET S27
Elimination success
M3
S32
Elimination success
N10
S27
S
Y3
Eliminating cylinder
X5
SET S31
Eliminating rod out
N11
S31
S
X12
SET S32
Eliminating rod back
N12
S32
S
RST M3
Elimination success



/*Transfer condition for parallel merge*/

/*Transfer capping qualified M1 state to labelling qualified state M2*/

/*Transfer labelling qualified M2 to elimination(of disqualified) success state M3*/

N13
S28
S30
S32
X10
X11
X12
SET S29
Capping rod back
Labelling rod back
Eiminating rod back
N14
S29
M2
Labelling qualified
SET M3
Elimination success
RST M2
Labelling qualified
M1
Capping qualified
SET M2
Labelling qualified
RST M1
Capping qualified
SM0
S0
N15
RET

Chapter 8 Operating guide for high-speed input function

This chapter detailedly describes the usages and notes about the high-speedinput function, including high-speed counters and pulse capture.

Chapter 8 Operating guide for high-speed input function 274
8.1 High-speed counter 275
8.1.1 Configuration....................275
8.1.2 Relationship between high-speed counter and SM auxiliary relay 276
8.1.3 How to use the high-speed counters....................278
8.1.4 Notes about the VC1 series high-speed counters....................281
8.2 External pulse capture function 282
8.3 Notes on High-speed input application....................282

8.1 High-speed counter

8.1.1 Configuration

The built-in high-speed counter for VC series micro-PLCs are configured as follows:

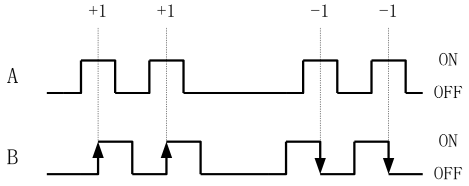
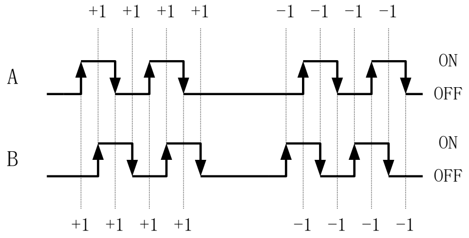
- Configuration table of high-speed counter

Input point / Counter		X0	X1	X2	X3	X4	X5	X6	X7	Max. frequency (kHz)		
										VC5 VC3	VC2 VC1	VC1S
Single-phase one point input mode	C236	Up/Down								200	50	10
	C237		Up/Down									
	C238			Up/Down							10	
	C239				Up/Down							
	C240					Up/Down						
	C241						Up/Down					
	C242							Up/Down			10	/
	C243								Up/Down			
	C244	Up/Down		Reset							50	10
	C245					Up/Down		Reset			10	
	C246	Up/Down		Reset	Start						50	
	C247					Up/Down		Reset	Start		10	
Single-phase bidirectional count input	C248	Up	Down							200	30	10
	C249			Up	Down						5	
	C250					Up	Down				5	/
	C251							Up	Down		5	10
	C252	Up	Down	Reset							30	
	C253					Up	Down	Reset			5	
	C254	Up	Down	Reset	Start						30	
	C255					Up	Down	Reset	Start		5	
Two-phase count input mode	C256	A phase	B phase							200	30	5
	C257			A phase	B phase						5	/
	C258					A phase	B phase					
	C259							A phase	B phase			
	C260	A phase	B phase	Reset							30	5
	C261					A phase	B phase	Reset			5	
	C262	A phase	B phase	Reset	Start						30	
	C263					A phase	B phase	Reset	Start		5	
SPD instruction		Input point	Input point	Input point	Input point	Input point	Input point	Input point	Input point	200	10	10
Pulse capture function		Input point	Input point	Input point	Input point	Input point	Input point	Input point	Input point			

External interrupt number	0/8	1/9	2/10	3/11	4/12	5/13	6/14	7/15	

In the modes listed in the preceding table, the high-speed counters act according to certain input and handle high-speed action according to interrupts. The counting action is unrelated to the PLC scan cycle.

This kind of high-speed counters are of the 32-bit bi-directional counting type. According to their different counting up/down switch over methods, they can be divided into the following four categories:

Counting mode	Counting action
Single-phase one port count input	Counters C236–C247 count down/up when SM236–SM247 are ON/OFF.
Single-phase bidirectional count input	Corresponding to the action of counting up or down input, the counters C248–C255 are automatically incremented/decremented. You can know the current count direction of the corresponding counter through SM248–SM255. The counter counts up when the SM element is OFF, or counts down when OFF.
Two-phase count input	When SM100–SM103 is set to OFF, counters C256–C263 are automatically incremented and decremented according to the two-phase input. You can know the current count direction of the corresponding counter through SM256–SM263. The counter counts up when the SM element is OFF, or counts down when OFF. The count direction is defined as follows: 
Two-phase quad frequency count input	When SM100–SM103 is set to ON, counters C256–C263 are automatically quadrupled up and down according to the two-phase input. You can know the current count direction of the corresponding counter through SM256–SM263. The counter counts up when the SM element is OFF, or counts down when OFF. The count direction is defined as follows:  +1 +1 +1 +1 −1 −1 −1 −1 A B ON OFF ON OFF +1 +1 +1 +1 −1 −1 −1 −1

8.1.2 Relationship between high-speed counter and SM auxiliary relay

- Special auxiliary relay for controlling count direction

Type	Counter SN	Up/down control
Single-phase one point count input	C236	SM236
	C237	SM237
	C238	SM238
	C239	SM239
	C240	SM240
	C241	SM241
	C242	SM242
	C243	SM243
	C244	SM244
	C245	SM245
	C246	SM246
	C247	SM247

- Special auxiliary relay for monitoring count direction

Type	Counter SN	Up/down monitor
Single-phase bidirectional count input	C248	SM248
	C249	SM249
	C250	SM250
	C251	SM251
	C252	SM252

- Special auxiliary relay for controlling quad-frequency

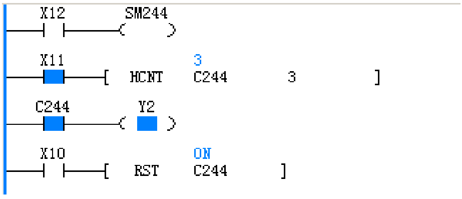
Type	Counter SN	Quad-frequency setting
Two-phase count input	C256	SM100
	C257	SM101
	C258	SM102
	C259	SM103
	C260	SM100
	C261	SM102
	C262	SM100
	C263	SM102

Type	Counter SN	Up/down monitor
	C253	SM253
	C254	SM254
	C255	SM255
Two-phase count input	C256	SM256
	C257	SM257
	C258	SM258
	C259	SM259
	C260	SM260
	C261	SM261
	C262	SM262
	C263	SM263

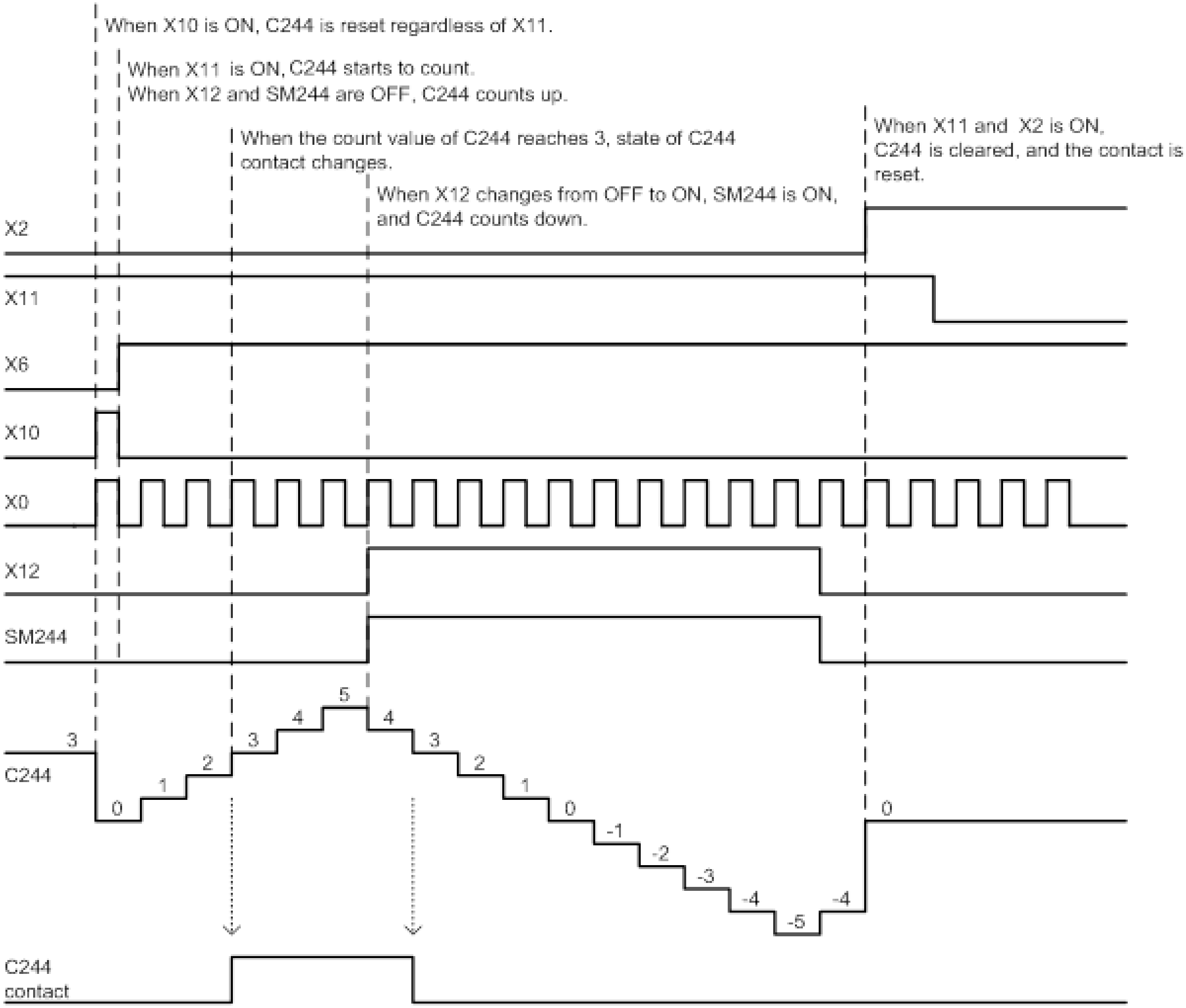
8.1.3 How to use the high-speed counters

- How to use the single-phase one point input high-speed counters

Characteristics: A single-phase one point input high-speed counter starts to count only when the pulse input changes from OFF to ON, and the count direction is determined by its corresponding special auxiliary relay SM. An example of the action is shown in the following figure.



The time sequence chart of the contact actions in the program is shown in the following figure.


When X10 is ON, C244 is reset regardless of X11.

When X11 is ON, C244 starts to count.
When X12 and SM244 are OFF, C244 counts up.

When the count value of C244 reaches 3, state of C244 contact changes.

When X12 changes from OFF to ON, SM244 is ON, and C244 counts down.

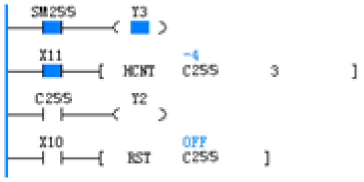
When X11 and X2 is ON, C244 is cleared, and the contact is reset.

Note:
1. Counter input points: X0.
2. High-Speed counters, when used in DHSCS, DHSCR, DHSZ, DHSP, and DHST instructions, can trigger operations or controls free from the scan cycle.

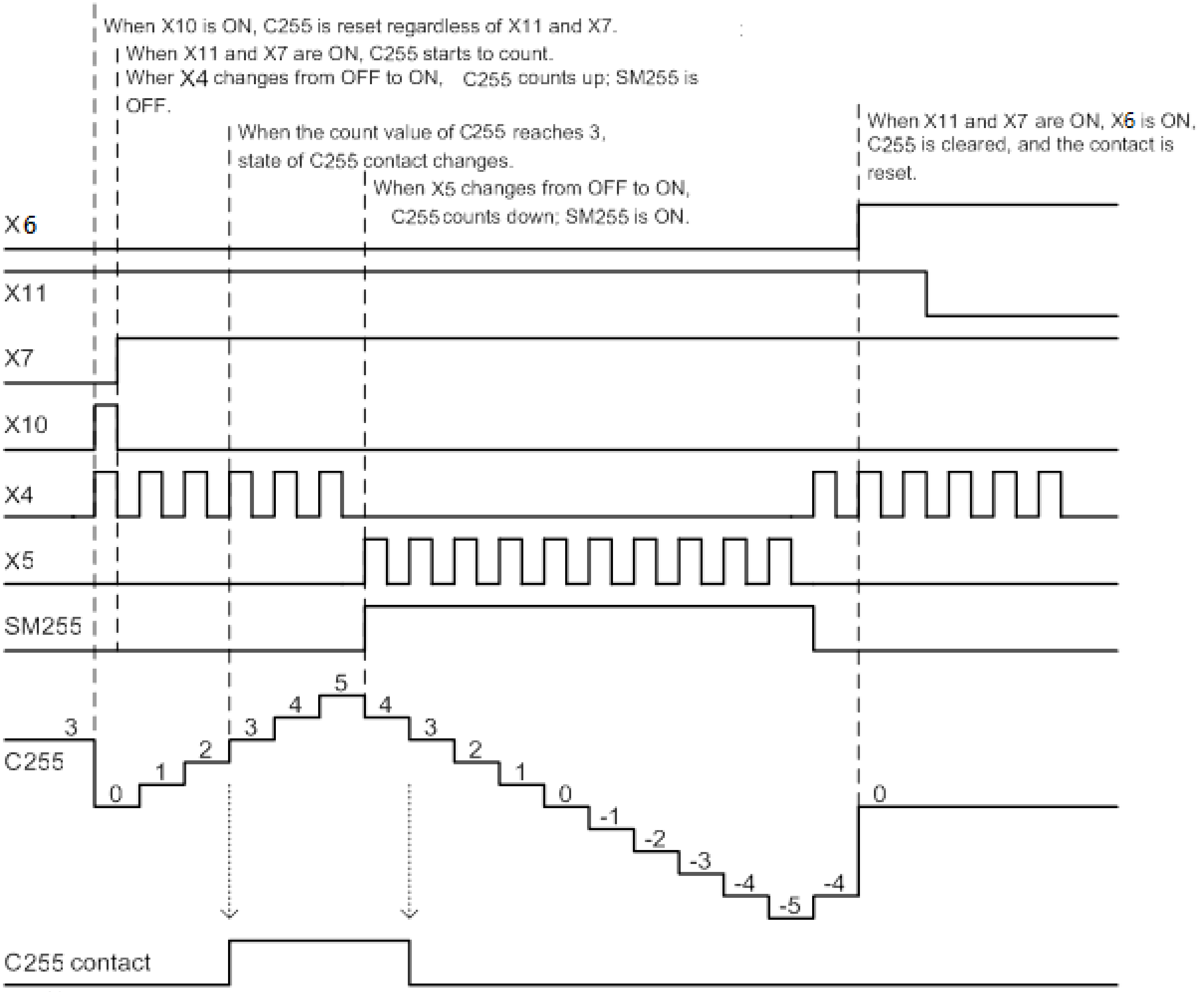
- How to use the single-phase bidirectional count inputhigh-speed counters

Characteristics: A single-phase bidirectional count input high-speed counter starts to count only when the pulse input changes from OFF to ON, and the count direction is determined by their corresponding two input points. The state of the high-speed counter is determined by its corresponding special auxiliary relay SM.

An example of the action is shown in the following figure.


SM255
Y3
X11
HCNT
-4
C255
3
]
C255
Y2
X10
RST
OFF
C255
]

The time sequence chart of the contact actions in the program is shown in the following figure.


When X10 is ON, C255 is reset regardless of X11 and X7.

When X11 and X7 are ON, C255 starts to count.

When X4 changes from OFF to ON, C255 counts up; SM255 is OFF.

When the count value of C255 reaches 3, state of C255 contact changes.

When X5 changes from OFF to ON, C255 counts down; SM255 is ON.

When X11 and X7 are ON, X6 is ON, C255 is cleared, and the contact is reset.

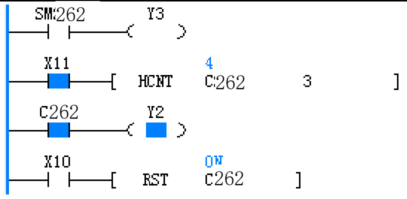
X6
X11
X7
X10
X4
X5
SM255
C255
3
0
1
2
3
4
5
4
3
2
1
0
-1
-2
-3
-4
-5
-4
0
C255 contact

Note:
1. Counter input points: X4 and X5.
2. High-Speed counters, when used in DHSCS, DHSCR, DHSZ, DHSP, and DHST instructions, can trigger operations or controls free from the scan cycle.

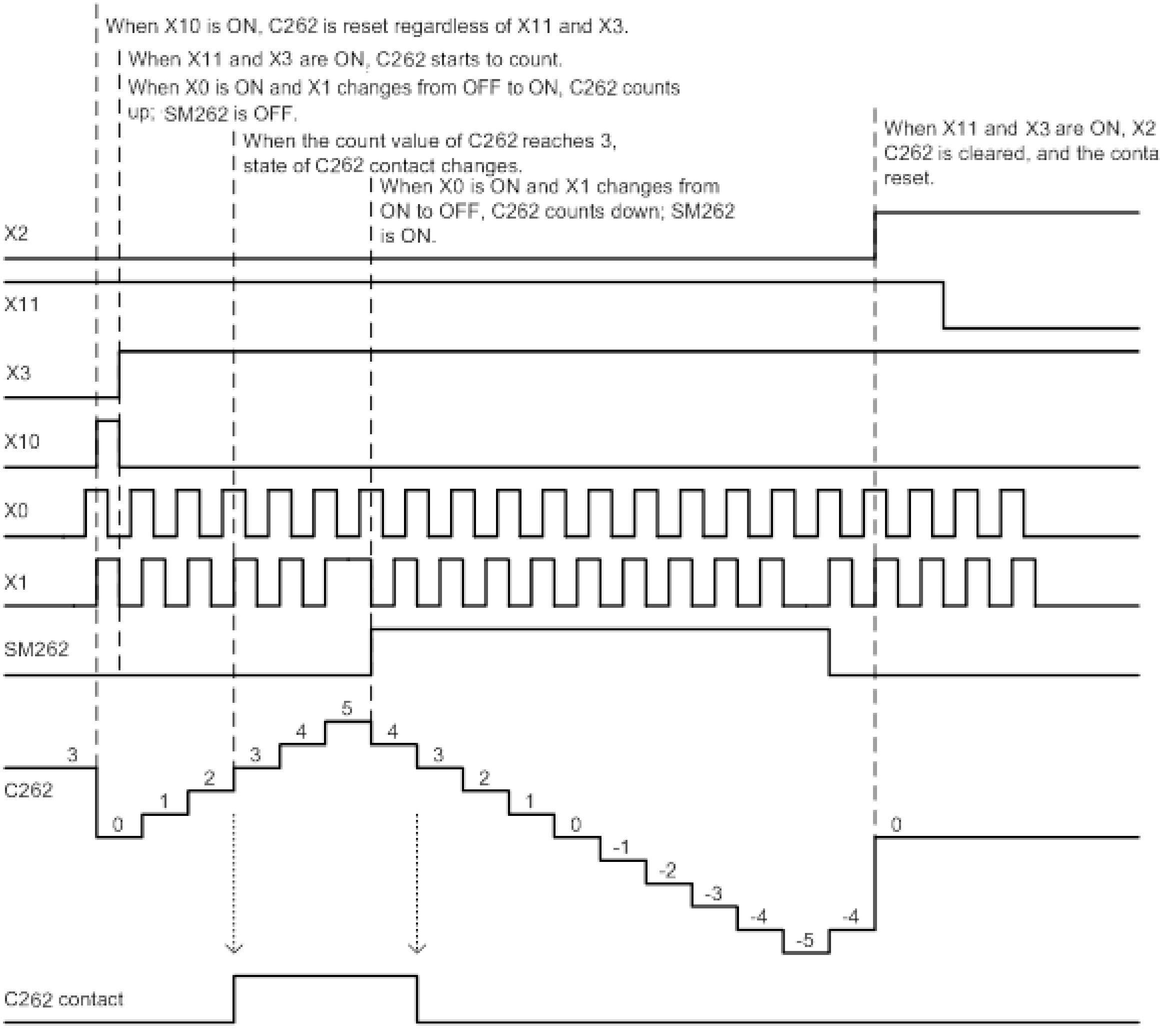
- How to use the two-phase count input high-speed counters

Characteristics: A two-phase count input high-speed counter starts to count only when the pulse input changes from OFF to ON, and the count direction is determined by the phase difference between the two input points. The state of the high-speed counter is determined by its corresponding special auxiliary relay SM.

An example of the action is shown in the following figure.


SM262
Y3
X11
HCNT
4
C262
3
]
C262
Y2
X10
RST
ON
C262
]

The time sequence chart of the contact actions in the program is shown in the following figure.

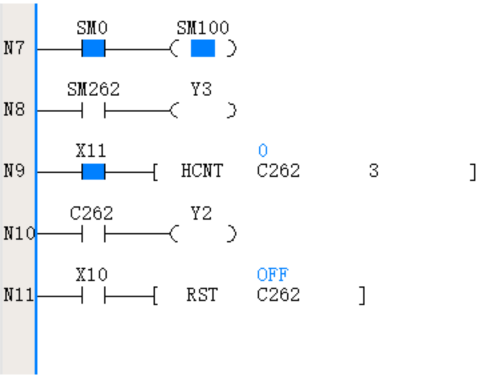


Note:
1. Counter input points: X0 and X1.
2. High-Speed counters, when used in DHSCS, DHSCR, DHSZ, DHSP, and DHST instructions, can trigger operations or controls free from the scan cycle.

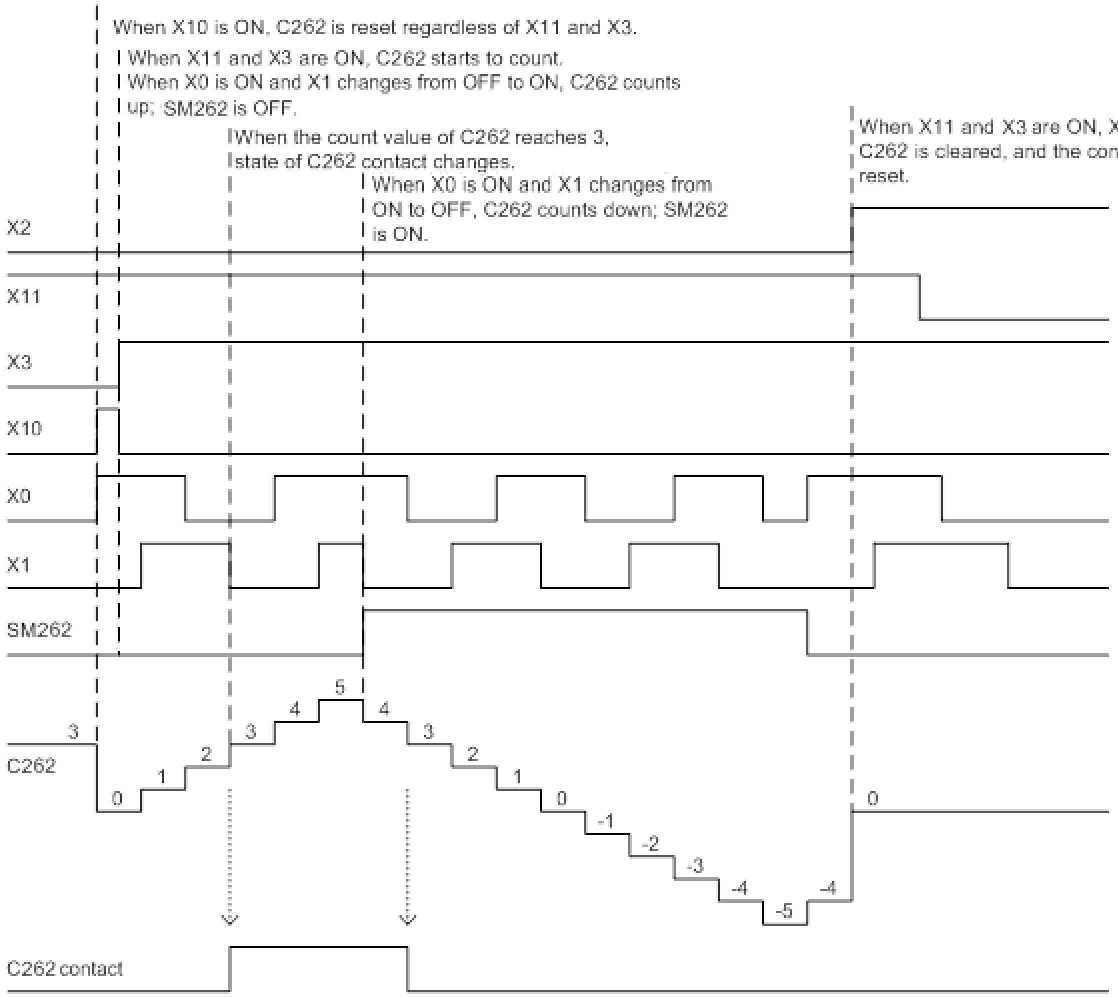
- How to use the two-phase quad frequency count input high-speed counters

Characteristics: A two-phase quad frequency count input high-speed counter starts to count when the pulse input changes from OFF to ON or ON to OFF, and the count direction is determined by the phase difference between the two input points. The state of the high-speed counter is determined by its corresponding special auxiliary relay SM.

An example of the action is shown in the following figure.



The time sequence chart of the contact actions in the program is shown in the following figure.



Note:

1. Counter input points: X0 and X1.
2. High-Speed counters, when used in DHSCS, DHSCR, DHSZ, DHSP, and DHST instructions, can trigger operations or controls free from the scan cycle.

8.1.4 Notes about the VC1 series high-speed counters

- Classification of high-speed counters

  C0-C235 are software counters, C236-C263 are hardware counters, Please select it according to the usage mode.

- Max. frequency sum

  The max. combined frequency cannot exceed 60kHz when multiple high-speed counters (hardware counting mode) are used simultaneously or the high-speed counters (hardware counting mode) and the SPD instruction are used at the same time.

  Total input frequency when multiple software high-speed counters, or when high-speed counters and the SPD instruction, are used at the same time, is shown in the following table.

conditions of usage	Total input frequency
DHSCS, DHSCR, DHSCI, DHSZ, DHSP, and DHSTare not used	≤60 kHz
DHSCS, DHSCR, DHSCI, DHSP, DHST are used	≤30 kHz
DHSZ is used	≤20 kHz

- Max. frequency of hardware counters

  Counters C236, C237 and C256 are hardware counters. Among which:

  - C236 and C237 are single-phase counters with max. counting frequency of 50kHz.
  - C256 is a two-phase counterwith max. counting frequency of 30kHz.

- **Max.**frequency **of software counters**

The high speed counters used in DHSCS, DHSCR, DHSCI, DHSP or DHST instructions works in the software counter mode. The max. input frequency for the single-phase counters and two-phase counters are 10kHz and 5kHz respectively.

When used in the DHSZ instruction, the max. input frequency for the single-phase counters and two-phase counters are 5 kHz and 4 kHz respectively.

8.2 External pulse capture function

The input points that provide the external pulse capture function are X0 – X7. The corresponding SM soft elements arelisted below:

Input hardware point	SM element
X0	SM90
X1	SM91
X2	SM92
X3	SM93
X4	SM94
X5	SM95
X6	SM96
X7	SM97

📖 Note

1. When the external input points change from OFF to ON, the SM soft elements of the corresponding portsare set to ON.

2. SM90–SM97 are cleared when the user programstarts.

3. When using the pulse capture, the counters need to observe the sum limit of the input pulse frequency of each PLC series, otherwise an abnormality may occur.

4. When the high-speed counter or SPD instruction corresponding to HCNT are used on the same input point, the pulse capture becomes invalid after the first scan cycle, regardless of whether the instructionis valid or not.

8.3 Notes on High-speed input application

The input points X0–X7 are used as input signals in functions such as high-speed counter, SPD instruction, pulse capture, and external interrupt. However, these functions cannot be used at the same time, because many different functions may use the same one or multiple input points. Therefore, during the PLC programming, only one of the several functions that aninput point can provide is available. If the X0–X7 input pointsare used repeatedly in the user program, the user program cannot be compiled.

Chapter 9 Using Interrupts

This chapter detailedly describes the mechanisms, processing procedures, and usages of various interrupts.

Chapter 9 Using Interrupts..283
- 9.1 Interrupt program..284
- 9.2 Processing interrupt event..286
- 9.3 Using timed interrupt..286
- 9.4 Using external interrupts..289
- 9.5 Using high-speed counter interrupt..290
- 9.6 Using PTO output completion interrupt..291
- 9.7 Using serial port-based interrupt..292

9.1 Interrupt program

When an interrupt event occurs, the normal scan cycle is interrupted and the interrupt program is called and executed with priority,which is called the interrupt processing mechanism. For the time-critical and event-triggered control tasks, you often need to adopt this special system processing mechanism.

The system provides many kinds of programmable interrupt resources. Each kind of interrupt resource can trigger a type of interrupt events, and each type of interrupt eventsis independently numbered.

In order to deal with a certain interrupt event, you must compile a processing program, that is, an interrupt program,

which is an independent POU that constitutes the user program.

An event number needs to be designated for each interrupt program in order to link the user interrupt program with the interrupt events designated with the event number. When responding to the interrupt request of the interrupt event, the system calls the corresponding user interrupt program based on the interrupt event number.

The following list shows the interrupt resources provided by the VC series micro-PLCs:

Event number	Interrupt event	Enabling SM	Event number	Interrupt event	Interrupt enabling SM
0	X0 input rising edge interrupt	SM25	33	High-speed counter interrupt 0	SM58
1	X1 input rising edge interrupt	SM26	34	High-speed counter interrupt 1	SM58
2	X2 input rising edge interrupt	SM27	35	High-speed counter interrupt 2	SM58
3	X3 input rising edge interrupt	SM28	36	High-speed counter interrupt 3	SM58
4	X4 input rising edge interrupt	SM29	37	High-speed counter interrupt 4	SM58
5	X5 input rising edge interrupt	SM30	38	High-speed counter interrupt 5	SM58
6	X6 input rising edge interrupt	SM21	39	High-speed counter interrupt 6	SM58
7	X7 input rising edge interrupt	SM32	40	High-speed counter interrupt 7	SM58
8	X0 input falling edge interrupt	SM33	41	Interpolation completion interrupt 1	SM69
9	X1 input falling edge interrupt	SM34	42	Interpolation completion interrupt 2	SM69
10	X2 input falling edge interrupt	SM35	43	Interpolation completion interrupt 3	SM69
11	X3 input falling edge interrupt	SM36	44	Interpolation completion interrupt 4	SM69
12	X4 input falling edge interrupt	SM37	45	Position-based interrupt 0 of high-speed output	SM61
13	X5 input falling edge interrupt	SM38	46	Position-based interrupt 1 of high-speed output	SM62
14	X6 input falling edge interrupt	SM39	47	Position-based interrupt 2 of high-speed output	SM105
15	X7 input falling edge interrupt	SM40	48	Position-based interrupt 3 of high-speed output	SM106
16	COM 0 frame transmission interrupt	SM41	49	Position-based interrupt 4 of high-speed output	SM107
17	COM 0 frame receiving interrupt	SM42	50	Position-based interrupt 5 of high-speed output	SM108
18	COM 1 frame transmission interrupt	SM43	51	Position-based interrupt 6 of high-speed output	SM98
19	COM 1 frame receiving interrupt	SM44	52	Position-based interrupt 7 of high-speed output	SM99
20	COM 2 frame transmission interrupt	SM45			
21	COM 2 frame receiving interrupt	SM46			
22	Timed interrupt 0	SM47			
23	Timed interrupt 1	SM48			
24	Timed interrupt 2	SM49			
25	PTO (Y0) output completioninterrupt	SM50			
26	PTO (Y1) output completion interrupt	SM51			
27	PTO (Y2) output completion interrupt	SM52			

28	PTO (Y3) output completion interrupt	SM53				
29	PTO (Y4) output completion interrupt	SM54				
30	PTO (Y5) output completion interrupt	SM55				
31	PTO (Y6) output completion interrupt	SM56				
32	PTO (Y7) output completion interrupt	SM57				

9.2 Processing interrupt event

1. When a certain interrupt event occurs, if it has been enabled, its corresponding event number is added to the interrupt request queue record, which is afirst-in-first-out queue with 8-record long.

2. Processing of the interrupt request by the system:

(1) If the system detects that any request in the interrupt queue, it stops the normal execution of the user program.

(2) The system reads the head record in the interrupt request queue, which records the number of the interrupt event that occurred first. The user interrupt program corresponding to the event number is called and executed.

(3) When the corresponding interrupt program has been executed (the interrupt return instruction has been executed), the interrupt request is processed, the corresponding head record of the request queue is deleted from the queue, and all the following records takes one step forward.

(4) The system detects again whether the interrupt request queue is empty. If it is not empty, the system repeats the above procedures until the queue is empty.

(5) When the interrupt request queue is empty, the system continues to execute the interrupted main program.

3. The system can handle only one interrupt request at one time. When the system is processing an interrupt request,a new interrupt event is added to the interrupt request queue rather than being responded immediately. The system processes it after all the requests ahead of it in the queue are processed.

4. When there are 8 records in the interrupt request queue, the system automatically masks the new interrupt eventso that no new request is added to the queue. The mask is not cancelled until all the requests in the queue have been processed and the interrupted main program has been executed.

🕮 Note

1. The interrupts should be brief, or abnormalities may occur, including the mask of other interrupt events (missing of interruptrequests), system scan overtime and low execution efficiency of the main program.
2. It is prohibited to call other subprograms in the interrupt program.
3. If you want to refresh I/O immediately during the interrupt, you can use the REF instruction. You need to note that the execution time of the REF instruction is related to the number of the I/O points to be refreshed.
4. An interrupt event can generate an interrupt request only when the corresponding interrupt event flag is enabled (each type of interrupt event enable/disable control corresponds to the related SM elements. To enable some type of interrupt events requiressetting the corresponding SM element to ON), and the global interrupt enable flag is turned on.
5. When an interrupt request with no corresponding interrupt program in the user program is generated, the system responds to the interrupt request, but no operation.

9.3 Using timed interrupt

- Description of the timed interrupt

The timed interrupt is the interrupt event generated by the system periodically based on the set intervals.

The timed interrupt program is mainly applicable to the situation that requires timed and immediate processing by the system,such as the timed sampling of analog inputs, and timed analog output updating based on certain waveforms.

You can set the intervals (unit: ms) for the timed interrupts by setting the corresponding SD elements. The system generates the interrupt event only when the set time interval is reached (recommended mini.interval: > 4ms).

The ON/OFF state of certain SM elements can enable/disable the corresponding timed interrupts.

The VC Series PLCs provides you with three timed interrupt resources as follows.

Table 9-1 Timed interrupt resource list

Timed interrupt	Interrupt event number	(SD)Intervals of timed interrupt (SD)	Enable control (SM)
0	22	SD47	SM47
1	23	SD48	SM48
2	24	SD49	SM49

🕮 Note

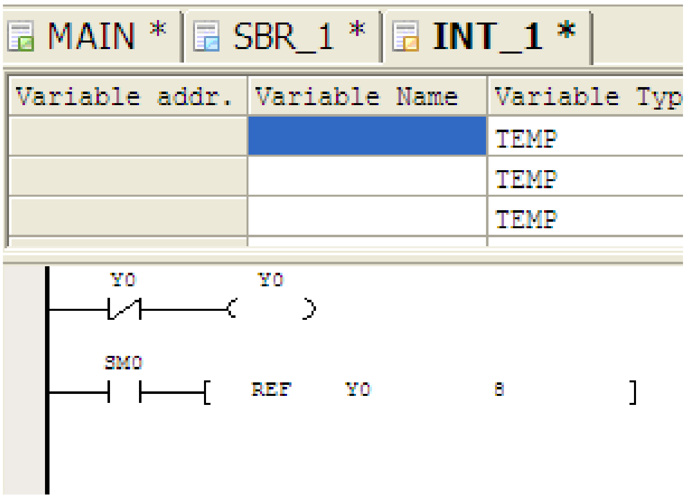
1. Setting of enable control elements cannot affect the execution of the timed interrupts in the interrupt request queue.
2. The timing for a re-enabled interrupt starts from zero.

To change the interval of the timed interrupt when the program is running, it is recommended to follow the following procedures:1. To disable the timed interrupt.2. To change the interval.3. To enable the timed interrupt.

- Application instance of the timed interrupt

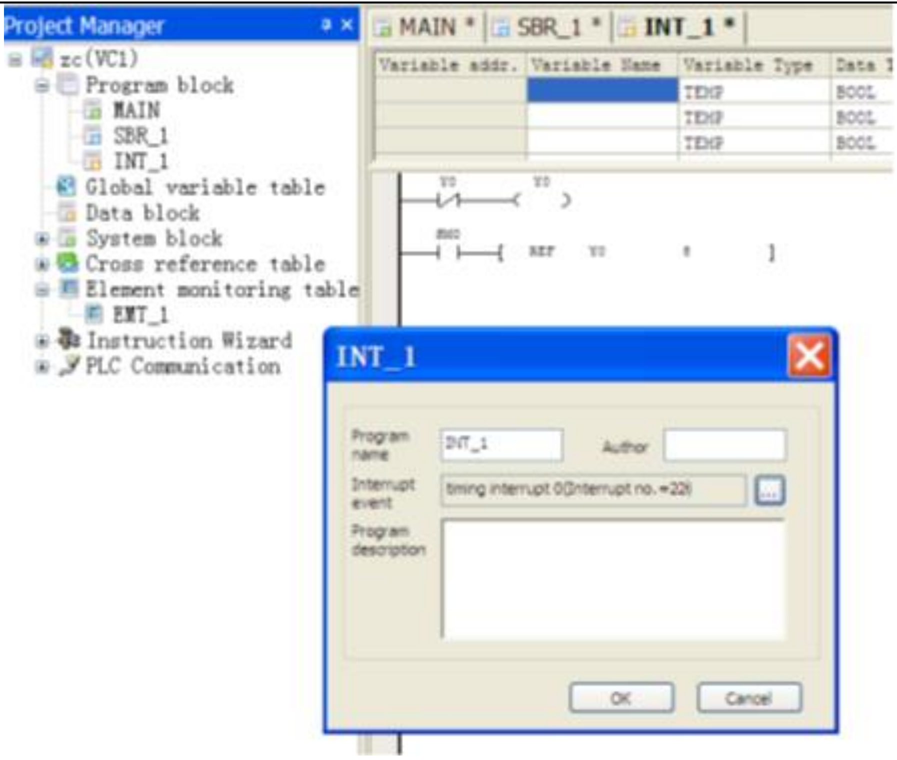
In the application instance, the system upsets the output of Y0 once a second based on the timed interrupt 0, which makes Y0 flashing periodically.

1. Writing an interrupt program for the interrupt event.



Variable addr.	Variable Name	Variable Typ
		TEMP
		TEMP
		TEMP

2. Specifying an interrupt event number for the interrupt program:


Project Manager
zc(VC1)
Program block
MAIN
SBR_1
INT_1
Global variable table
Data block
System block
Cross reference table
Element monitoring table
EMT_1
Instruction Wizard
PLC Communication
MAIN *
SBR_1 *
INT_1 *
INT_1
OK
Cancel

Variable addr.	Variable Name	Variable Type	Data T
		TEMP	BOOL
		TEMP	BOOL
		TEMP	BOOL

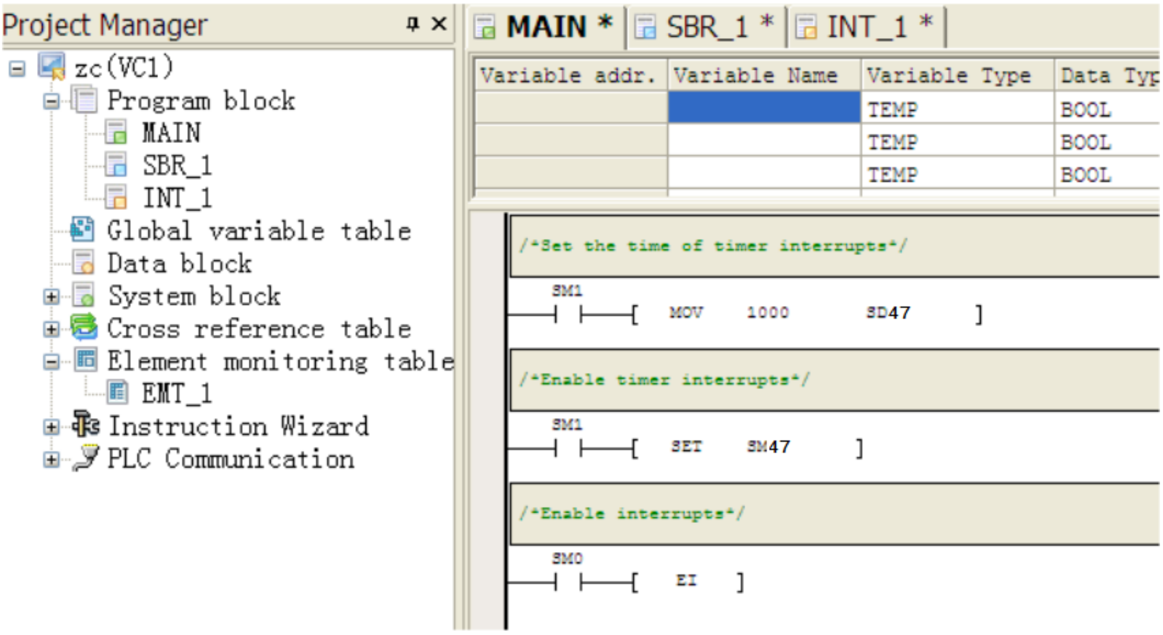
Program name: INT_1

Author:

Interrupt event: timing interrupt 0(interrupt no.=22)

Program description:

3. Setting the interval for the timed interrupt and enabling the timed interrupt in the main program (MAIN).



Variable addr.	Variable Name	Variable Type	Data Typ
		TEMP	BOOL
		TEMP	BOOL
		TEMP	BOOL

```
/*Set the time of timer interrupts*/
SM1
--| |----[ MOV    1000    SD47    ]
/*Enable timer interrupts*/
SM1
--| |----[ SET    SM47    ]
/*Enable interrupts*/
SM0
--| |----[ EI    ]
```

9.4 Using external interrupts

- Description of the external interrupts

The external interrupts are related to the actual input points of the PLCs, which are classified into input rising edge interrupt and input falling edge interrupt. In the user program, they add the actions related to external events to the external interrupt program.

The highest response frequency of the system to the external event is 1K. The external events over 1K may be lost.

The rising edge interrupt and falling edge interrupt cannot be used on the same port simultaneously. All the external interrupts are only valid when the global interrupt control EI and corresponding enabling SM are valid.

The detailed relationship is as follows:

Interrupt number	Enabling element		Interrupt number	Enabling element
0 or 8	SM25/SM33		4 or 14	SM29/SM37
1 or 9	SM26/SM34		5 or 15	SM30/SM38
2 or 10	SM27/SM35		6 or 16	SM31/SM39
3 or 11	SM28/SM36		7 or 17	SM32/SM40

The external interrupts are numbered as follows:

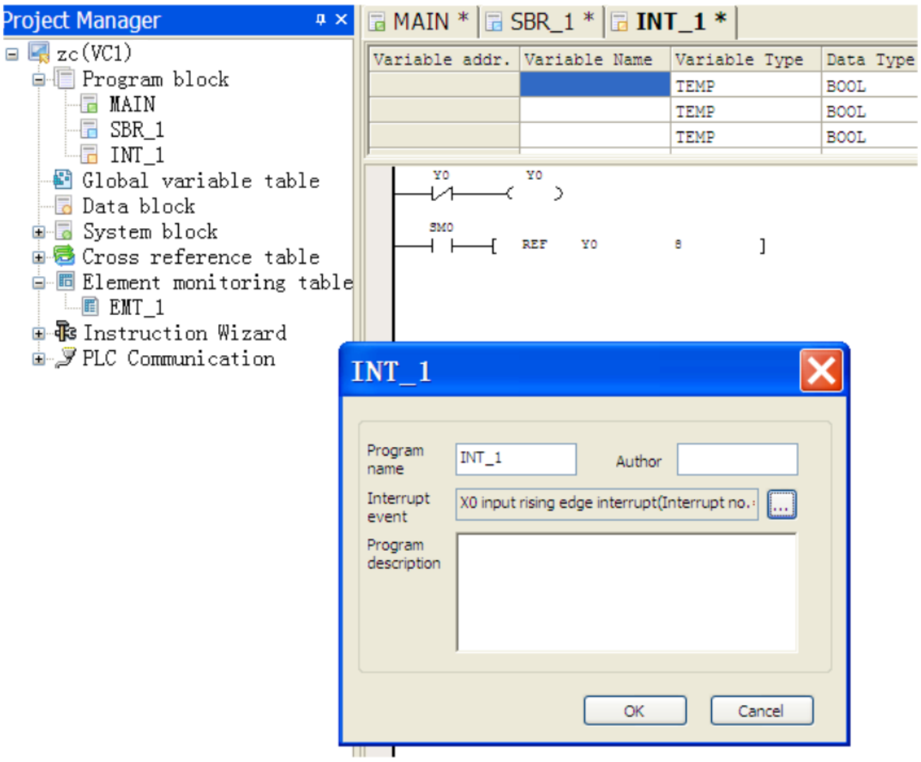
Interrupt number	Interrupt source		Interrupt number	Interrupt source
0	X0 input rising edge interrupt		9	X1 input falling edge interrupt
1	X1 input rising edge interrupt		10	X2 input falling edge interrupt
2	X2 input rising edge interrupt		11	X3 input falling edge interrupt
3	X3 input rising edge interrupt		12	X4 input falling edge interrupt
4	X4 input rising edge interrupt		13	X5 input falling edge interrupt
5	X5 input rising edge interrupt		14	X6 input falling edge interrupt
6	X6 input rising edge interrupt		15	X7 input falling edge interrupt
7	X7 input rising edge interrupt		16	Reserved
8	X0 input falling edge interrupt		17	Reserved

The single input pulse frequency of X0 - X7 is less than 200Hz.

- Application instance of the external interrupt

In the application instance, the system upsets the output of Y0 based on the corresponding external interrupt 0 function and rising edge input event of X0.

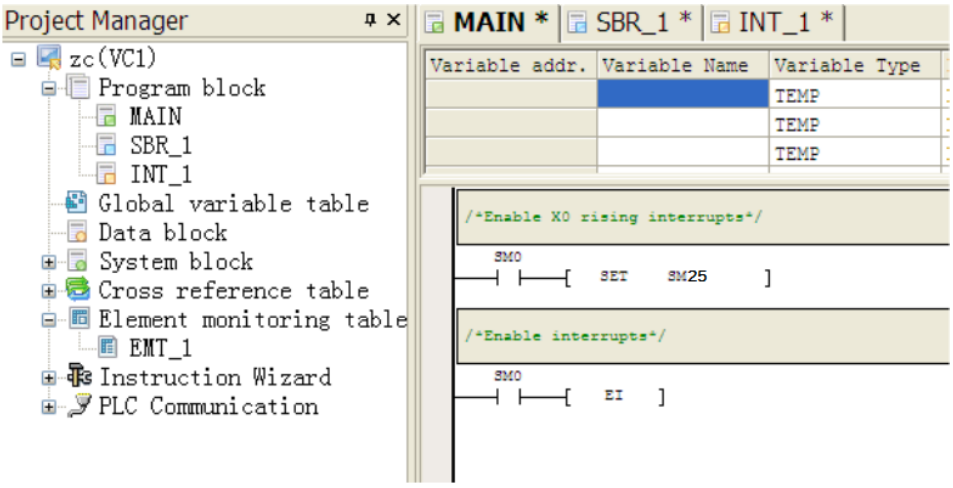
1. Writing the interrupt program to upset the state of Y0 once upon every interrupt and output immediately. To use an interrupt,you should select its corresponding interrupt number. For details, you can see the specific operation in the following figure.


Project Manager
zc(VC1)
Program block
MAIN
SBR_1
INT_1
Global variable table
Data block
System block
Cross reference table
Element monitoring table
EMT_1
Instruction Wizard
PLC Communication
MAIN *
SBR_1 *
INT_1 *
Y0
Y0
SM0
REF
Y0
8
INT_1
OK
Cancel

Variable addr.	Variable Name	Variable Type	Data Type
		TEMP	BOOL
		TEMP	BOOL
		TEMP	BOOL

Program name: INT_1
Author:
Interrupt event: X0 input rising edge interrupt(Interrupt no.:
Program description:

2. Writing the EI instruction in MAIN, and setting SM25, the interrupt enabling flag corresponding to X0 input rising edge interrupt, to be valid.


Project Manager
zc(VC1)
Program block
MAIN
SBR_1
INT_1
Global variable table
Data block
System block
Cross reference table
Element monitoring table
EMT_1
Instruction Wizard
PLC Communication
MAIN *
SBR_1 *
INT_1 *
SM0
SET
SM25
SM0
EI

Variable addr.	Variable Name	Variable Type
		TEMP
		TEMP
		TEMP

/*Enable X0 rising interrupts*/

/*Enable interrupts*/

9.5 Using high-speed counter interrupt

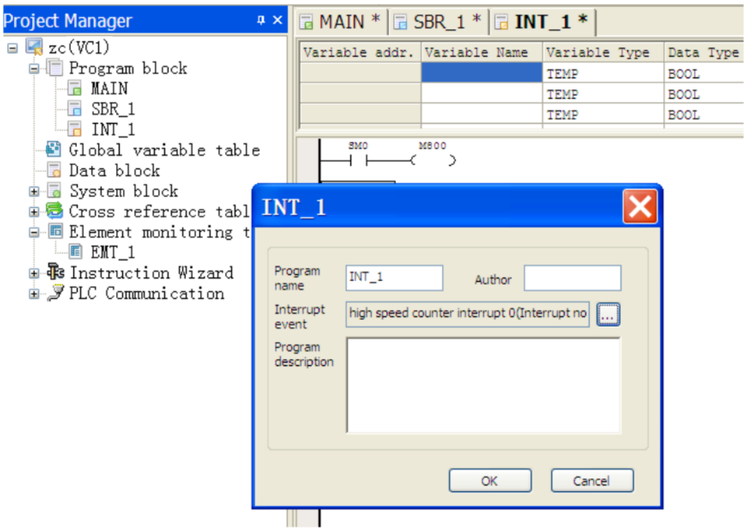
- Description of the high-speed counter interrupt

The high-speed counter interrupt is valid only when it is used together with the HCNT instruction or DHSCI instruction, and generates high-speed counter interrupt based on the count value of the high-speed counter. You can compile programs related to external pulse input in the high-speed interrupt program. All high-speed counter interrupts (33-40) arevalid only when the EI instruction and corresponding interrupt enable flag are valid.

- Application instance of the high-speed counter interrupt

In the application instance, the system uses the high-speed counter interrupt instruction of X0 to respond to the interrupt program (number 33) when the value of the externalcounter C236 reaches the value specified through the DHSCI instruction.

1. Writing the interrupt subprogram. To use an interrupt subprogram, you need to select its corresponding interrupt number. For details, you can see the specific operation in the following figure.

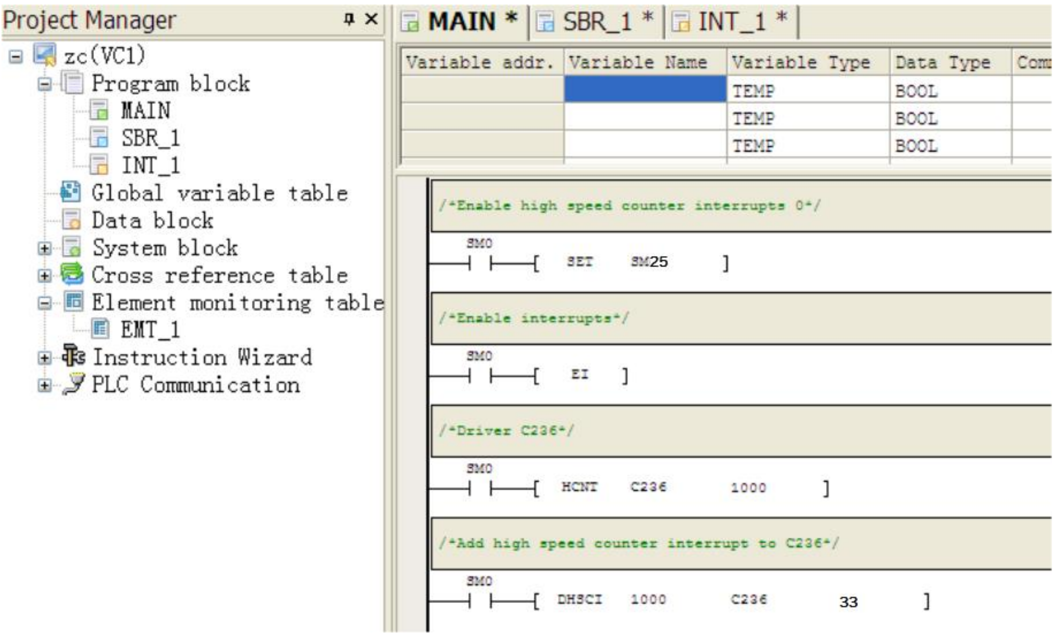

Project Manager
zc(VC1)
Program block
MAIN
SBR_1
INT_1
Global variable table
Data block
System block
Cross reference tabl
Element monitoring t
EMT_1
Instruction Wizard
PLC Communication
MAIN *
SBR_1 *
INT_1 *
Variable addr.
Variable Name
Variable Type
Data Type
TEMP
BOOL
TEMP
BOOL
TEMP
BOOL
SM0
M800
INT_1
Author
...
OK
Cancel

Program name: INT_1

Interrupt event: high speed counter interrupt 0(Interrupt no

Program description:

2. Writing the EI instruction in MAIN, and setting SM58, the interrupt enabling flag corresponding to the high-speed counter interrupt, to be valid. Driving the high-speed counter C236 and high-speed counter interrupt instruction.


Project Manager
zc(VC1)
Program block
MAIN
SBR_1
INT_1
Global variable table
Data block
System block
Cross reference table
Element monitoring table
EMT_1
Instruction Wizard
PLC Communication
MAIN *
SBR_1 *
INT_1 *
Variable addr.
Variable Name
Variable Type
Data Type
Com
TEMP
BOOL
TEMP
BOOL
TEMP
BOOL

```
/*Enable high speed counter interrupts 0*/
SM0
--| |--[ SET    SM25 ]

/*Enable interrupts*/
SM0
--| |--[ EI ]

/*Driver C236*/
SM0
--| |--[ HCNT   C236   1000 ]

/*Add high speed counter interrupt to C236*/
SM0
--| |--[ DHSCI  1000   C236   33 ]
```

9.6 Using PTO output completion interrupt

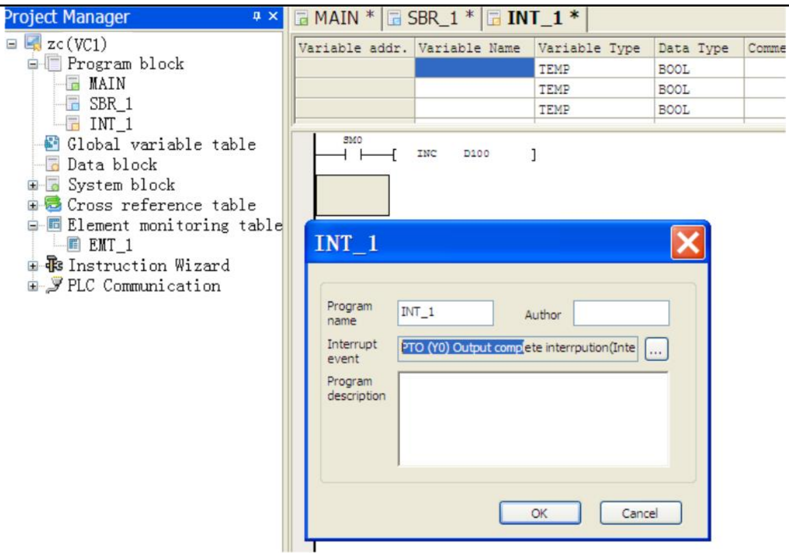
- Description of the PTO output completioninterrupt

The PTO output completion interrupt is triggered when the enable flag (SM50,SM51,SM52) is set and the high-speed pulse output at Y0,Y1,Y2 is finished. You can carry out the relevant processing in the interrupt subprogram.

- Application instance of the PTO output completion interrupt

In the application instance, the system uses the high-speed pulse output of Y0 to respond to the interrupt program (number 25) when Y0 high-speed pulse output is finished.

1. Writing the interrupt program (INT_1): Writing a program for the interrupt code to realize the control. To use an interrupt subprogram, you need to select its corresponding interrupt number. For details, you can see the specific operation in the following figure.


Project Manager
zc(VC1)
Program block
MAIN
SBR_1
INT_1
Global variable table
Data block
System block
Cross reference table
Element monitoring table
EMT_1
Instruction Wizard
PLC Communication
MAIN *
SBR_1 *
INT_1 *
Variable addr.
Variable Name
Variable Type
Data Type
Comme
TEMP
BOOL
TEMP
BOOL
TEMP
BOOL
SM0
INC
D100
INT_1
OK
Cancel

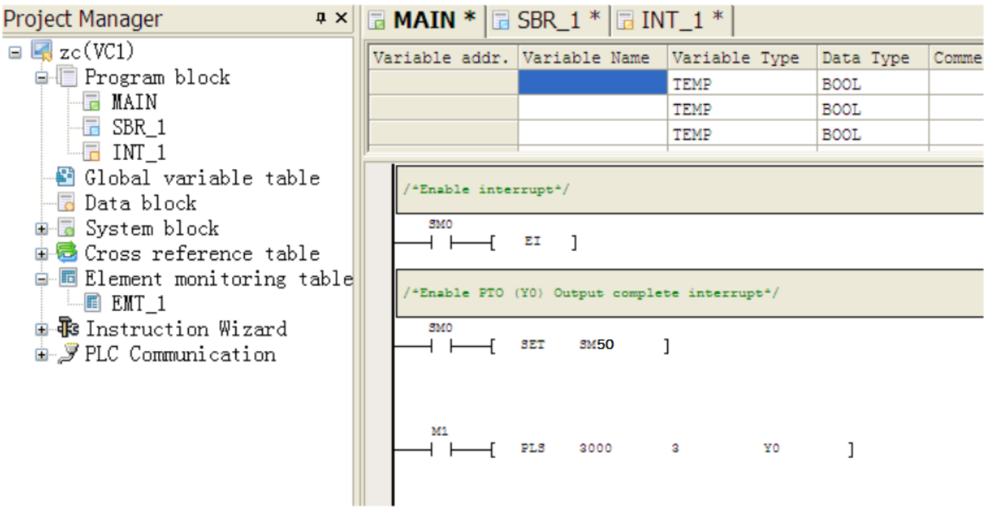
Program name: INT_1

Author:

Interrupt event: PTO (Y0) Output complete interrpution(Inte

Program description:

2. Writing the functions in MAIN: Enabling the global interrupt of the system and setting the PTO output completion interrupt enable flag SM50 to be valid. Using the PLS instruction.


Project Manager
zc(VC1)
Program block
MAIN
SBR_1
INT_1
Global variable table
Data block
System block
Cross reference table
Element monitoring table
EMT_1
Instruction Wizard
PLC Communication
MAIN *
SBR_1 *
INT_1 *
Variable addr.
Variable Name
Variable Type
Data Type
Comme
TEMP
BOOL
TEMP
BOOL
TEMP
BOOL

```
/*Enable interrupt*/
SM0
--| |--[ EI ]

/*Enable PTO (Y0) Output complete interrupt*/
SM0
--| |--[ SET   SM50   ]

M1
--| |--[ PLS   3000   3   Y0   ]
```

9.7 Using serial port-based interrupt

- Description of the serial port-based interrupt

When a serial port is in the free-port protocol mode, the system generates an interrupt event according to the sending or receiving events of the serial port.

For each serial port, the system provides two interrupt resources for you. The serial port-based interrupt programis mainly applicable to the scenario that requires special processing on the receiving and sending of character/frame (XMT and RCV) at the serial port andtimely processing by the system. It can immediately respond to the processing requirements after a character or frame is transmitted or received, regardless ofthe scan time.

You can set the ON/OFF state of the corresponding SM elements to enable or disable the serial port-based interrupt. When the serial port-based interrupt is disabled, the ones that have been added to the interrupt queue continue to be executed. The XMT instruction cannot be called in the character transmission interrupt processing subprogram when the power flow is normally on. Otherwise, it may lead to the interrupt subprogram nesting and thus block the execution of the user program.

Frame receiving and sending interrupts refer to the interrupt events that are triggered after the XMT and RCV instructions are executed.

Serial-port interrupt resource list:

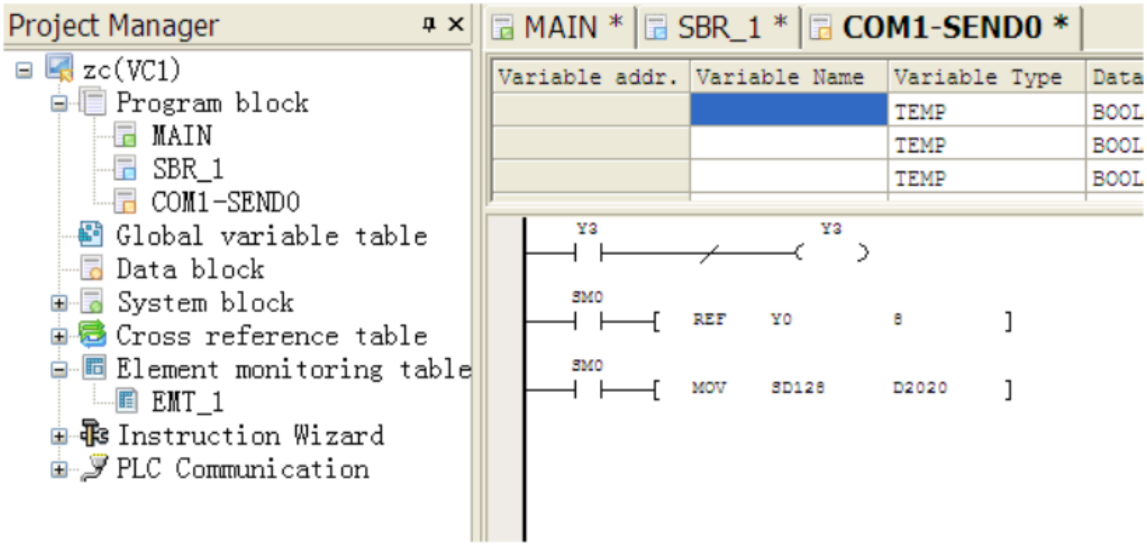
Interrupt number	Interrupt event	Interrupt enabling SM
16	Frame sending interrupt of PORT0	SM41

Interrupt number	Interrupt event	Interrupt enabling SM
17	Frame receiving interrupt of PORT0	SM42
18	Frame sending interrupt of PORT1	SM43
19	Frame receiving interrupt of PORT1	SM44
20	Frame sending interrupt of PORT2	SM45
21	Frame receiving interrupt of PORT2	SM46

- Application instance of theserial port-based interrupt

In the example, with the serial port-based frame sending function, the system upsets the Y3 output once when aframe is sent out and generate flashing effect based on the frequency of the character sending frame.

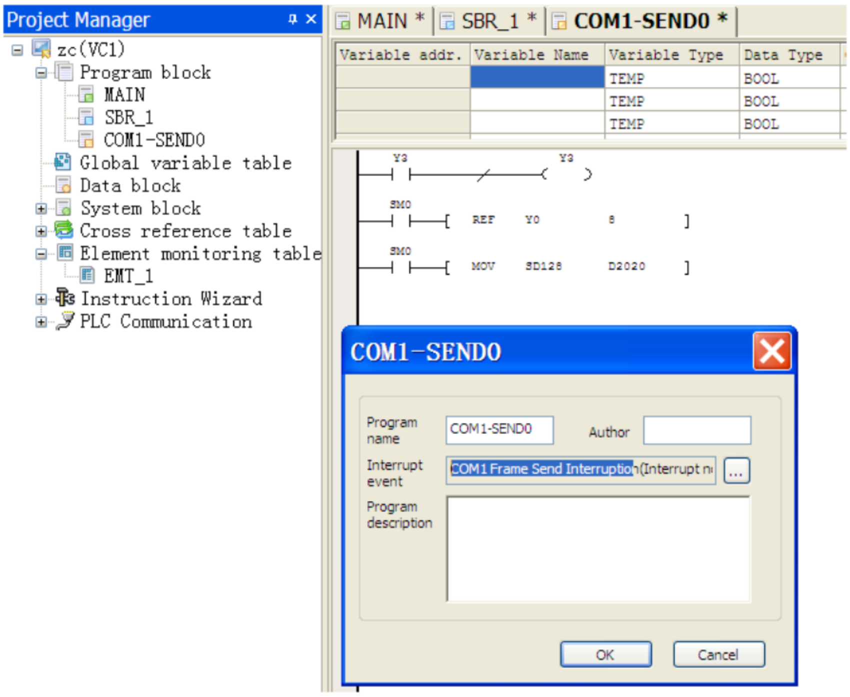
1. Writing the interrupt program and the processing code when the serial port sending frame is completed and the interrupt is triggered.


Project Manager
zc(VC1)
Program block
MAIN
SBR_1
COM1-SEND0
Global variable table
Data block
System block
Cross reference table
Element monitoring table
EMT_1
Instruction Wizard
PLC Communication
MAIN *
SBR_1 *
COM1-SEND0 *

Variable addr.	Variable Name	Variable Type	Data
		TEMP	BOOL
		TEMP	BOOL
		TEMP	BOOL

```
Y3                 Y3
─┤ ├───/───(  )
SM0
─┤ ├──[ REF   Y0     8      ]
SM0
─┤ ├──[ MOV   SD128  D2020  ]
```

2. Specifying the corresponding interrupt event number for the interrupt program:


Project Manager
COM1-SEND0
OK
Cancel

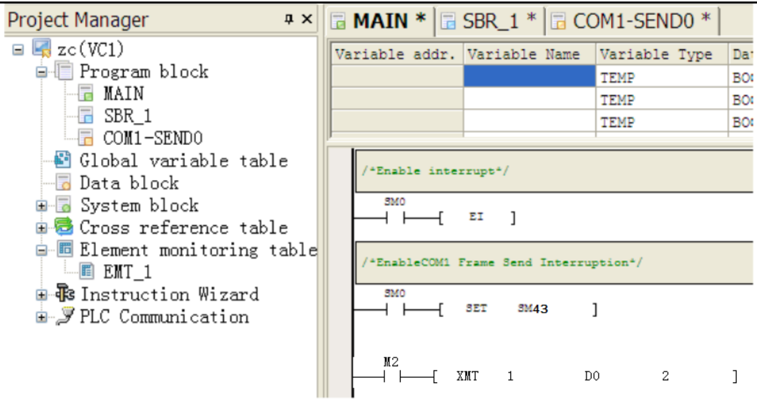
Program name: COM1-SEND0

Author:

Interrupt event: COM1 Frame Send Interruption(Interrupt n

Program description:

3. Writing the code of enabling serial port sending frame interrupt in MAIN.


Project Manager
zc(VC1)
Program block
MAIN
SBR_1
COM1-SEND0
Global variable table
Data block
System block
Cross reference table
Element monitoring table
EMT_1
Instruction Wizard
PLC Communication
MAIN *
SBR_1 *
COM1-SEND0 *

Variable addr.	Variable Name	Variable Type	Da
		TEMP	BO
		TEMP	BO
		TEMP	BO

```
/*Enable interrupt*/
 SM0
─┤ ├──[  EI  ]

/*EnableCOM1 Frame Send Interruption*/
 SM0
─┤ ├──[  SET   SM43    ]

 M2
─┤ ├──[  XMT   1         D0         2         ]
```

For details about the application instance of the serial port interrupt, refer to Chapter 10"Communication function guide"

Chapter 10 Communication function guide

This chapter detailedly introduces the communication functions of the VC series micro-PLCs, including the communication resources and communication protocols, and uses application instances to illustrate.

Chapter 10 Communication function guide 295
10.1 Communication resources 296
10.2 Programming port communication protocol 296
10.3 Free-port communication protocol 296
10.3.1 Introduction 296
10.3.2 Parameters setting of free-port........296
10.3.3 Free-port instruction 298
10.4 Modbus communication protocol 299
10.4.1 Introduction 299
10.4.2 Characteristics of links 299
10.4.3 RTU transmission mode........299
10.4.4 Supported Modbus function codes........300
10.4.5 Addressing mode of the PLC elements........300
10.4.6 Modbus slave station 301
10.4.7 Reading and writing elements........301
10.4.8 Processing of double word element 302
10.4.9 Processing of LONG INT data........302
10.4.10 Diagnostic function code 302
10.4.11 Error code 303
10.4.12 Modbus parameter setting........303
10.4.13 Modbus instruction 304
10.5 N:N communication protocol 306
10.5.1 N:N introduction 306
10.5.2 N:N network data transmission form 306
10.5.3 N:N network structure 307
10.5.4 N:N refresh mode........308
10.5.5 Enhanced refresh mode........312
10.5.6 N:N parameter setting 313
10.6 Several control strategies........315
10.6.1 Determination of the master station 315
10.6.2 Maximum number of inspection stations 315
10.6.3 Multi-master and slave (M: N) 315
10.6.4 Examples of using N:N........315

10.1 Communication resources

The baud rates applicable to VC series small PLC are listed in the following table:

Communication protocol	Applicable baud rate
Free-port protocol, Modbus protocol	115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200
N:N protocol	115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200

Communication protocols supported by the VC series micro-PLCs are listed in the following table.

Main module	Communication port	Port type	Supported protocol
VC1S (Planning)	PORT0	RS232	Programming port protocol, free-port protocol, Modbus communication protocol (slave station), and N:N communication protocol (master and slave station)
	PORT1	RS485	Free-port protocol, Modbus communication protocol (master station and slave station), and N:N communication protocol (master and slave station)
VC1	PORT0	RS232	Programming port protocol, Modbus communication protocol (slave station)
	PORT1	RS485	Free-port protocol, Modbus communication protocol (master station and slave station), and N:N communication protocol (master and slave station)
	PORT2	RS485	Free-port protocol, Modbus communication protocol (master station and slave station), and N:N communication protocol (master and slave station)
	PORT3	USB	Programming port protocol
VC2 VC3 VC5 (Planning)	PORT0	RS232	Programming port protocol, free-port protocol, Modbus communication protocol (slave station), and N:N communication protocol (master and slave station)
	PORT1	RS485	Free-port protocol, Modbus communication protocol (master station and slave station), and N:N communication protocol (master and slave station)
	PORT2	RS485	Free-port protocol, Modbus communication protocol (master station and slave station), and N:N communication protocol (master and slave station)
	PORT3	USB	Programming port protocol
	PORT4	Ethernet	Programming port protocol, Modbus_TCP(master and slave station)
	PORT5	CAN	CANopen(master and slave station)

Besides, you can set the mode selection switch of the VC series micro-PLCs (RUN-STOP) so as to forcibly transfer PORT0 into the programming port protocol.

10.2 Programming port communication protocol

The programming port protocol is an internal protocol dedicated to the communication between the PC software Auto Studio and the main module.USB, network port, and serial port can be used to communicate with the main module. Only one of them can be selected for monitoring, and they cannot be used simultaneously.

10.3 Free-port communication protocol

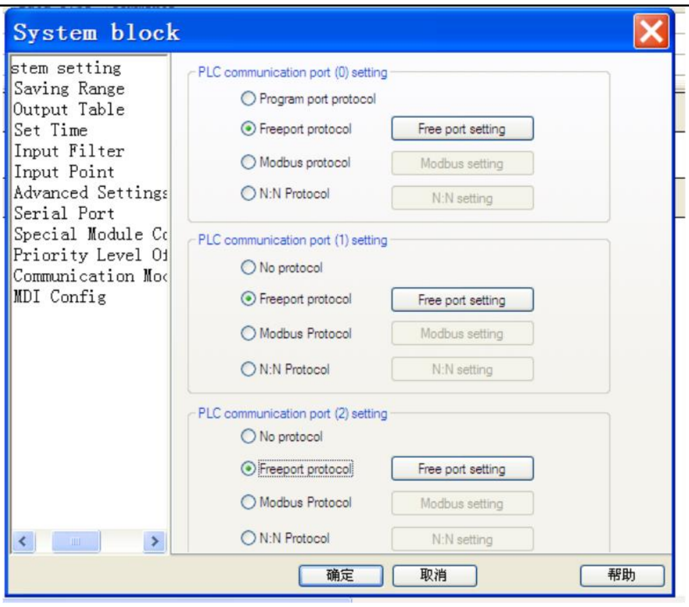
10.3.1 Introduction

The free-port protocol is a communication mode with user-defined data file format, which can realizes data sending and receiving through the instructions. It supports two data formats: ASCII and binary. The free-port communication is available only when the PLC is in RUN state.

The free-port communication instructions include XMT (free-port sending instruction) and RCV (free-port receiving instruction).

10.3.2 Parameters setting of free-port

Selecting**Communication Port** in the system block dialogue box, and selecting **Free port protocol** in he corresponding setting area to enable the Free port setting button, as shown in the following figure.


System block
stem setting
Saving Range
Output Table
Set Time
Input Filter
Input Point
Advanced Settings
Serial Port
Special Module Cc
Priority Level Oi
Communication Moc
MDI Config
Free port setting
Modbus setting
N:N setting
确定
取消
帮助

PLC communication port (0) setting

○ Program port protocol

● Freeport protocol

○ Modbus protocol

○ N:N Protocol

PLC communication port (1) setting

○ No protocol

● Freeport protocol

○ Modbus Protocol

○ N:N Protocol

PLC communication port (2) setting

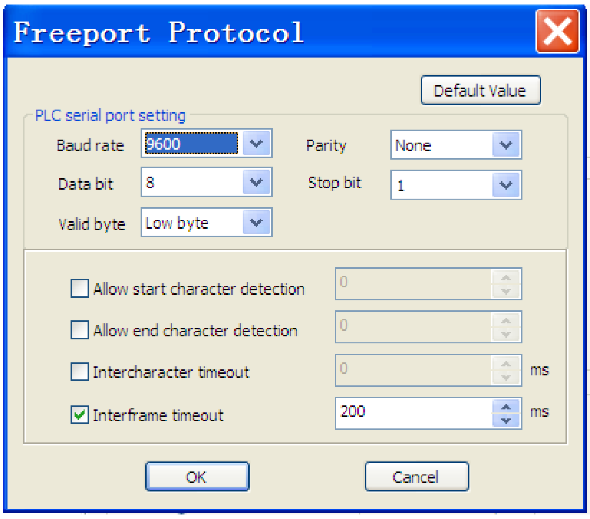
○ No protocol

● Freeport protocol

○ Modbus Protocol

○ N:N Protocol

Clicking any one of the free port setting buttons to enter the Freeport parameter setting interface, as shown in the following figure.


Freeport Protocol
Default Value
OK
Cancel

PLC serial port setting

Baud rate: 9600

Parity: None

Data bit: 8

Stop bit: 1

Valid byte: Low byte

☐ Allow start character detection: 0

☐ Allow end character detection: 0

☐ Intercharacter timeout: 0 ms

☑ Interframe timeout: 200 ms

Configurable items are listed in the following table.

Item	Setting content	Remark
Baud rate	115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200. Default: 9600	—
Data bit	7 or 8 (default)	—
Parity check	None (default), odd, even	—
Stop bit	1 (default) or 2	—
Allow the start character detection	Enabled or disabled (default)	—
Start character detection	0–255 (corresponding to 00–FF)	Start receiving after the designated start character is detected, and saving the received characters (including the start character) to the designated buffer
Allow the end character detection	Enabled or disabled (default)	—
End character detection	0–255 (corresponding to 00–FF)	Stopping receiving after the preset end character is received, and saving the end character to the buffer
Allow the interframe timeout	Enabled or disabled (default)	—
Intercharacter timeout	0–65535 ms	Stop receiving if the interval between two received characters is longer than the preset intercharacter timeout

Interframe timeout enable	Enabled or disabled (default)	—
Interframe timeout	0–65535 ms	When the energy flow is turned on and the communication conditions are met, that is, the timing starts when the communication serial port starts to receive frames. If the receiving of one frame is finished when the time is up, the receiving is terminated.

10.3.3 Free-port instruction

- Note

The free-port instructions XMT and RCV can be used to send/receive data to/from the designated communication port.For details about the usage of the free-port instructions, refer to section 6.12.11 "XMT: Free-port sending instruction" and section 6.12.12 "RCV: Free-port receiving instruction".

You need to note that to use the free-port instruction on a certain port, you need to set the free-port protocol and communication parameter for the communication port through the system block of Auto Studio. After setting, you need to download the system setting to the PLCs and restart it.

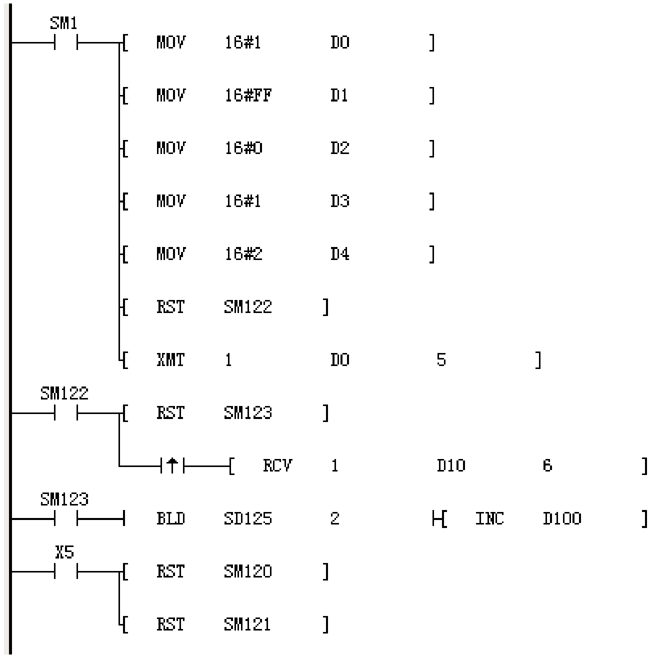
- Application instance

Example 1: Sending a 5-byte data, and then receiving a 6-byte data through PORT1.

Data to be sent:	01	FF	00	01	02		Data to received	01	FF	02	03	05	FE

Storing the received data in D elements starting from D10. Each byte occupies one D element, as shown below:

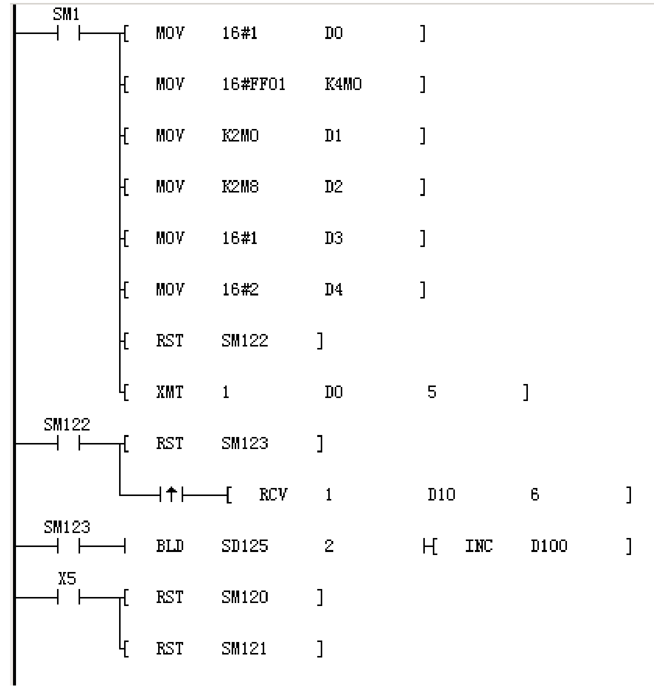
01	FF	02	03	05	FE
D10	D11	D12	D13	D14	D15



```
SM1
-| |--+-[ MOV   16#1    D0    ]
      +-[ MOV   16#FF   D1    ]
      +-[ MOV   16#0    D2    ]
      +-[ MOV   16#1    D3    ]
      +-[ MOV   16#2    D4    ]
      +-[ RST   SM122   ]
      +-[ XMT   1       D0    5    ]
SM122
-| |--+-[ RST   SM123   ]
      +--|↑|--[ RCV   1    D10    6    ]
SM123
-| |-----| BLD   SD125   2     |-[ INC   D100   ]
X5
-| |--+-[ RST   SM120   ]
      +-[ RST   SM121   ]
```

1. First, you need to change the setting of communication port in the system block to free-port communication and set the related parameters, such as baud rate and parity.
2. Storing the to-be-sent data in the communication BFM buffer starting from D0, sending data through the XMT instruction, and resetting SM122 (sending completion flag bit) before the sending when SM1 is valid.
3. Setting SM122 after the data is sent, and beginning to receive data upon the rising edge. The max. length for the received characters is 6.
4. Setting SM123 after the data is received, and performing the corresponding operations based on the receiving completion information register (SD125).
5. Using X5 as the enable bit for interrupting the sending and receiving.

Example 2: Sending the data through PORT1, and then receiving the data.



```
SM1
-| |--[ MOV   16#1     D0    ]
     [ MOV   16#FF01  K4M0  ]
     [ MOV   K2M0     D1    ]
     [ MOV   K2M8     D2    ]
     [ MOV   16#1     D3    ]
     [ MOV   16#2     D4    ]
     [ RST   SM122    ]
     [ XMT   1        D0     5    ]
SM122
-| |--[ RST   SM123    ]
     --|↑|--[ RCV   1        D10    6    ]
SM123
-| |--| BLD   SD125    2     |-[ INC   D100   ]
X5
-| |--[ RST   SM120    ]
     [ RST   SM121    ]
```

Different from "**Example 1**", before sending the high and low bytes of a word element, the word element needs to be divided into two parts: high and low bytes.
For instance, if you want to send the content of D2, you can store its high bytes and low by tesseparately in D3 and D4, and then send D3 and D4. You can also store the data in a K4MX (such as K4M0 in this program)element, and then take K2M0 as the high byte and K2M8 as the low byte.

10.4 Modbus communication protocol

10.4.1 Introduction

For the serial communication of the VC series micro-PLCs, Modbus communication protocol is available. RTU mode is supported. The PLCs can be set as the master or slave station.

10.4.2 Characteristics of links

1. Physical layer: RS-232 and RS-485

2. Link layer: asynchronous transmission mode

(1) Data bit: 8 bits (RTU)

(2) Transmission rate: 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200

(3) Parity method: even , odd, or none

(4) Stop bit: 1 or 2 stop bits

3. Networking configuration: a maximum of 31 sets of equipments. Address range: 1﹣247. Broadcast is supported.

10.4.3 RTU transmission mode

1. Hex data.

2. The interval between two characters shall not be less than the time of 1.5 characters.

3. There is no frame head or tail, and the interval between two frames is at least the time of 3.5 characters.

4. Using the CRC16 check.

5. The max. length of the RTU frame is 256 bytes and the structure of a frame is shown as follows:

Structure of a frame	Address	Function code	Data	CRC
Number of bytes	1	1	0–252	2

6. Calculation of interval among characters:

If the communication baud rate is 19200, the interval of 1.5 characters is 1/19200×11×1.5×1000 = 0.86ms.

The interval of 3.5 characters is 1/19200×11×3.5×1000 = 2ms.

10.4.4 Supported Modbus function codes

Supported Modbus function codesinclude 01, 02, 03, 05, 06, 08, 15, and 16.

10.4.5 Addressing mode of the PLC elements

1. Mapping between read-write element function codes and the elements:

Function code	Name of function code	Modicon data address	Type of operable element	Remark
01	Read coils	0[note 1]:xxxx	Y, X, M, SM, S, T, and C	Bit read
02	Read discrete input	1[note 2]:xxxx	X	Bit read
03	Read registers	4[note 3]:xxxx[note 4]	D, SD, Z, T, C, and R	Word read
05	Write single coil	0:xxxx	Y, M, SM, S, T, and C	Bit write
06	Write single register	4:xxxx	D, SD, Z, T, C, and R	Word write
15	Write multiple coils	0:xxxx	Y, M, SM, S, T, and C	Bit write
16	Write multiple registers	4:xxxx	D, SD, Z, T, C, and R	Word write

Notes:
1. 0 indicates "coil".
2. 1 indicates "discrete input".
3. 4 indicates "register".
4. xxxx indicates range "1–9999". Each type has an independent logic address range from 1 to 9999 (protocol address starts from 0).
5. 0, 1 and 4 do not have the physical meaning and are not involved in actual addressing.
6. Users shall not write X element with function codes 05 and 15, otherwise, the system does not feed back the error information if the
written operands and data are correct, but the system does not perform any operation on the write instruction.

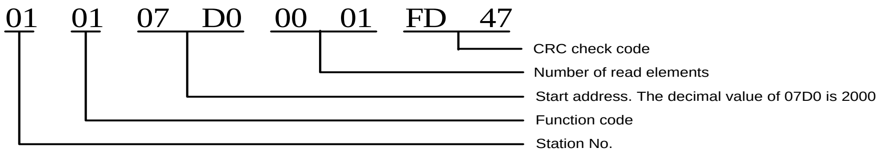
2. Mapping between the PLC elements and Modbus communication protocol address:

Element	Type	Physical element	Protocol address	Supported function code	Remark
Y	Bit	Y0–Y777 (octal code) 512 points in total	0000–0511	01, 05, and 15	Output state, element number: Y0–Y7 and Y10–Y17
X	Bit	X0–X777 (octal code) 512 points in total	1200–01711	01, 02, 05 and 15	Input state, it supports two kinds of addresses, and the element number is same as the above
M	Bit	M0–M2047 M2048–M10239	2000–4047 12000-20191	01, 05, and 15	
SM	Bit	SM0–SM255 SM256–SM1023	4400–4655 30000-30767	01, 05, and 15	
S	Bit	S0–S1023 S1024–S4095	6000-7023 31000-34071	01, 05, and 15	
T	Bit	T0–T255 T256–T511	8000–8255 11000-11255	01, 05, and 15	State of T element
C	Bit	C0–C255 C256–C511	9200–9455 10000-10511	01, 05, and 15	State of C element
D	Word	D0–D7999	0000–7999	03, 06, and 16	
SD	Word	SD0–SD255 SD256–SD1023	8000–8255 12000-12767	03, 06, and 16	
Z	Word	Z0–Z15	8500–8515	03, 06, and 16	
T	Word	T0–T255 T256–T511	9000–9255 11000-11255	03, 06, and 16	Current value of T element
C	Word	C0–C199	9500–9699	03, 06, and 16	Current value of C element (INT)
C	Double word	C200–C255	9700–9811	03 and 16	Current value of C element (DINT)

Element	Type	Physical element	Protocol address	Supported function code	Remark
C	Double word	C256–C306	10000-10101	03 and 16	Current value of C element (DINT)
R	Word	R0–R32767	13000-45767	03, 06, and 16	

Note:

The protocol address is the address used on data transmission and corresponds with the logic address of Modicon data. The protocol address starts from 0 while the logic address of Modicon data starts from 1, that is, protocol address + 1 = logic address of Modicon data. For example, if M0 protocol address is 2000, and its corresponding logic address of Modicon data is 0:2001. In practice, the read and write of M0 is completed through the protocol address, for example, frame for reading M0 element (sent by the master station).



10.4.6 Modbus slave station

Modbus slave station responds to the master station according to the received message of local addressing, rather than sending out message actively. The slave station only supports Modbus function codes 01, 02, 03, 05, 06, 08, 15, and 16, and the other codes are "illegal function codes"(except broadcast frame).

10.4.7 Reading and writing elements

All the function codes supported except 08 are used for reading and writing elements. In principle, in one frame, there are 2000 bit elements and 125 word elements for reading, 1968 bit elements and 120 word elements for writing at most. However, theactual protocol addresses for different types of elements are different and discontinuous (for example, Y377's protocol addressis 255 while X0's protocol address is 1200). Therefore, when reading or writing an element, the element read at one time can only be the same type, and the max. number of the read elements depends on the number of the elements thatare actually defined. For example, when reading Y element (Y0 – Y377, 256 points in total), the protocol address ranges from 0 to 255, the corresponding logic address of Modicon data is from 1 to 256, and the max. number of the Y elements that can be readis 256.

The examples are as follows:

1. Sending from the master station: 01 01 00 00 01 00 3D 9A

01 – address; 01- function code; 00 00 – starting address; 01 00 – number of read elements; 3D 9A – check

Response of the slave station: providing correct response

2. Sending from the master station: 01 01 00 00 01 01 FC 5A

The master station reads 01 01 (257) elements starting from 0000, which is beyond the defined number of Y elements.

Response of the slave station: 01 81 03 00 51

The response of the slave station is illegal data value, because 257>256, and 256 is the allowed max. number of Y elements.

3. Sending from the master station: 01 01 00 64 00 A0 7D AD

00 64 (decimal 100) is the start address for themaster station to read, and 00 A0 (decimal 160) is the number of elements.

Response of the slave station: 01 81 02 C1 91

The response of the slave station is illegal data address, because there are only 156 Y elements which are defined to the protocol address 100, but 160>156, so 160 is illegal.

4. Sending from the master station: 01 04 00 02 00 0A D1 CD

The master station sends the frame of function code 04

Response of the slave station: 01 84 01 82 C0

The response of the slave station is illegal function codes.

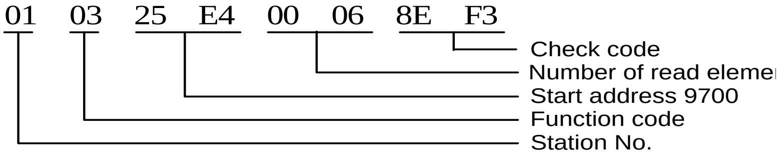
🕮 Note

1. X element does not support the write operation (that is, the write of element X is invalid). For details about the writable properties of SM and SDelements, refer to Appendix A "Special auxiliary relay" and Appendix B "Special data register" (if the element is unwritable,the write operation is invalid).
2. The address of the slave station is 01, the last two bytes are CRC check codes and the second byte is function code.

10.4.8 Processing of double word element

The current count value of C element is word or double word element. The values from C200 to C255 are double word elements, which are read and written through the function codes (03 and 16) of read/write registers. The address of every two registers corresponds to one C double word element, and the registers can only be read or written in pair.

For example, reading the RTU frame of three C double word elements from C200 to C202.



In the returned data, two addresses 9700 and 9701 indicate the contents of C200. 9700 is MSB, and 9701 is LSB.

When reading a double word element, if the start address for reading is not an even number, then the system responds with errorcode of illegal address. If the number of the read registers is not an even number, the system responds with error code of illegal data.

For example:

Sending from the master station: 01 03 25 E5 00 04 5E F2

The start address for the reading of master station: 4 word elements of 25 E5 (decimal 9701)

Response of the slave station: 01 83 02 C0 F1

Response of the slave station: illegal data address

Sending from the master station: 01 03 25 E4 00 05 CE F2

The start address for the master station reading: 5 word elements of 25 E5

Response of the slave station: 01 83 03 01 31

The slave station returns the illegal data.

10.4.9 Processing of LONG INT data

One LONG INT data can be saved in two D elements. For example, if a LONG INT data is saved in D3 and D4, the VEICHI PLC thinks that MSB are stored in D3 and LSB are stored in D4. When the master station reads the LONG INT data through Modbus, the 32-bit data shall be regrouped based on the LONG INT storage principle of VEICHI PLC after reading the data.The storage principle of FLOAT is the same as the storage principle of LONG INT.

10.4.10 Diagnostic function code

The diagnostic function code is used for testing the communication between the master and slave station, and various internal error conditions of the slave station. The supported diagnostic subfunction codes are shown in the following table:

Function code	Subfunction code	Name of the subfunction code		Function code	Subfunction code	Name of the subfunction code
08	00	Return the query data		08	12	Return the bus communication error count
08	01	Restart the communication option		08	13	Return the bus exceptional error count
08	04	Forced listen only mode		08	14	Return the slave message count
08	10	Clear the counter		08	15	Return the slave station no response count
08	11	Return the bus message count		08	18	Return the bus character overrun count

Applicable to the VC2/VC3 series

10.4.11 Error code

When the instruction sent from the master station is in the normal response state, the slave station returns data or statistic value in the data field. But in the abnormal response state, the server returns error codes in the data field. The error codes are shown in the following table.

Error code	Meaning of error code
0x01	Error function code
0x02	Error register address
0x03	Error data

In addition, if the slave station receives data under the following situations, no message is returned:

(1)When there are errors in the broadcast frame, such as data error, address error, etc.

(2)When the character is overrun, no message is returned. For example, the RTU frame is more than 256 bytes;

(3)When in the RTU transmission mode, the interval between characters times out, which is equivalent to receiving an error frame,and no message is returned;

(4) When the slave station is in the listen-only mode, no message is returned.

(5)The slave station received ASCII error frame, including frame tail error and character range error.

📖 Note

Read station is configured with compulsory elements. What is read is the value run by the program, which may be inconsistent with the compulsory value.

10.4.12 Modbus parameter setting

- Setting the communication port in the system block
  There are two serial ports (PORT0 and PORT1) on the communication port interface. PORT0 only supports the Modbus slave station while PORT1 and PORT 2 supports both the Modbus master and slave stations.

- Setting the Modbus communication protocol parameters
  There is a button of default value on the Modbus operand interface. The default value is the communication setting recommended by the Modbus communication protocol. For details about the parameter setting items, you can refer to the following table.

Item	Setting content
Station No.	0–247
Baud rate	115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200
Data bit	Setting to 7 or 8 bits:7 bits for ASCII mode, and 8 bits for RTU mode
Parity check	Setting to none, odd, and even
Stop bit	Setting to 1 or 2: 1 for odd or even check, and 2 for none.
Modbus master/slave	It can be set to master or slave station: PORT1 can be set to master/slave station, and PORT0 can only be set to slave station
Transmission mode	Selecting RTU or ASCII mode

Item	Setting content
Timeout time of the master mode	The time for waiting the slave response by the master station exceeds the preset time
Note: After the operand is set and downloaded in the system block, it is valid only after one operation.	

10.4.13 Modbus instruction

When a PLC is used as a Modbus master station, the Modbus data frame can be sent/received through the Modbus instruction provided by the system. For details about the use of the Modbus instruction, refer to section 6.12.1 "Modbus: Master station communication instruction".

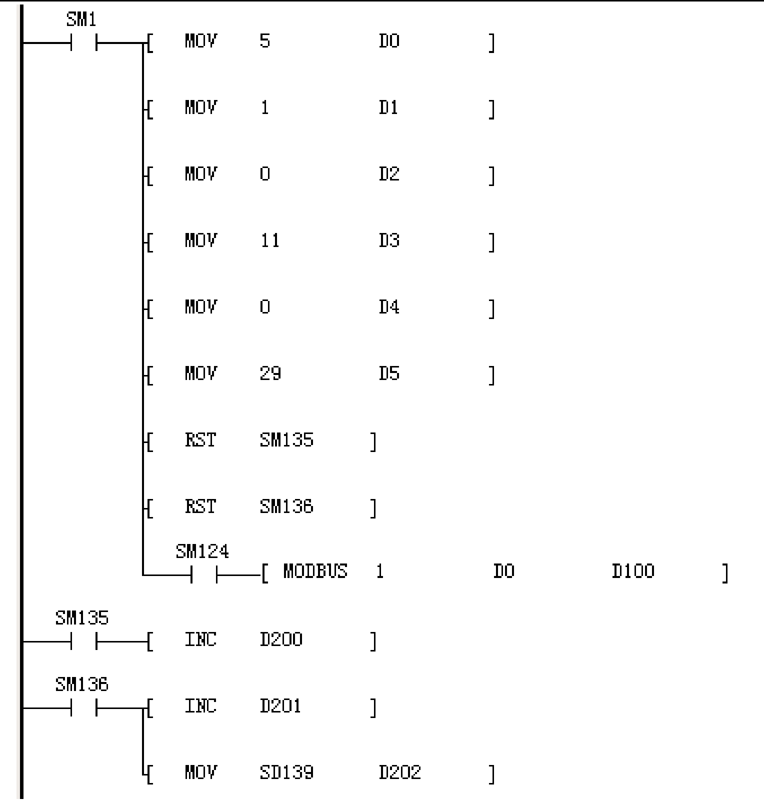
If a PLC is set to be a master station, there is a timeout item in the master mode when setting the Modbus parameter in the system block. To ensure the correctness of the received data, the timeout time shall be longer than a scan cycle of the Modbus slave station and with reasonable margin. For example, if a VC1 series PLC is used as a slave station, and a scan cycle of the VC1 series PLCs is 300ms,the master mode timeout of the master station shall be over 300ms. It is proper to set the timeout to be 350ms.

Application program

Example 1: When a VC1 series PLC is a Modbus master and slave station, it reads the bit element state of 5# station. The protocol address of the slave station read by the master station is the bit values ranging from 11 to 39. Assuming that the read data are as follows, the storage location for the received data starts from D100, D100 saves the address, D101 saves the function code, D102 saves the number of registers, and the units starting from D103 save the read bit element values.

42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27
X	X	X	0	0	1	1	0	0	0	0	1	0	1	1	0
D106								D105							
26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
1	1	0	0	0	1	1	0	1	1	1	0	0	0	0	1
D104								D103							

When the number of registers read is not a multiple of 8, the insufficient upper bits are filled with zero. In the example, the three upper bits (40, 41, and 42) of D106 are filled with zeros.



```
SM1
[ MOV 5 D0 ]
[ MOV 1 D1 ]
[ MOV 0 D2 ]
[ MOV 11 D3 ]
[ MOV 0 D4 ]
[ MOV 29 D5 ]
[ RST SM135 ]
[ RST SM136 ]
SM124
[ MODBUS 1 D0 D100 ]
SM135
[ INC D200 ]
SM136
[ INC D201 ]
[ MOV SD139 D202 ]
```

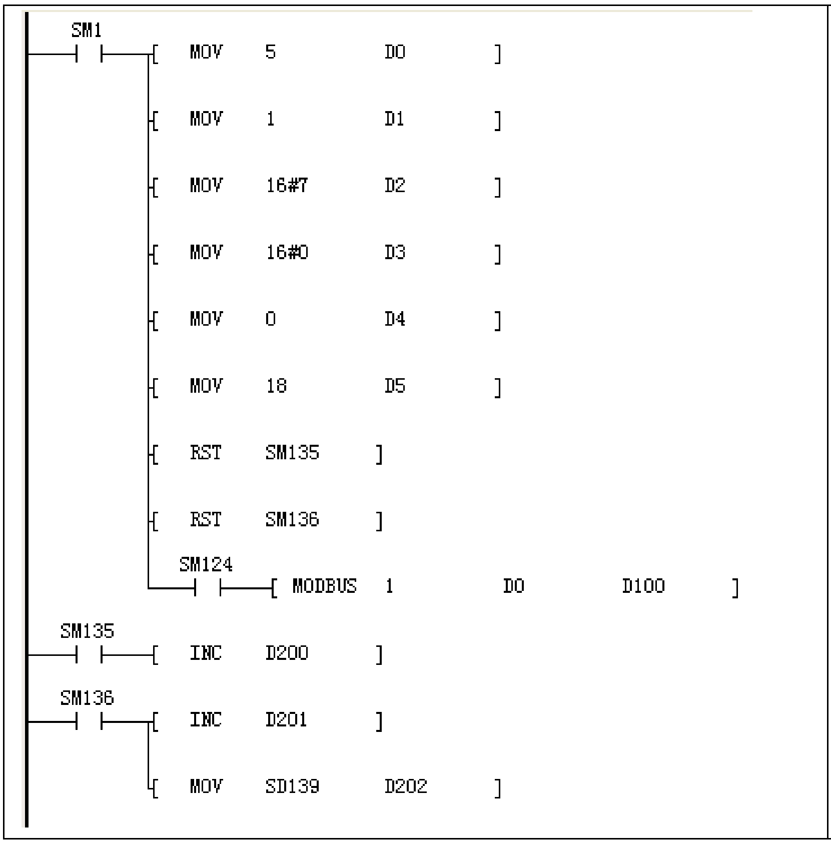
1. Designating 5 as the address of the slave station
to be accessed (it is saved to D0).
2. Designating 1 as the function code (it is saved to D1).
3. The address of the register to be read is 11 (it is saved to D2/D3 according to MSB and LSB).
4. The number of the registers to be read is 29 (it is saved to D4/D5 according to MSB and LSB).
5. The received data is saved to D100.
6. Adding 1 to D200 if the receiving is completed (SM135 is set).
7. Adding 1 to D201, and saving the error code to D202 if the communication fails (SM136 is set).
8. SM124 is the idle flag of the communication port.

Note

1. When the logic address is used for addressing the bit elements of the VC1 series PLCs, the logic address 1 is the protocol address 0. In the above example, to read the values of 11–39 bit elements (protocol address) in the slave station, the logic address shall be started from 12.
2. The failure of this communication does not affect the next communication, that is, if there are two Modbus XMT instructions inone user program, the first communication fails and has error code, it does not affect the data sending of the second Modbus instruction. Thus, in the example, the error code in SD139 is placed in D202, so the code can be observed through D202.
3. For the message sending of the slave station, if the master station is in listen-only mode, there is no data to be returned and the system sets the error flag. Therefore, when you use the Modbus network of the VC1 series, if the VC1 is used as the master station, the user shall clearly know which PLC slave station is under listen-only mode, so as to ensure that the failure of the communication is not caused by the listen-only mode of the slave station.

Example 2: When an VC1 series PLC is a Modbus master and slave station, it reads the states of the bit elements 2000–2017 of 5# station protocol address.

The read datas are as follows. The received frame starts from D100, D100 saves the address, D101 saves the function code, D102 saves the number of registers, and the units starting from D103 save the read bit element values.

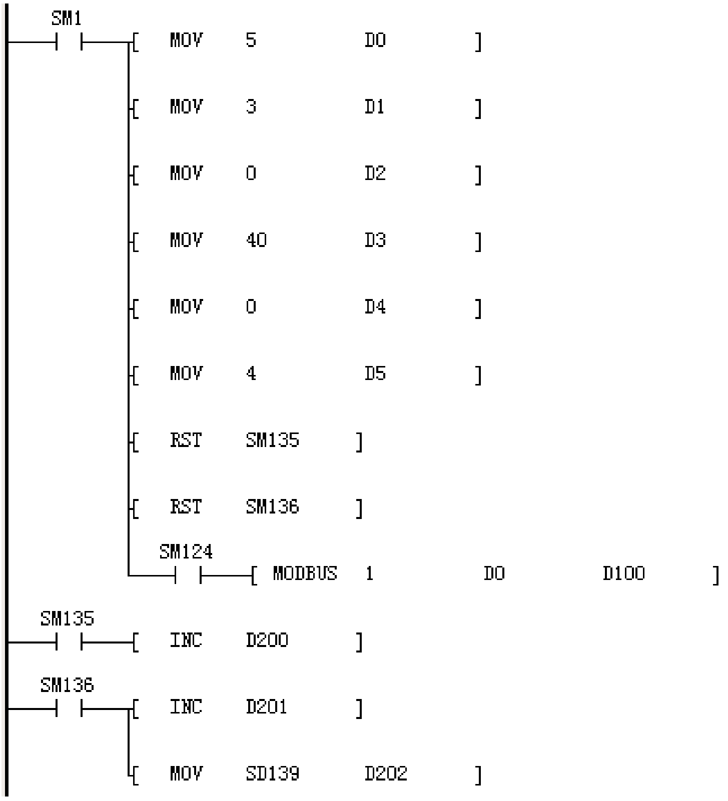


```
SM1
--| |--[ MOV     5        D0     ]
       [ MOV     1        D1     ]
       [ MOV     16#7     D2     ]
       [ MOV     16#0     D3     ]
       [ MOV     0        D4     ]
       [ MOV     18       D5     ]
       [ RST     SM135  ]
       [ RST     SM136  ]
        SM124
       --| |--[ MODBUS  1   D0   D100  ]
SM135
--| |--[ INC     D200   ]
SM136
--| |--[ INC     D201   ]
       [ MOV     SD139    D202   ]
```

1. Designating 5 as the address of the slave station to be accessed (it is saved to D0).
2. Designating 1 as the function code (it is saved to D1).
3. The start address of the register to be read is 07D0 (hex) (it is saved to D2/D3 according to MSB and LSB).
4. The number of the registers to be read is 29 (it is saved to D4/D5 according to MSB and LSB).
5. The received data is saved to D100.
6. Adding 1 to D200 if the receiving is completed (SM135 is set).
7. Adding 1 to D201, and saving the error code to D202 if the communication fails (SM136 is set).
8. SM124 is the idle flag of the communication port.

Example 3: When an VC1 series PLC is a Modbus master and slave station, it reads the values of the word elements 40–43 of 5# station protocol address.

MSB of element 40	LSB of element 40	MSB of element 41	LSB of element 41	MSB of element 42	LSB of element 42	MSB of element 43	LSB of element 43
D103	D104	D105	D106	D107	D108	D109	D110

 SM1 [MOV 5 D0] [MOV 3 D1] [MOV 0 D2] [MOV 40 D3] [MOV 0 D4] [MOV 4 D5] [RST SM135] [RST SM136] SM124 [MODBUS 1 D0 D100] SM135 [INC D200] SM136 [INC D201] [MOV SD139 D202]	1. Designating 5 as the address of the slave station to be accessed (it is saved to D0). 2. Designating 3 as the function code (it is saved to D1). 3. The start address of the register to be read is 40 (it is saved to D2/D3 according to MSB and LSB). 4. The number of the registers to be read is 4. 5. The received data is saved to D100. 6. Adding 1 to D200 if the receiving is completed (SM135 is set). 7. Adding 1 to D201 and saving the error code to D202 if the communication fails (SM136 is set). 8. SM124 is the idle flag of the communication port.

10.5 N:N communication protocol

10.5.1 N:N introduction

N:N is a micro-PLC network developed by Suzhou Veichi Electric Co.,Ltd. The physical layer of N:N adopts RS485, so the PLCs can be directly connected through PORT1 or connected through PORT0 by a RS-232/RS-485 converter to N:N network. The PLCs that are connected to N:N can automatically exchange thevalues between certain D and M elements, which makes the access to the other PLC elements in the network as simple and convenient as accessing its own element. In N:N, the data access between the PLCs is completely equivalent (N:N communication network).

It is convenient to configure N:N. Most parameters of N:N only need to be configured for No.0 PLC. In addition,N:N supports the online modification of the network parameters, and is able to detect the newly added PLCs automatically in the network. If any PLC is disconnected from the network, the other PLCs continue to exchange the data. It isalso able to monitor the communication state of the whole network through the relevant SM elements of any PLC inN:N.

10.5.2 N:N network data transmission form

There are two kinds of messages in N:N tokens issued by the master station, and broadcast by each PLC to its own data.

The token is uniformly issued by the master station. The master station first holds the token. After the data is broadcasted, the token is cyclically and sequentially issued to each slave station. Only the slave station that receives the token can broadcast to other PLCs (including the master station).

Figure 10-1–Figure 10-5 show the main processes of network communication. In the figure, 1# stands as the master station. It needs to be noted that in general, 0# is amaster station by default while 1# is a standby master station (when the master station communication error or power failure occurs, it is switched to the master station).

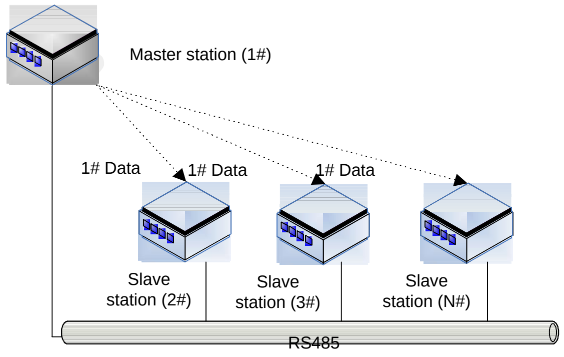

Master station (1#)
1# Data
1# Data
1# Data
Slave station (2#)
Slave station (3#)
Slave station (N#)
RS485

Figure 10-1 Master station broadcasts

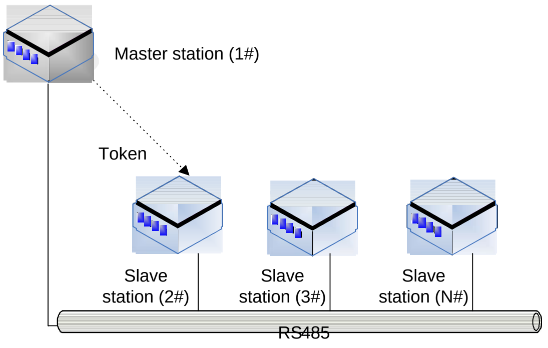

Master station (1#)
Token
Slave station (2#)
Slave station (3#)
Slave station (N#)
RS485

Figure 10-2 Master station sends the token to 2# slave station

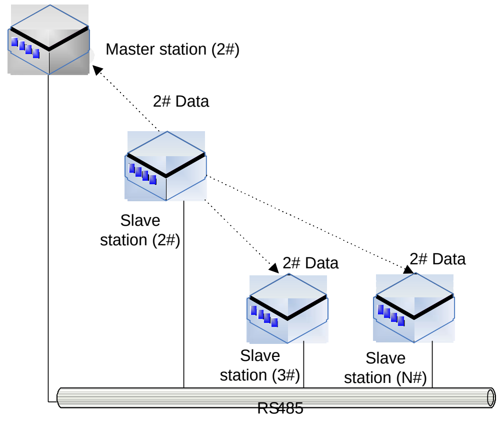

Master station (2#)
2# Data
Slave station (2#)
2# Data
2# Data
Slave station (3#)
Slave station (N#)
RS485

Figure 10-3 2＃slave station broadcasts

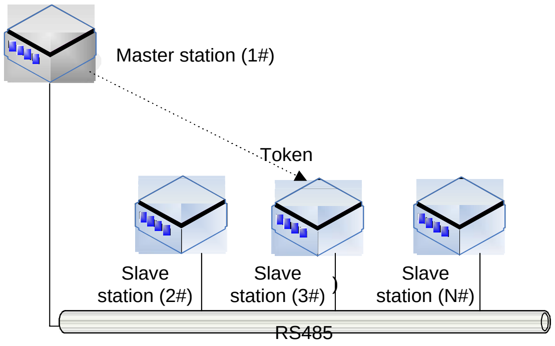

Master station (1#)
Token
Slave station (2#)
Slave station (3#)
Slave station (N#)
RS485

Figure 10-4 Master station sends the token to 3# slave station

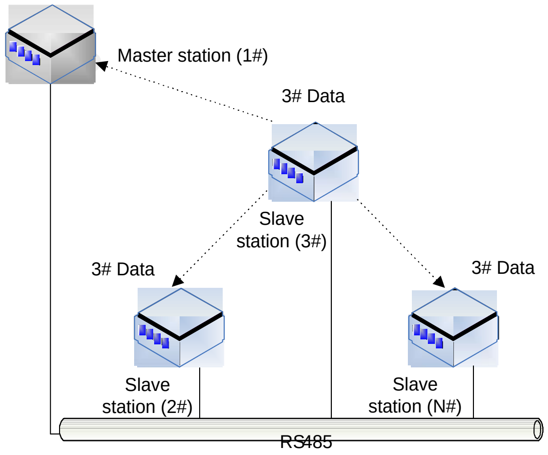

Master station (1#)
3# Data
Slave station (3#)
3# Data
3# Data
Slave station (2#)
Slave station (N#)
RS485

Figure 10-5 3＃slave station broadcasts

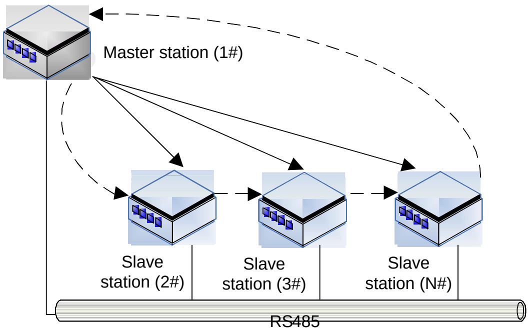

Master station (1#)
Slave station (2#)
Slave station (3#)
Slave station (N#)
RS485

Figure 10-6 Token issue and flow orders

Figure 10-6 shows the flow order of tokens. The heavy line indicates the actual process of issuing the token, and the dashed line indicates the order of the station in which the token is held and broadcasted. You need to note that the token is not passed by one slave station(such as 2#PLC) to another slave station (such as 3#PLC), but the token is first issued by the master station to 2#PLC, and then issued by the master station to 3# PLC.

10.5.3 N:N network structure

N:N supports two kinds of networks: single-layer network and multiple-layer network, as shown in the following figures.

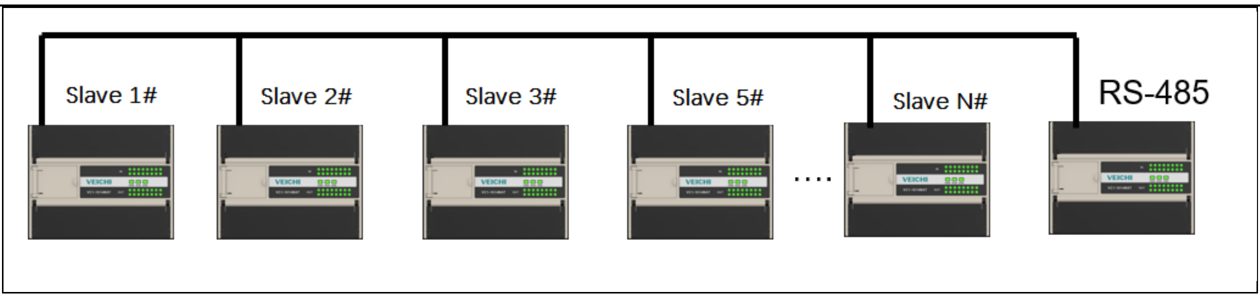

Slave 1#
Slave 2#
Slave 3#
Slave 5#
Slave N#
RS-485
....

Figure10-7 N:N single-layer network

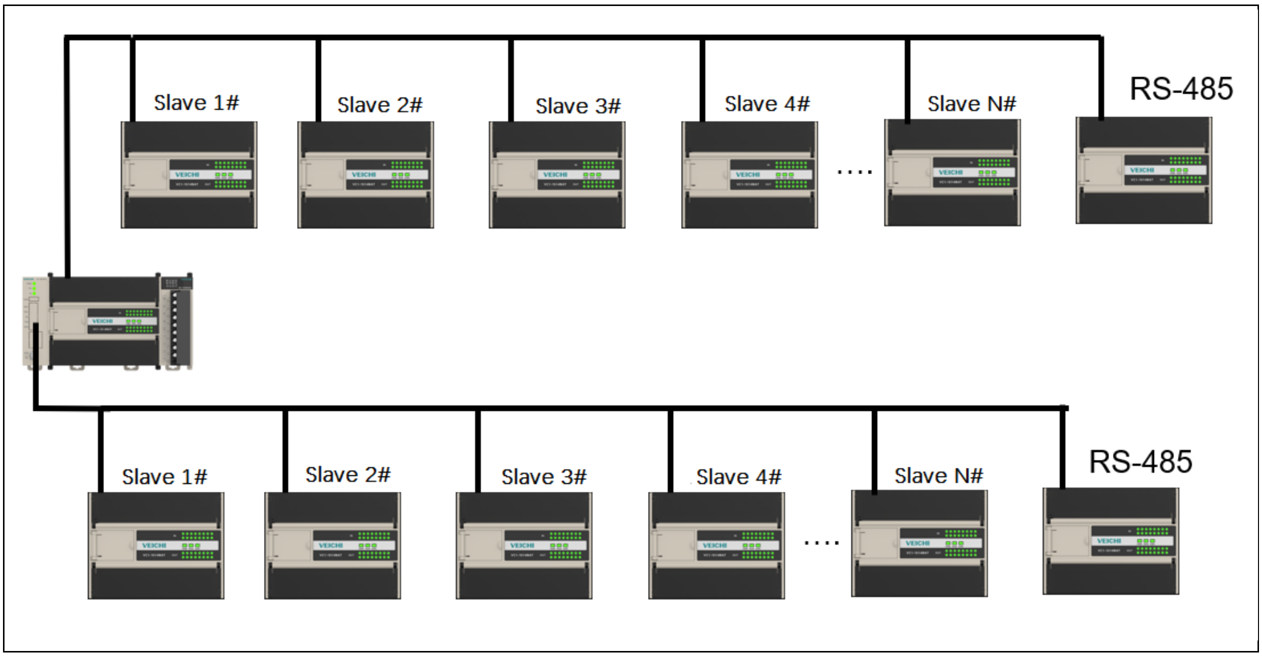
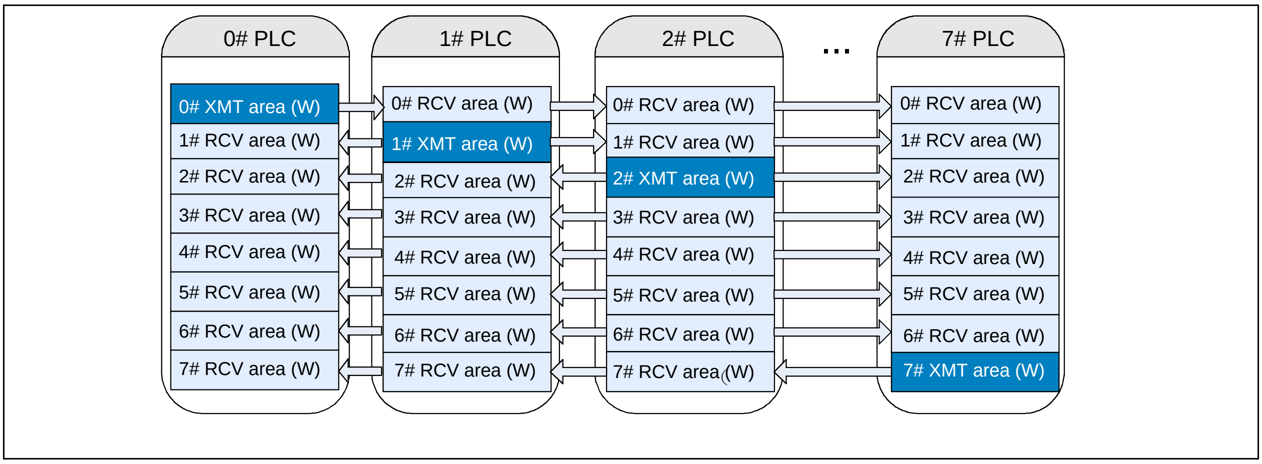

Slave 1#
Slave 2#
Slave 3#
Slave 4#
Slave N#
RS-485
....
Slave 1#
Slave 2#
Slave 3#
Slave 4#
Slave N#
RS-485
....

Figure10-8 N:N multiple-layer network

In the single-layer network, each PLC is connected to N:N through only one port. In the multi-layer network, the layer-to-layer PLC (intermediate node) shall be connected, and the two communication ports of the PLC are respectively connected to different layers. The single-layer network can support a maximum of 32 PLCs, and each layer of the multi-layer network can support a maximum of 16 PLCs.

10.5.4 N:N refresh mode

The PLCs connected to N:N can automatically exchange between parts of D elements and M elements in the network. The quantity and No. of these D and M elements are fixed, and these elements are called "Elements Sharing Area". If the PLC uses N:N, the value of the Elements Sharing Area keeps refreshing automatically, so as to keep the value consistency of the Elements Sharing Area for each PLC in the network.


0# PLC
0# XMT area (W)
1# RCV area (W)
2# RCV area (W)
3# RCV area (W)
4# RCV area (W)
5# RCV area (W)
6# RCV area (W)
7# RCV area (W)
1# PLC
0# RCV area (W)
1# XMT area (W)
2# RCV area (W)
3# RCV area (W)
4# RCV area (W)
5# RCV area (W)
6# RCV area (W)
7# RCV area (W)
2# PLC
0# RCV area (W)
1# RCV area (W)
2# XMT area (W)
3# RCV area (W)
4# RCV area (W)
5# RCV area (W)
6# RCV area (W)
7# RCV area (W)
...
7# PLC
0# RCV area (W)
1# RCV area (W)
2# RCV area (W)
3# RCV area (W)
4# RCV area (W)
5# RCV area (W)
6# RCV area (W)
7# XMT area (W)

As shown in the above figure, each PLC connected to N:N has a writable sending area in the Element Sharing Area. N:N automatically sends the information (values of designated D and M elements) of the writable sending area to other PLCs, receives the information from other PLCs, and saves it to the read-only sending area.

The number of elements in the Element Sharing Area is fixed (a total of 64 D components and 512 M components can be shared), and these elements are distributed to more than one PLC. Therefore, the fewer PLCs are connected to the network, the more elements can be distributed to each PLC. This relationship is defined by N:N refresh mode table.

- Distribution of D element on N:N single-layer network

Distribution of D elements in sending area	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
D7700–D7701	#0	#0	#0	#0	#0
D7702–D7703	#1				
D7704–D7705	#2	#1			
D7706–D7707	#3				
D7708–D7709	#4	#2	#1		
D7710–D7711	#5				
D7712–D7713	#6	#3			
D7714–D7715	#7				
D7716–D7717	#8	#4	#2	#1	
D7718–D7719	#9				
D7720–D7721	#10	#5			
D7722–D7723	#11				
D7724–D7725	#12	#6	#3		
D7726–D7727	#13				
D7728–D7729	#14	#7			
D7730–D7731	#15				
D7732–D7733	#16	#8	#4	#2	#1
D7734–D7735	#17				
D7736–D7737	#18	#9			
D7738–D7739	#19				
D7740–D7741	#20	#10	#5		
D7742–D7743	#21				
D7744–D7745	#22	#11			
D7746–D7747	#23				
D7748–D7749	#24	#12	#6	#3	
D7750–D7751	#25				
D7752–D7753	#26	#13			
D7754–D7755	#27				
D7756–D7757	#28	#14	#7		
D7758–D7759	#29				
D7760–D7761	#30	#15			
D7762–D7763	#31				

For example:

(1) In mode 1, the D elements distributed to the sending zone assigned by 0# station are D7700–D7701. The 0# station PLC can write values to D7700 and D7701, and other stations (1#-–31#) can directly read the values of D7700 and D7701.

(2) In mode 2, the D elements distributed to the sending zone assigned by 0# station are D7700–D7703. The 0# station PLC can write values to D7700, D7701, D7702, D7703, and other stations (1#-–15#) can directly read the values of D7700–D7704.

- Distribution of M element on N:N single-layer network

Distribution of M elements in sending area	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
M1400–M1415	#0	#0	#0	#0	#0
M1416–M1431	#1				
M1432–M1447	#2	#1			

Distribution of M elements in sending area	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
M1448–M1463	#3				
M1464–M1479	#4	#2	#1		
M1480–M1495	#5				
M1496–M1511	#6	#3			
M1512–M1527	#7				
M1528–M1543	#8	#4	#2	#1	
M1544–M1559	#9				
M1560–M1575	#10	#5			
M1576–M1591	#11				
M1592–M1607	#12	#6	#3		#0
M1608–M1623	#13				
M1624–M1639	#14	#7			
M1640–M1655	#15				
M1656–M1671	#16	#8	#4	#2	#1
M1672–M1687	#17				
M1688–M1703	#18	#9			
M1704–M1719	#19				
M1720–M1735	#20	#10	#5		
M1736–M1751	#21				
M1752–M1767	#22	#11			
M1768–M1783	#23				
M1784–M1799	#24	#12	#6	#3	
M1800–M1815	#25				
M1816–M1831	#26	#13			
M1832–M1847	#27				
M1848–M1863	#28	#14	#7		
M1864–M1879	#29				
M1880–M1895	#30	#15			
M1896–M1911	#31				

For example:

(1) In mode 1, the Melements distributed to the sending zone assigned by 0# station are M1400﹣M1415. The 0# station PLC can write values to M1400–M1415, and other stations (1#-﹣31#) can directly read the values of M1400﹣M1415.

(2) In mode 2, the Melements distributed to the sending zone assigned by 0# station are M1400–M1431. The 0# station PLC can write values to M1400–M1431, and other stations (1#-﹣31#) can directly read the values of M1400﹣M1431.

- Distribution of D element on N:N multiple-layer network (layer 0)

Distribution of D elements in sending area	Mode 6	Mode 7	Mode 8	Mode 9
D7700–D7701	#0	#0	#0	#0
D7702–D7703	#1			
D7704–D7705	#2	#1		
D7706–D7707	#3			
D7708–D7709	#4	#2	#1	
D7710–D7711	#5			
D7712–D7713	#6	#3		
D7714–D7715	#7			
D7716–D7717	#8	#4	#2	#1
D7718–D7719	#9			
D7720–D7721	#10	#5		
D7722–D7723	#11			
D7724–D7725	#12	#6	#3	
D7726–D7727	#13			
D7728–D7729	#14	#7		
D7730–D7731	#15			

For example:

In mode 6, the D elements distributed to the sending zone assigned by 0# station are D7700–D7701. The 0# station PLC can write values to D7700–D7701, and other stations (1#-–15#) can read directly the value of D7700–D7701.

- Distribution of D elements on N:N multiple-layer network (layer 1)

Distribution of D elements in sending area	Mode 10	Mode 11	Mode 12	Mode 13
D7732–D7733	#0	#0	#0	#0
D7734–D7735	#1			
D7736–D7737	#2	#1		
D7738–D7739	#3			
D7740–D7741	#4	#2	#1	
D7742–D7743	#5			
D7744–D7745	#6	#3		
D7746–D7747	#7			
D7748–D7749	#8	#4	#2	#1
D7750–D7751	#9			
D7752–D7753	#10	#5		
D7754–D7755	#11			
D7756–D7757	#12	#6	#3	
D7758–D7759	#13			
D7760–D7761	#14	#7		
D7762–D7763	#15			

For example:

In mode 10, the D elements distributed to the sending zone assigned by 0# station (layer 1) are D7732–D7733. The 0# station PLC can write values to D7732–D7733, and other stations (1#-–15#) can read directly the value of D7732–D7733.

- Distribution of M elements on N:N multiple-layer network (layer 0)

Distribution of M elements in sending area	Mode 6	Mode 7	Mode 8	Mode 9
M1400–M1415	#0	#0	#0	#0
M1416–M1431	#1			
M1432–M1447	#2	#1		
M1448–M1463	#3			
M1464–M1479	#4	#2	#1	
M1480–M1495	#5			
M1496–M1511	#6	#3		
M1512–M1527	#7			
M1528–M1543	#8	#4	#2	#1
M1544–M1559	#9			
M1560–M1575	#10	#5		
M1576–M1591	#11			
M1592–M1607	#12	#6	#3	
M1608–M1623	#13			
M1624–M1639	#14	#7		
M1640–M1655	#15			

For example:

In mode 6, the M elements distributed to the sending zone assigned by 0# station (layer 0) are M1400–M1415. The 0# station PLC can write values to M1400–M1415, and other stations (1#-–15#) can read directly the value of M1400–M1415.

- Distribution of M elements on N:N multiple-layer network (layer 1)

Distribution of M elements in sending area	Mode 10	Mode 11	Mode 12	Mode 13
M1656–M1671	#0	#0	#0	#0
M1672–M1687	#1			
M1688–M1703	#2	#1		
M1704–M1719	#3			
M1720–M1735	#4	#2	#1	
M1736–M1751	#5			
M1752–M1767	#6	#3		
M1768–M1783	#7			
M1784–M1799	#8	#4	#2	#1
M1800–M1815	#9			
M1816–M1831	#10	#5		
M1832–M1847	#11			
M1848–M1863	#12	#6	#3	
M1864–M1879	#13			
M1880–M1895	#14	#7		
M1896–M1911	#15			

For example:

In mode 10, the M elements distributed to the sending zone assigned by 0# station (layer 1) are M1656﹣M1671. The 0# station PLC can write values to M1656﹣M1671, and other stations (1#-–15#) can read directly the value of M1656﹣M1671.

Note

Once the PLC is configured with the N:N communication protocol,D7700–D7763 and M1400–M1911 are used as the public resources for network data exchange. You need to pay attention to these elements when using them in the program!

10.5.5 Enhanced refresh mode

To support more elements sharing, the VC series micro-PLCs offer modes 14–18. These modes are only applicable to the single-layer structure, or situations where there are many shared components. M components and D components have been expanded on the original basis (M1400-M1911 and D7500–D7755).

The M component areas (512) are shown in the following table:

Distribution of M elements	Mode 14	Mode 15	Mode 16	Mode 17	Mode 18
M1400-M1415	#0	#0	#0	#0	#0
M1416-M1431	#1				
M1432-M1447	#2	#1			
M1448-M1463	#3				
M1464-M1479	#4	#2	#1		
M1480-M1495	#5				
M1496-M1511	#6	#3			
M1512-M1527	#7				
M1528-M1543	#8	#4	#2	#1	
M1544-M1559	#9				
M1560-M1575	#10	#5			
M1576-M1591	#11				
M1592-M1607	#12	#6	#3		
M1608-M1623	#13				
M1624-M1639	#14	#7			
M1640-M1655	#15				
M1656-M1671	#16	#8	#4	#2	#1
M1672-M1687	#17				
M1688-M1703	#18	#9			
M1704-M1719	#19				
M1720-M1735	#20	#10	#5		
M1736-M1751	#21				
M1752-M1767	#22	#11			
M1768-M1783	#23				

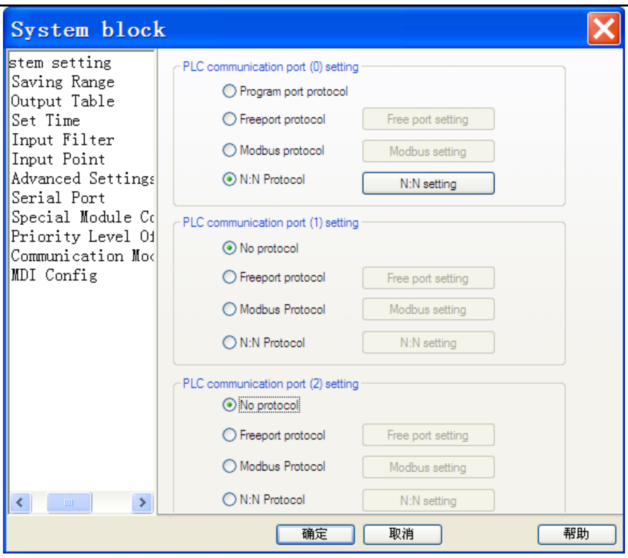
M1784-M1799	#24	#12	#6	#3	
M1800-M1815	#25				
M1816-M1831	#26	#13			
M1832-M1847	#27				
M1848-M1863	#28	#14	#7		
M1864-M1879	#29				
M1880-M1895	#30	#15			
M1896-M1911	#31				

The D element areas (256) are shown in the following table:

Distribution of D elements	Mode 14	Mode 15	Mode 16	Mode 17	Mode 18
D7500–D7507	#0	#0	#0	#0	#0
D7508–D7515	#1				
D7516–D7523	#2	#1			
D7524–D7531	#3				
D7532–D7539	#4	#2	#1		
D7540–D7547	#5				
D7548–D7555	#6	#3			
D7556–D7563	#7				
D7564–D7571	#8	#4	#2	#1	
D7572–D7579	#9				
D7580–D7587	#10	#5			
D7588–D7595	#11				
D7596–D7603	#12	#6	#3		
D7604–D7611	#13				
D7612–D7619	#14	#7			
D7620–D7627	#15				
D7628–D7635	#16	#8	#4	#2	#1
D7636–D7643	#17				
D7644–D7651	#18	#9			
D7652–D7659	#19				
D7660–D7667	#20	#10	#5		
D7668–D7675	#21				
D7676–D7683	#22	#11			
D7684–D7691	#23				
D7692–D7699	#24	#12	#6	#3	
D7700–D7707	#25				
D7708–D7715	#26	#13			
D7716–D7723	#27				
D7724–D7731	#28	#14	#7		
D7732–D7739	#29				
D7740–D7747	#30	#15			
D7748–D7755	#31				

10.5.6 N:N parameter setting

Selecting the **Communication Port** item in the System block, and selecting the **N:N protocol** in the **PLC communication port 1 setting** to enable the corresponding **N:N setting** button as shown below.

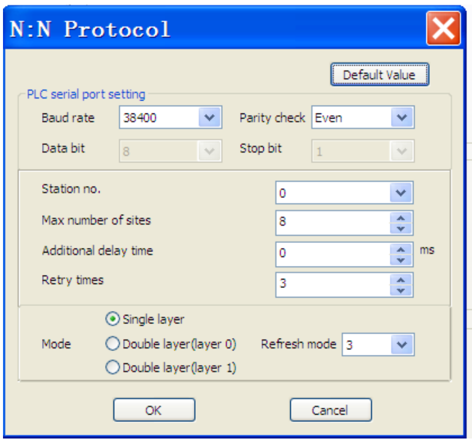

System block
stem setting
Saving Range
Output Table
Set Time
Input Filter
Input Point
Advanced Settings
Serial Port
Special Module Cc
Priority Level Of
Communication Moc
MDI Config
Free port setting
Modbus setting
N:N setting
确定
取消
帮助

PLC communication port (0) setting
○ Program port protocol
○ Freeport protocol
○ Modbus protocol
● N:N Protocol

PLC communication port (1) setting
● No protocol
○ Freeport protocol
○ Modbus Protocol
○ N:N Protocol

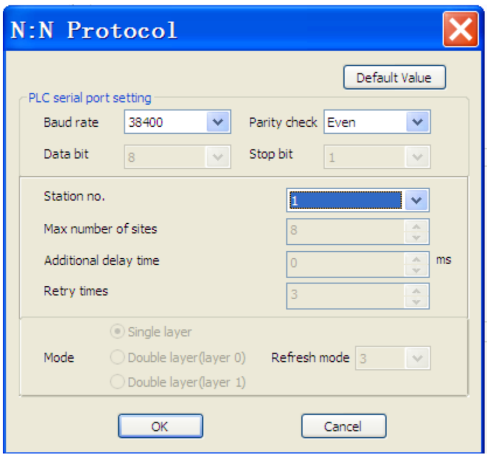
PLC communication port (2) setting
● No protocol
○ Freeport protocol
○ Modbus Protocol
○ N:N Protocol

Clicking the **N:N setting** button to enter the **N:N Protocol** setting interface, as shown below.


N:N Protocol
Default Value
OK
Cancel

PLC serial port setting
Baud rate 38400
Parity check Even
Data bit 8
Stop bit 1
Station no. 0
Max number of sites 8
Additional delay time 0 ms
Retry times 3
Mode ● Single layer ○ Double layer(layer 0) ○ Double layer(layer 1)
Refresh mode 3

As shown in the preceding figure, the N:N parameters are set through the system block. The **Station no.**shall be set starting from 0#, and set in order. Several PLCs cannot share the same station number. 0#station is used for starting and setting the whole network. The setting of **Max number of stations**, **Additional delay time**, **Retry times**, **Mode** can be realized through 0#station. For the stations with other station numbers, except that the baud rate and parity check shall be consistent with those of 0#station, they only need to set their own station number, as shown in the following figure.


N:N Protocol
Default Value
OK
Cancel

PLC serial port setting
Baud rate 38400
Parity check Even
Data bit 8
Stop bit 1
Station no. 1
Max number of sites 8
Additional delay time 0 ms
Retry times 3
Mode ● Single layer ○ Double layer(layer 0) ○ Double layer(layer 1)
Refresh mode 3

The maximum number of inspection stations refers to the total number of PLCs used in the network. If you use a total of 6 PLCs, you need to set the maximum number of inspection stations to 6, and set the station number of these 6 PLCs to 0-5. If you want to add another two PLCs to the network without any interruption of the network, you can set the maximum number of inspection stations to 8 and the PLC stations to be added in the future to 6 and 7respectively. When 6 and 7 are connected to the network, they are automatically detected by N:N within one second and included into the data exchange with 0-5.

10.6 Several control strategies

10.6.1 Determination of the master station

0# station is a master station by default. Only 0#station can initialize and start the entire network. The settings about N:N, such as refresh mode, additional delay time, and retry times, needs to be configured only through the 0# station. When 0# stationmodifies the relevant configuration online and the standby master station takes over the network during the process of downloading the system blocks. When 0#station completes the downloading of the system block, the standby master station givesthe place of the master station to 0#station.

Master station strategy in the network: The station with the mini. station number acts as the master station.

10.6.2 Maximum number of inspection stations

When setting the max. number of the inspection stations, it is recommended to set the max. number of the inspection stations to the total number of PLCs included in the actual network, and set the station number starting from 0# in turn. When the max. number of the inspection stations is set to N#, the network only manages the stations ranging from 0# to N-1#. In particular, when the max. number of the inspection stations set by the user is incorrect, that is, when the maximum number of inspection stations is smaller than the number of PLCs included in the actual 485 network, PLCs with station numbers that are greater than or equal to the max. number of the inspection stations cannot broadcast their data but can receive the broadcast data of the PLCs with station numbers that are less than the max. number of the inspection stations.

10.6.3 Multi-master and slave (M: N)

A network of multi-master and multi-slave structures can be builtby using N:N. The meaning of "master" and "slave" here is: "master" indicates a PLC that cannot only write its own M and D elements, but also read M and D elements of other stations; "slave" indicates a PLC that can only read M and D elements of other stations. Under the set maximum number of inspection stations (this number is also subject to the refresh mode), the PLC whose station number is less than the number of inspection stations can be used as the "master" while the PLC whose station number is greater than the number of inspection stations can only be used as the "slave". The slave station can only read the relevant M and D elements of the master station. These M and D elements have the corresponding relationship with each master station according to the refresh mode in the master station. You can refer to the N:N shared M and D element table. There are no corresponding M and D elements for the slave station in these tables.

10.6.4 Examples of using N:N

There are 5 PLCs in total, and the station number ranges from 0# to 4#. You can select 3 for the refresh mode 3. If you want to store the sum of D100 in 0#PLC and D305 in 2#PLC in D500 of 4#PLC, you can program as follows:

Programming 0#: **MOV D100 D7700**

Programming 2#: **MOV D305 D7716**

Programming 4#: **ADD D7700 D7716 D500**

Description: This example show the N:N single-layer network. There are 5 PLC stations on the network and the refresh mode is set to 3. Each station can be distributed with 8 D elements and 64 M elements. The D elements distributed to 0# station ranges from D7700 to D7707, the ones distributed to 2# station ranges from D7716 to D7723, and the ones distributed to 4# station ranges from D7732 to D7739. Storing the D100 value of 0# station in a write common area D7700 distributed by the network, and the D305 value of 2# station in a write common area D7716 distributed by the network. Executing the sum operation of D7700 and D7716 in 4#PLC and storing the sum in the local element D500.

Chapter 11 Positioning function guide

Chapter 11 Positioning function guide 316
11.1 Positioning control system....................317
11.1.1 Absolute position system....................317
11.1.2 Positioning control system....................318
11.1.3 Process of positioning control 319
11.2 Overview of the VC series PLC positioning functions....................319
11.3 Note of using the positioning instructions 322
11.4 Special elements related to the positioning instructions 323
11.4.1 Outputaxes of the VC series PLC 323
11.4.2 Outputaxes of the VC1 series PLC 324
11.5 Application instance....................324
11.5.1 Configuration method of the PLS envelope instruction 324

11.1 Positioning control system

11.1.1 Absolute position system

The absolute position system obtains the absolute position data of the servo motor on the stroke by detecting the current position data of the servo motor encoder and total number of running turns. According to this principle, an absolute coordinate system can be established on the mechanical stroke. The following figure is a functional block diagram of an absolute position system.

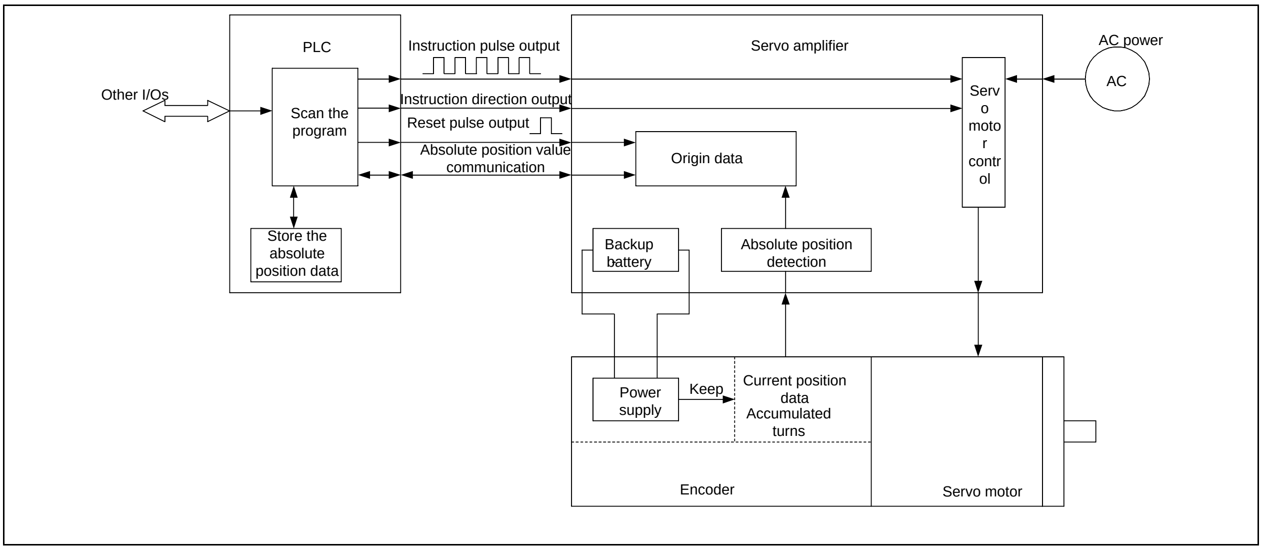

PLC
Instruction pulse output
Servo amplifier
AC power
AC
Other I/Os
Scan the program
Instruction direction output
Reset pulse output
Absolute position value communication
Origin data
Servo motor control
Store the absolute position data
Backup battery
Absolute position detection
Power supply
Keep
Current position data
Accumulated turns
Encoder
Servo motor

Figure 11-1 Absolute position system function diagram

In the above figure, unlike the conventional incremental encoder, the current position data and accumulated turns of the encoder of the absolute position system can be maintained. These data can be maintained by the power supply of a backup battery. Even in the event of a power failure, the servo drive can obtain the current absolute position data through being powered on again.

After the PLC is powered on, the absolute position data can be obtained from the servo drive through communication or other special methods, and the stroke coordinate position is determined. The PLC uses the positioning instruction to control the servo drive and the motor, thereby achieving the precise positioning on the stroke, and automatically refreshing the absolute position data automatically. In this way, a working system based on the absolute position coordinates can be formed.

The following diagram is a simplified mechanical example of an absolute position system built on the VC series PLC positioning instructions.

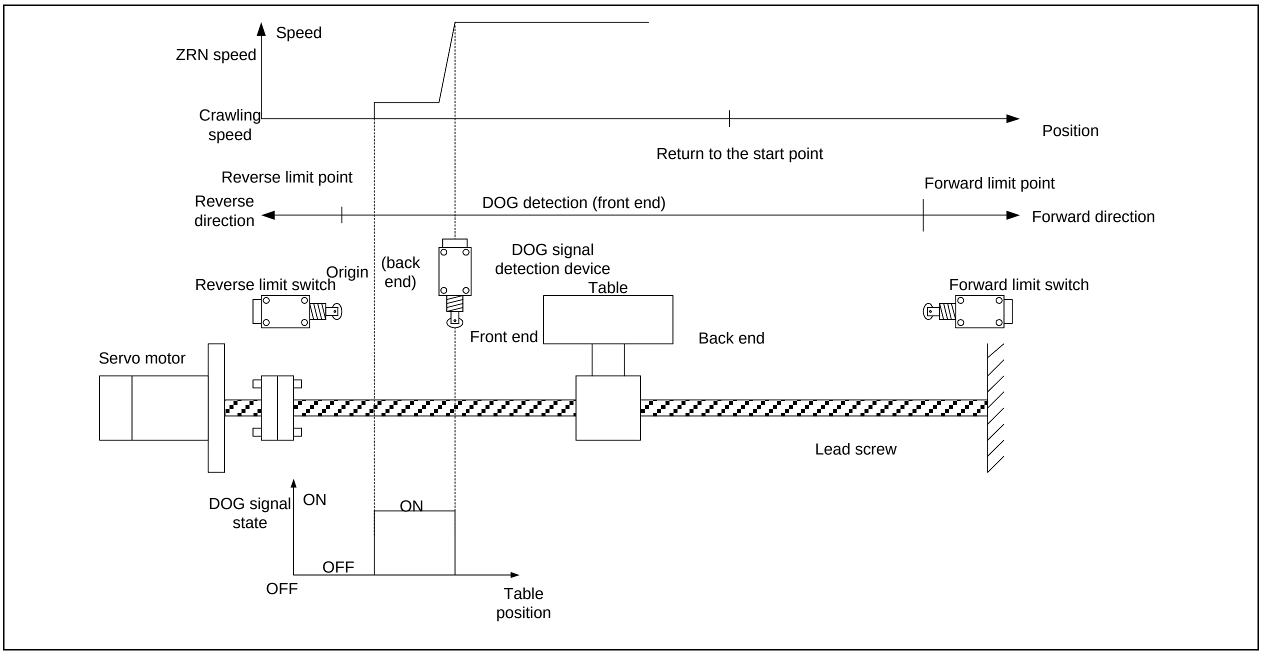

Speed
ZRN speed
Crawling speed
Position
Return to the start point
Reverse limit point
Forward limit point
Reverse direction
DOG detection (front end)
Forward direction
DOG signal detection device
Origin
(back end)
Reverse limit switch
Table
Forward limit switch
Front end
Back end
Servo motor
Lead screw
DOG signal state
ON
ON
OFF
OFF
Table position

Figure 11-2 Mechanical example diagram of an absolute position system

The system drives the lead screw with a servo motor to drive the table. The position of the table in the stroke is detected by an absolute encoder. During the zero return process,when the DOG signal device detects the front end of the table (set), the servo motor decelerates to the crawling speed; when the DOG signal device detects that the rear end of the table (reset), it is an original position arriving signal, and the PLC stops the high-speed pulse output. The forward limit switch and the reverse limit switch must be set. Because the ZRN instruction is incapable of searching for DOG signal automatically, it is required to start the ZRN operation further than the front end of DOG detection device. You can perform the inching operation to manually adjust the position of the table through the design and programming.

11.1.2 Positioning control system

The positioning control system can be divided into open-loop control system, semi-closed loop control system and closed-loop control system according to different control modes.

The open-loop control system is a control system in which the regulating system does not accept the control of the feedback, and only controls the output, also known as the no feedback control system. The open-loop control system is mostly composed of a controller, stepping driver, and stepping motor. The controller sends a pulse instruction to the stepping driver, and then the stepping motor drives the table to move a certain distance. This kind of system is relatively simple, stable in operation, and easy to grasp, but it cannot detect errors, nor can it correct errors. Its control accuracy and the performance of suppressing interference are relatively poor, and it is sensitive to the changes in system parameters. Therefore, it is generally only used in applications where the external influence is not considered, or the inertia is small, or the accuracy is not high.

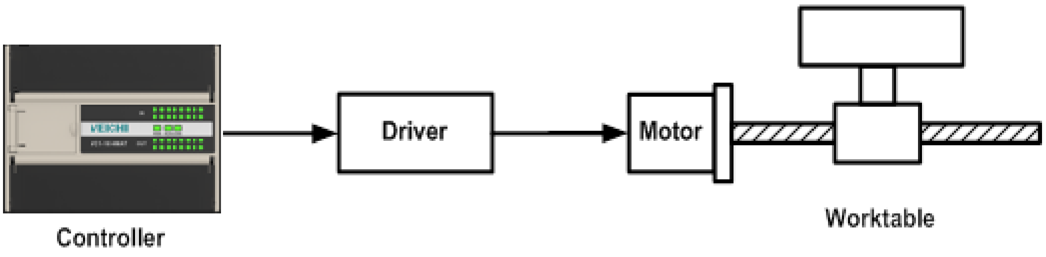


Figure 11-3 Open-loop control system diagram

The closed-loop control system is an automatic control system in which a closed loop consists of a signal forward path and feedback path, also known as a feedback control system. The closed-loop control system generally consists of a controller, servo drive, servo motor, detector, etc. The system automatically detects the actual displacement of the table and feeds back to the controller for the closed-loop control. This kind of system has the high positioning accuracy, but the system is complicated, and difficult to debug and maintain, and its price is relatively expensive. It is mainly used for the high-precision applications and large CNC machines.

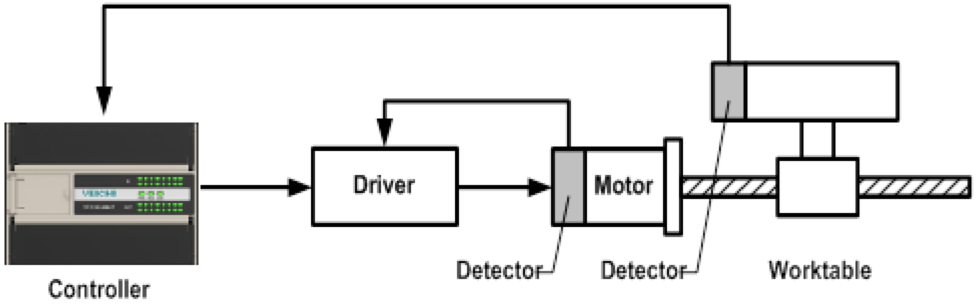


Figure 11-4 Closed-loop control systemdiagram

The semi-closed loop control system works similarly to the closed-loop control system except that the detector is not mounted on the table but mounted on the shaft of the servo motor. The accuracy, speed and dynamic characteristics of this system are superior to the open-loop control system. Its complexity and cost are lower than that of the closed-loop control system. They are mainly used for medium-precision applications and most of small and medium-sized CNC machines.

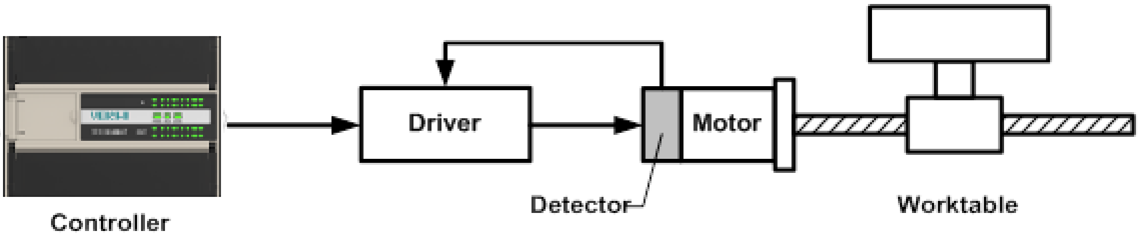

Driver
Motor
Detector
Worktable
Controller

Figure 11-5Semi-closed loop control system diagram

11.1.3 Process of positioning control

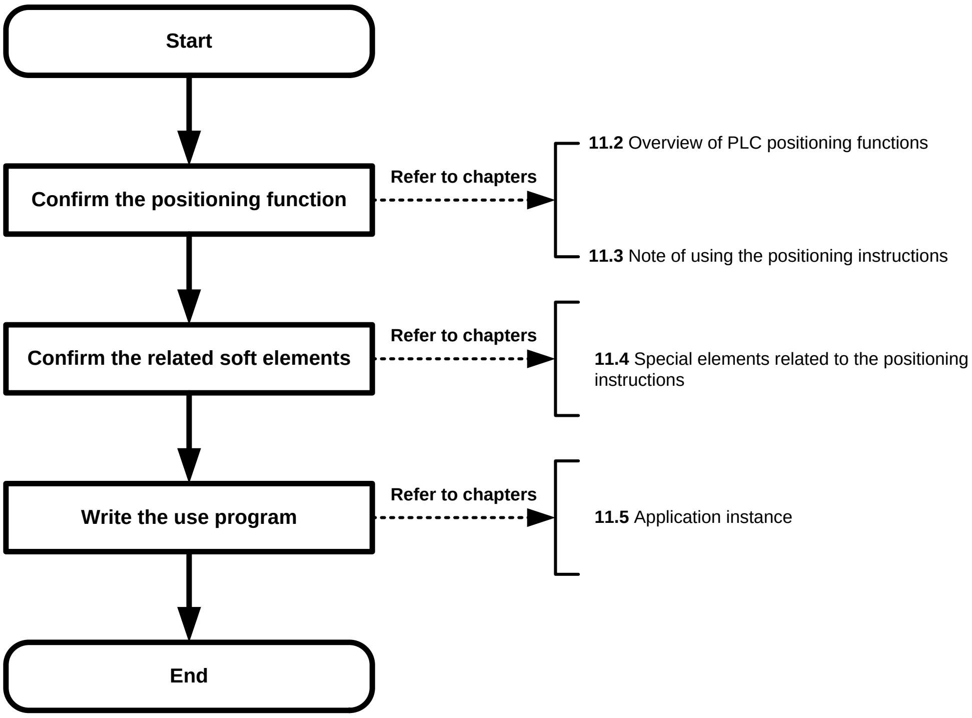

Start
Confirm the positioning function
Refer to chapters
11.2 Overview of PLC positioning functions
11.3 Note of using the positioning instructions
Confirm the related soft elements
Refer to chapters
11.4 Special elements related to the positioning instructions
Write the use program
Refer to chapters
11.5 Application instance
End

Figure 11-6 Process of positioning control

11.2 Overview of the VC series PLC positioning functions

The positioning functions supported by VC series PLCs include simple pulse output positioning, two-axis linear and arc trajectory interpolation, and inter-axis synchronous motion control, which can be widely used in positioning control systems to control the stepping and servo drives of various brands. The absolute position data can be obtained by the means provided by the corresponding servo drive.

Table 11-1 Overview of positioning functions of the VC series PLC main module

Name	VC3(Planning)	VC2(pianning)	VC1	VC1S(plannning)
Controlled axes	8-axis	3-axis	3-axis	2-axis
Max. output frequency	200 kHz	100 kHz	100 kHz	100 kHz
Pulse output mode	Open collector	Open collector	Open collector	Open collector
Pulse output type	Pulse+direction, positive and negative pulses	Pulse+direction	Pulse+direction	Pulse+direction
Acceleration and deceleration treatment	Trapezoidal acceleration/deceleration, triangle acceleration and deceleration disabling	Trapezoidal acceleration/deceleration	Trapezoidal acceleration/deceleration	Trapezoidal acceleration/deceleration

Interpolation function	2-axis linear interpolation and circular interpolation	—	—	—
Synchronization function	Position synchronization, electronic gear	—	—	—
Absolute position detection	ABS instruction read		—	—
Positioning range	-2,147,483,648–+2,147,483,647 (pulse)			

*Note: Only interpolation instruction supports the positive and negative pulse output modes.

When connecting to the servo, you need to set the input signal of the servo amplifier to the negative logic mode. The pulse output form is defined as follows:

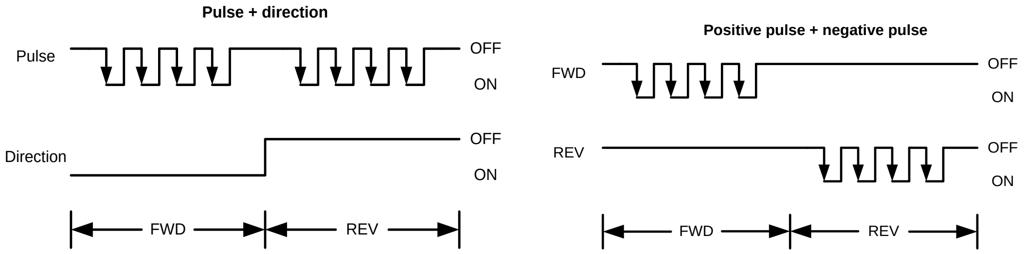
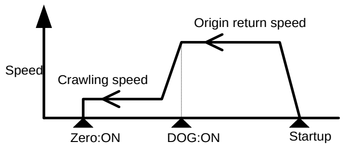
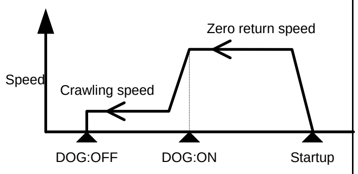
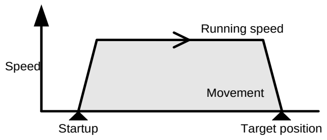
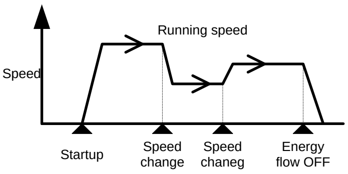
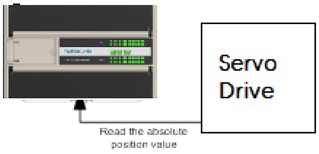
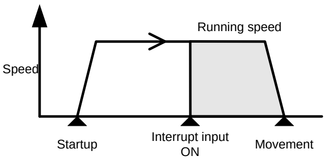
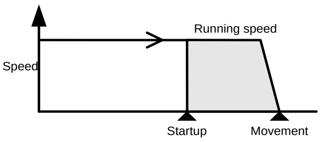
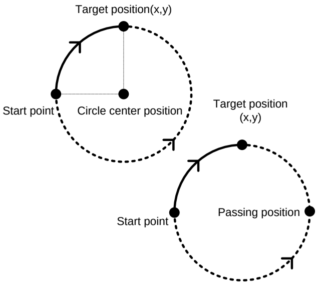
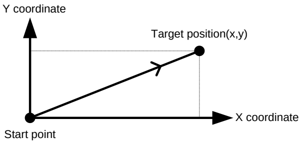

Pulse + direction
Pulse
OFF
ON
Direction
OFF
ON
FWD
REV
Positive pulse + negative pulse
FWD
OFF
ON
REV
OFF
ON
FWD
REV

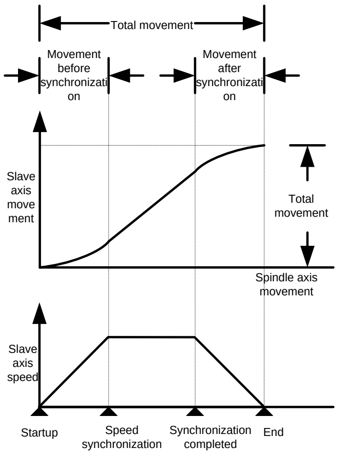
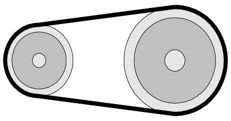
Figure 11-7 Definition of pulse output form

Note: The high-speed I/O instruction can also be used as a pulse output, but only for the output control of the pulse signal, and does not control the direction signal. When these high-speed I/O instructions are used, the corresponding position SD elements are subjected to the accumulation processing in the forward direction. When you want to drive the servo in forward motion, you needs to program and set the servo direction signal to ON, and set to OFF in reverse motion.

Table 11-2 Positioning function list of VC series PLC main modules

Name	Action	Content	VC3	VC2	VC1	VC1S
DSZR	 Origin return speed Speed Crawling speed Zero:ON DOG:ON Startup	It acts according to the specified zero return speed and can automatically search for the DOG signal. Decelerating to the crawling speed once the DOG is detected (the sensor of DOG is ON). Stopping when a zero signal is inputted, the ZRN is completed.	•	•	•	
ZRN	 Zero return speed Speed Crawling speed DOG:OFF DOG:ON Startup	It acts according to the specified zero return speed and decelerates to the crawling speed once the DOG is detected (the sensor of DOG is ON). Stopping when the sensor of DOG is OFF, and the ZRN is completed.	•	•	•	•
DRVI	 Running speed Speed Movement Startup Target position	It acts according to the set running speed, stops at the target position, and uses relative coordinates for the position.	•	•	•	•

Name	Action	Content	VC3	VC2	VC1	VC1S
DRVA		It acts according to the set running speed, stops at the target position, and uses absolute coordinates for the position.	●	●	●	●
PLSV	 Running speed Speed Startup Speed change Speed chaneg Energy flow OFF	It acts according to the set running speed. If the running speed changes, it runs at the new speed; if the energy flow becomes invalid, the pulse output stops. When there is acceleration/deceleration, the acceleration/deceleration is executed when the speed is changed.	●	●	●	●
ABS	 Servo Drive Read the absolute position value	Reading the current absolute position data from the servo drive.				
DVIT	 Running speed Speed Startup Interrupt input ON Movement	It acts according to the set running speed. If the interrupt input is ON, it runs the specified number of pulses and then decelerates to stop.			●	
STOPDV	 Running speed Speed Startup Movement	When a positioning operation is being executed, if this instruction is started, it runs the specified number of pulses and then decelerates to stop.	●	●		
CW	 Target position(x,y) Start point Circle center position Target position (x,y) Start point Passing position	It moves to the target position along the circular trajectory in the clockwise direction at the specified linear speed.	●			
CCW		It moves to the target position along the circular trajectory in the anti-clockwise direction at the specified linear speed.	●			
LIN	 Y coordinate Target position(x,y) X coordinate Start point	It moves to the target position along a linear trajectory according to the specified vector speed.	●			

Name	Action	Content	VC3	VC2	VC1	VC1S
MOVELINK	 Total movement Movement before synchronization Movement after synchronization Slave axis movement Total movement Spindle axis movement Slave axis speed Startup Speed synchronization Synchronization completed End	The slave axis moves with the spindle axis and keeps pace with it within a specified position range, supporting the acceleration/deceleration control during the transition process before and after the synchronization.				
GEARBOX		It controls the slave axis to move with the spindle axis according to a certain electronic gear ratio.				

The positioning instruction and high-speed instruction output a controllable pulse at the high-speed port according to the settings, and the output of the pulse has nothing to do with the scan cycle of the user program. For details about the usages of these instructions, refer tosection 6.10 "High-speed I/O instruction". In the program, using positioning instructions or high-speed instructions for different output ports can obtain independent high-speed pulse output at the corresponding output port.

11.3 Note of using the positioning instructions

When the positioning instruction or high-speed instruction is valid (including the output is completed), other operations on the same port are invalid. Other instructions have the correct output only when the high-speed pulse output instruction is invalid.

When there are multiple positioning instructions or high-speed instructions on the same port, the first valid instruction occupies the output port, and the latter valid one does not occupy the output port.

- Transistor output
  An VC series PLC with transistor output is a must.

- Requirements for positioning instructions in programming
  Positioning instructions can be used repeatedly in the program, but you need to pay attention to:

  1. You cannot drive and use other positioning or high-speed pulse output instructions on the same high-speed pulse output point at the same time. A high-speed pulse output point can only be driven by a positioning instruction (or high-speed instruction) at any one time.

  2. When the energy flow of a positioning instruction is disconnected, the instruction can be driven again only when the energy flow is connected after one or more PLC scan cycles.

- The main points for using the high-speedand positioning instructions at the same time
  Considering from the function implementation, it is recommended to use the positioning instruction to replace these high-speed pulse output instructions (PLSY, PLSR, and PLS), so as to complete the automatic update of the absolute position SD elements.

  The absolute position SD elements can be used to store and update the current absolute position after the positioning instruction is used. The automatic increase and decrease of the absolute position SD element value is determined based on the change value of the output pulse accumulation SD element and the running direction when the positioning instruction is called, so they are linked. Do not perform the write operation on the the output pulse accumulation SD element when using the positioning instruction. Otherwise, the data of the absolute position SD element may be disordered.

If it is necessary to use both positioning instructions and other high-speed pulse output instructions (PLSY, PLSR, and PLS) at the same time, you need to write the PLC program so that the data of absolute position SD elements in the absolute position register can be correctly updated.

- Restrictive conditions for the actual output frequency of the positioning instruction

  When the positioning instruction is executed, the min. frequency of the actual output pulse is limited by the following formula:

$$F_{\min\_acc} = \sqrt{\frac{F_{\max} \times 500}{T}}$$

In the above formula, $F_{\max}$ indicates the max.speed; T indicates the acceleration/deceleration time , and the unit is ms. The calculation result $F_{\min\_acc}$ is the min. output frequency limit value.

If the output frequency specified in the positioning instruction is F, there are three cases of the actual output frequency as follows.

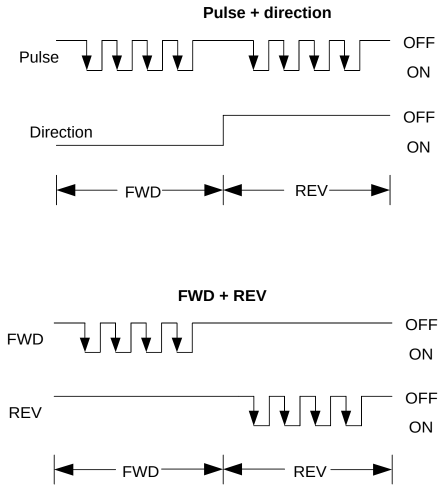
- When F is less than the base frequency or greater than the max. frequency $F_{\max}$, there is actually no output.
- When F is less than $F_{\min\_acc}$, and the actual output is $F_{\min\_acc}$.
- When F is greater than or equal to $F_{\min\_acc}$, and less than or equal to $F_{\max}$, the output is F.

11.4 Special elements related to the positioning instructions

11.4.1 Outputaxes of the VC series PLC

The definition and assignment of the output axes of the VC1 series PLCs are shown in the table below.

Table 11-3 Definition of the VC1-1616MAT output axis

Output axis	Supported mode	Definition of the output points		Output mode definition
0	Pulse + direction	Pulse	Y0	 Pulse + direction Pulse OFF ON Direction OFF ON FWD REV FWD + REV FWD OFF ON REV OFF ON FWD REV
		Direction	Any output points except Y0	
1	Pulse + direction	Pulse	Y1	
		Direction	Any output points except Y1	
2	Pulse + direction	Pulse	Y2	
		Direction	Any output points except Y2	

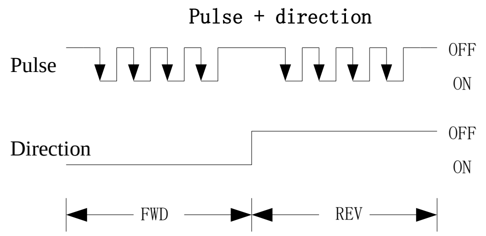
Note

When using any one of the output axes to connect the servo, you need to consider the pairing of the output points. All the output single-axis positioning can only use the "pulse + direction" mode. Only the interpolation instruction can use the "FWD + REV" mode. Only one group of interpolation instructions can be executed at one time. One group of interpolation instructions occupies 4 ports: Y0, Y1, Y2, Y3, among which Y0 and Y1 are X-axes, and Y 2 and Y3 are Y-axes.When there are double pulses, Y0 is a positive pulse and Y1 is a negative pulse while Y2 is a positive pulse and Y3 is a negative pulse. In the "pulse + direction" mode, you need to note that when the direction signal is selected, the output point cannot be used for other purposes at the same time. For example, you cannot do not define the same output point as that of the pulse or direction signal of other output axes.

11.4.2 Outputaxes of the VC1 series PLC

The definition and assignment of the output axes of the VC1 series PLCs are shown in the table below.

Table 11-5 Definition of the VC1 series output axis

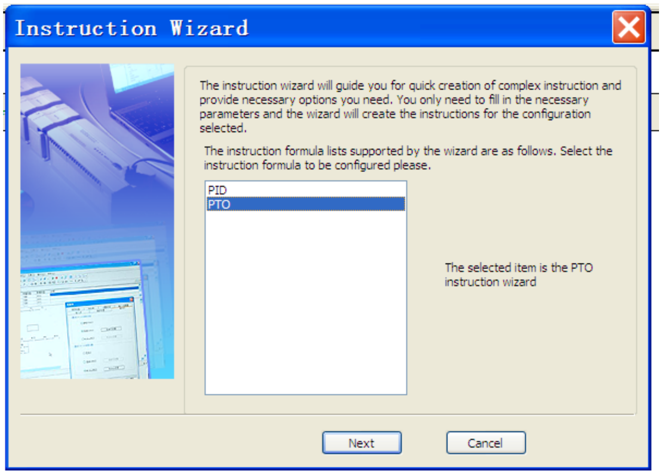
Output axis	Supported mode	Definition of the output points		Output mode definition
0	Pulse + direction	Pulse	Y0	
		Direction	Any output points except Y0	
1	Pulse + direction	Pulse	Y1	
		Direction	Any output points except Y1	
2	Pulse + direction	Pulse	Y2	
		Direction	Any output points except Y2	

11.5 Application instance

11.5.1 Configuration method of the PLS envelope instruction

- Configuration method of the PLS envelope instruction

The PLS instruction is generated by using the PTO instruction wizard. In Auto Studio, you can select **Tool** ->**Instruction Wizard**... to configure the PLS instruction. Selecting PTO as shown below.

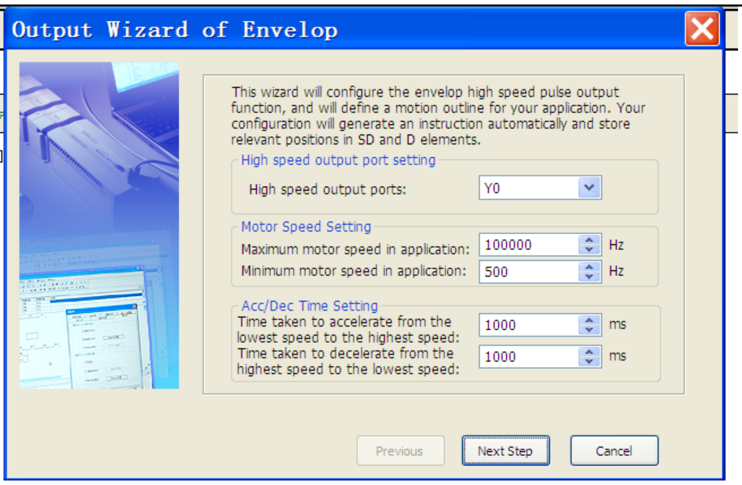


The instruction wizard will guide you for quick creation of complex instruction and provide necessary options you need. You only need to fill in the necessary parameters and the wizard will create the instructions for the configuration selected.

The instruction formula lists supported by the wizard are as follows. Select the instruction formula to be configured please.

The selected item is the PTO instruction wizard

Clicking **Next** to enter the configuration interface where high-speed pulse output points, max. and min. frequency of the high-speed pulse output, and ACC/DEC time can be set, shown in the following figure.



Output Wizard of Envelop

This wizard will configure the envelop high speed pulse output function, and will define a motion outline for your application. Your configuration will generate an instruction automatically and store relevant positions in SD and D elements.

High speed output port setting

High speed output ports: Y0

Motor Speed Setting

Maximum motor speed in application: 100000 Hz

Minimum motor speed in application: 500 Hz

Acc/Dec Time Setting

Time taken to accelerate from the lowest speed to the highest speed: 1000 ms

Time taken to decelerate from the highest speed to the lowest speed: 1000 ms

During acceleration/deceleration time, the acceleration speed of all the segments of the envelope curve is constant. For example, if the settings are as shown above, then the acceleration time taken to accelerate from 20000 Hz to 50000 Hz for the motor is:

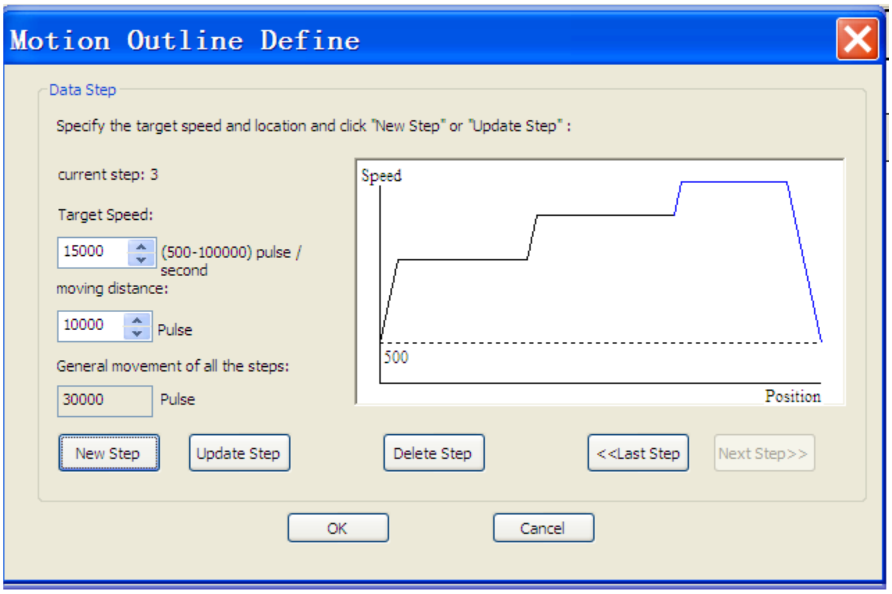
$$1000\times(50000-20000)\div(100000-5000)=316\ (\text{ms})=0.316\ (\text{s})$$

During the acceleration time, the total number of output pulses can be calculated through using the trapezoidal area calculation formula:

$$(20000+50000)\times0.316\div2=11060\ (\text{number of pulses})$$

Therefore, if you have a requirement for the time or number of pulses during acceleration/deceleration, you need to perform the related calculations before setting the max. speed, min. speed and acceleration/deceleration time.

After clicking **Next Step** in the above figure, enter **Motion Outline Define** in the figure below. You can enter the target speed and moving distance of the first step first, and click **New Step**. Then you can enter the target speed and moving distance of the second step, and click **New Step**, and so on, and finally click **OK**.



Motion Outline Define

Data Step

Specify the target speed and location and click "New Step" or "Update Step" :

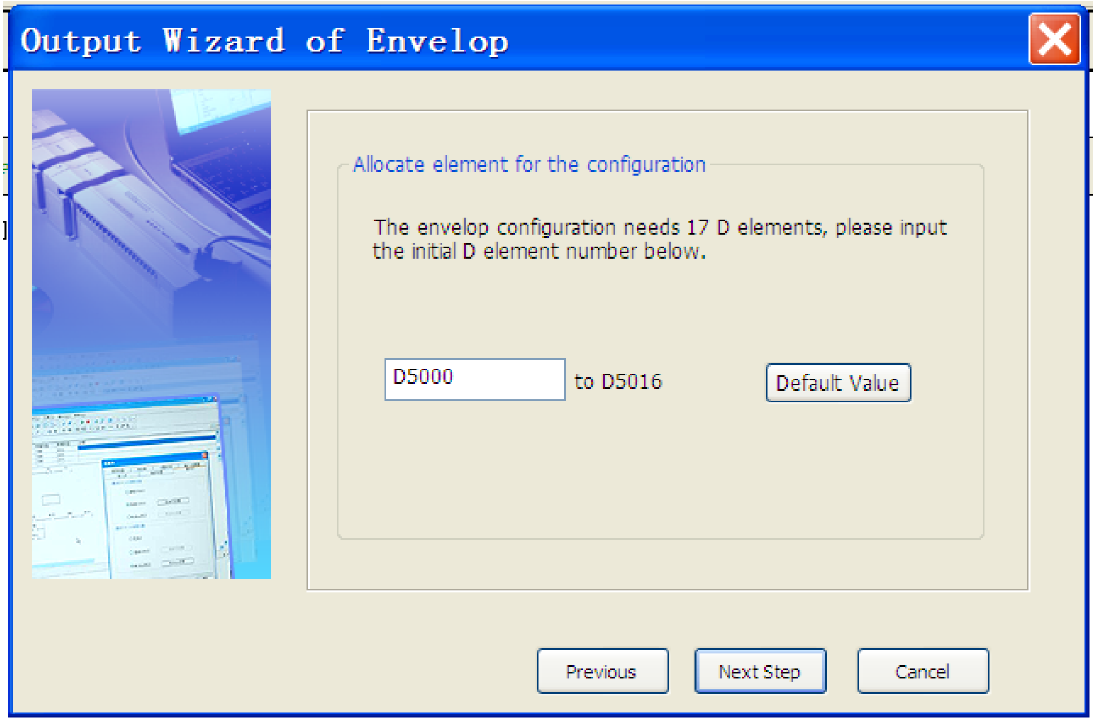
current step: 3

Target Speed: 15000 (500-100000) pulse / second

moving distance: 10000 Pulse

General movement of all the steps: 30000 Pulse

The above configuration is saved in the D elements, and you can choose which D elements to store these settings, as shown below.



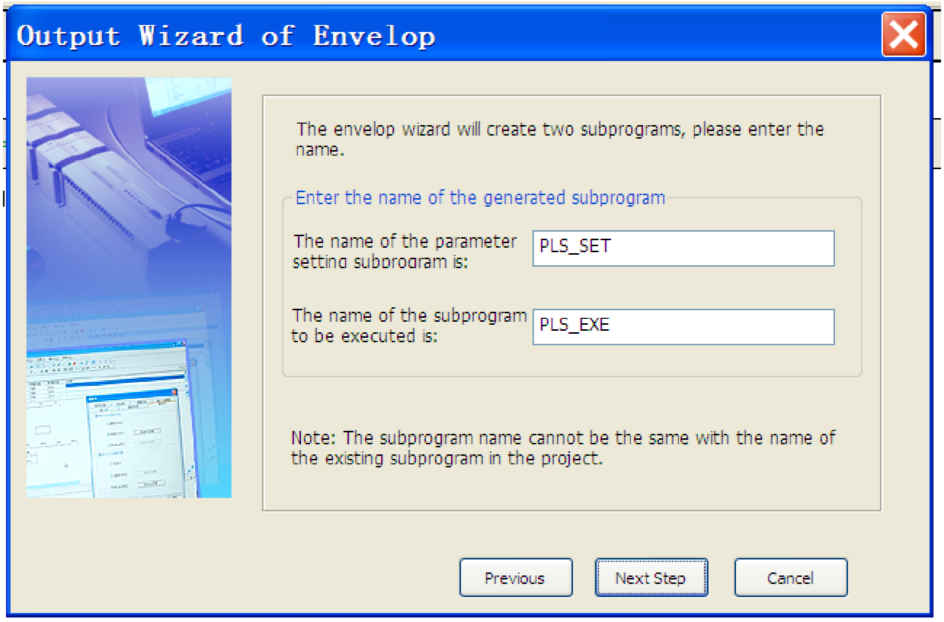
Output Wizard of Envelop

Allocate element for the configuration

The envelop configuration needs 17 D elements, please input the initial D element number below.

D5000 to D5016

The wizard will create two subprograms, one is the parameter setting subprogram, and the otheris the PLS execution subprogram, as shown below. You need to make sure that the parameter setting subprogram is correctly called and executed (the relevant D elementsare assigned) when programming in the main program, and then call the execution subprogram.



Output Wizard of Envelop

The envelop wizard will create two subprograms, please enter the name.

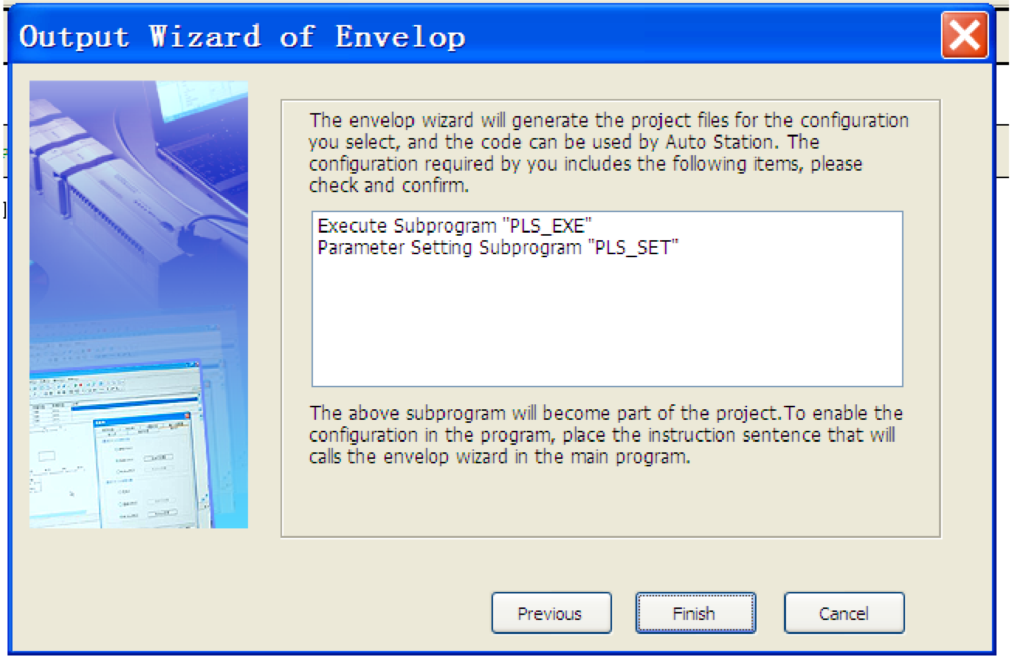
Enter the name of the generated subprogram

The name of the parameter setting subprogram is: PLS_SET

The name of the subprogram to be executed is: PLS_EXE

Note: The subprogram name cannot be the same with the name of the existing subprogram in the project.

At this point, all configuration has been completed, as shown in the figure below. Clicking **Finish** to complete the PTO configuration.



The envelop wizard will generate the project files for the configuration you select, and the code can be used by Auto Station. The configuration required by you includes the following items, please check and confirm.

Execute Subprogram "PLS_EXE"
Parameter Setting Subprogram "PLS_SET"

The above subprogram will become part of the project.To enable the configuration in the program, place the instruction sentence that will calls the envelop wizard in the main program.

Appendix A Special auxiliary relay

All special auxiliary relays are initialized when the PLCs change from STOP to RUN. The special auxiliary relays that have been set in the system setting are set to the preset value in the system block after the initialization.

📖 Note

The reserved SD and SM elements are not listed in the table. The reserved SM elements are by default read and write (R/W).

1. PLC working state flag

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM0	Monitoring run bit	This bit is always ON in RUN, and OFF in STOP.	R	√	√	√	√	√
SM1	Initial run pulse bit	This bit is set high when the user program changes from STOP to RUN, and set low after a scan cycle	R	√	√	√	√	√
SM2	Power-on flag bit	This bit is set ON after the system is powered-on, and set OFF after a scan cycle	R	√	√	√	√	√
SM3	System error	This bit is set when a system error occurs after power-on or after PLC changes from STOP to RUN, or reset if no system error occurs	R	√	√	√	√	√
SM4	Battery voltage low	This bit is set when the battery voltage is too low, or reset if the battery voltage is detected to be higher than 2.4 V.	R		√	√	√	√
SM7	No battery working mode	If this bit is set to 1 (configurable only through the system block), the system does not report errors caused by loss of battery backup data or loss of forcing tables upon backup battery failure.	R		√	√	√	√
SM8	Constant scan mode	Setting this bit, and the scan time is constant (configurable only through the system block)	R	√	√	√	√	√
SM9	Startup mode of the input point	Setting this bit, and the PLC can changes from STOP to RUN when the designated X input point is ON (configurable only through the system block)	R	√	√	√	√	√

2. Clock running bit

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM10	10 ms clock	Clock oscillation in 10 ms cycle (reversing every half cycle. The first half cycle is 0 during the operation of the user program)	R	√	√	√	√	√
SM11	100 ms clock	Clock oscillation in 100 ms cycle (reversing every half cycle. The first half cycle is 0 during the operation of the user program)	R	√	√	√	√	√
SM12	1 s clock	Clock oscillation in 1 s cycle (reversing every half cycle. The first half cycle is 0 during the operation of the user program)	R	√	√	√	√	√
SM13	1 min clock	Clock oscillation in 1 min cycle (reversing every half cycle. The first half cycle is 0 during the operation of the user program)	R	√	√	√	√	√
SM14	1 hour clock	Clock oscillation in 1 hour cycle (reversing every half cycle. The first half cycle is 0 during the operation of the user program)	R	√	√	√	√	√
SM15	Scan cycle oscillation bit	This bit is reversed once every scan cycle (The first cycle is 0 during the operation of the user program)	R	√	√	√	√	√

3. User program execution error

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM20	Instruction execution error	This bit is set upon the instruction execution error, and the corresponding error type code is written into SD20. This bit is reset after the application instruction is correctly executed.	R	√	√	√	√	√
SM21	Instruction element number subscript overflow	This bit is set upon the instruction execution error, and the corresponding error type code is written into SD20.	R	√	√	√	√	√
SM22	Instruction parameter illegal	This bit is set upon the instruction execution error, and the corresponding error type code is written into SD20. This bit is reset after the application instruction is correctly executed.	R	√	√	√	√	√

4. Interrupt control

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM25	X0 input rising edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM26	X1 input rising edge interrupt flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM27	X2 input rising edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM28	X3 input rising edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM29	X4 input rising edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM30	X5 input rising edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM31	X6 input rising edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM32	X7 input rising edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM33	X0 input falling edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM34	X1 input falling edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM35	X2 input falling edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM36	X3 input falling edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM37	X4 input falling edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM38	X5 input falling edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM39	X6 input falling edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM40	X7 input falling edge interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM41	COM 0 frame transmission interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM42	COM 0 frame receiving interrupt enable flag bit	Enable when set as 1	R/W		√	√	√	√
SM43	COM 1 frame transmission interrupt enable flag bit	Enable when set as 1	R/W		√	√	√	√

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM44	COM 1 frame receiving interrupt enable flag bit	Enable when set as 1	R/W		√	√	√	√
SM45	COM 2 frame transmission interrupt enable flag bit	Enable when set as 1	R/W		√	√	√	√
SM46	COM 2 frame receiving interrupt enable flag bit	Enable when set as 1	RW		√		√	
SM47	Timed interrupt 0 enable flag bit	Enable when set as 1	RW	√	√		√	
SM48	Timed interrupt 1 enable flag bit	Enable when set as 1	R/W	√	√	√	√	
SM49	Timed interrupt 2 enable flag bit	Enable when set as 1	R/W	√	√	√	√	
SM50	PTO (Y0) output completion interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM51	PTO (Y1) output completion interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM52	PTO (Y2) output completion interrupt enable flag bit	Enable when set as 1	R/W	√	√	√	√	√
SM53	PTO (Y3) output completion interrupt enable flag bit	Enable when set as 1	R/W			√	√	√
SM54	PTO (Y4) output completion interrupt enable flag bit	Enable when set as 1	R/W				√	
SM55	PTO (Y5) output completion interrupt enable flag bit	Enable when set as 1	R/W			√	√	
SM56	PTO (Y6) output completion interrupt enable flag bit	Enable when set as 1	R/W				√	
SM57	PTO (Y7) output completion interrupt enable flag bit	Enable when set as 1	R/W				√	
SM58	High-speed counter interrupt enable flag bit	Enable when set as 1	R/W				√	
SM59	Interpolation completion interrupt 1 enable flag bit	Enable when set as 1	R/W				√	
SM60	Interpolation completion interrupt 2 enable flag bit	Enable when set as 1	R/W				√	
SM61	Interpolation completion interrupt 3 enable flag bit	Enable when set as 1	R/W				√	
SM62	Interpolation completion interrupt 4 enable flag bit	Enable when set as 1	R/W				√	
SM63	Enable flag bit for position-based interrupt 0 in the positioning instruction	Enable when set as 1	R/W				√	
SM64	Enable flag bit for position-based interrupt 1 in the positioning instruction	Enable when set as 1	R/W				√	
SM65	Enable flag bit for position-based interrupt 2 in the positioning instruction	Enable when set as 1	R/W				√	
SM66	Enable flag bit for position-based interrupt 3 in the positioning instruction	Enable when set as 1	R/W				√	
SM67	Enable flag bit for position-based interrupt 4 in the positioning instruction	Enable when set as 1	R/W				√	
SM68	Enable flag bit for position-based interrupt 5 in the positioning instruction	Enable when set as 1	R/W					
SM69	Enable flag bit for position-based interrupt 6 in the positioning instruction	Enable when set as 1	R/W					
SM70	Enable flag bit for position-based interrupt 7 in the positioning instruction	Enable when set as 1	R/W				√	

5. Peripheral instruction

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM78	Printing mode selection	When this bit is set to 1, there are 1-16 characters. When it is set to 0, there areregularly 8 characters.	R/W			√	√	√
SM79	Printing Peripheral instruction	When this bit is set to 1, the printing is in progress	R			√	√	√

6. 0peration flag bit

Address	Name	Function	R/W	VC1S	VC1	VC2	VC3	VC5
SM80	Zero flag bit	This bit is set when the related calculated result is zero. You can clear this bit manually	R/W	√	√	√	√	√
SM81	Carry/overflow flag bit	This bit is set when the related calculated result is a carry. You can clear this bit manually	R/W	√	√	√	√	√
SM82	Borrow flag bit	This bit is set when the related calculated result is a borrow. You can clear this bit manually	R/W	√	√	√	√	√

7. DHST/DHSP Table comparison completion flag

Address	Name	Function	R/W	VC1S	VC1	VC2	VC3	VC5
SM83	Table comparison flag	Set to ON,when the table record is completed.	R/W	√	√	√	√	√

8. ASCII code conversion instruction flag

Address	Name	Function	R/W	VC1S	VC1	VC2	VC3	VC5
SM85	ASCII instruction storage mode flag	0: MSB/LSB of each word stores one ASCII code 1: LSB of each word stores one ASCII code	R/W	√	√	√	√	√

9. Pulse capture monitoring bit

Address	Name	Function	R/W	VC1S	VC1	VC2	VC3	VC5
SM90	Input X0 pulse capture monitoring bit	(1) All the elements in the table are cleared when the PLC changes from STOP to RUN; (2) The pulse capture is valid when the HCNT and SPD instructions are being executed at the same input port. For details, refer to section 6.10.9 "SPD: frequency measurement instruction" and section 6.10.1 "HCNT: high-speed counter drive instruction".	R/W	√	√			
SM91	Input X0 pulse capture monitoring bit		R/W	√	√			
SM92	Input X0 pulse capture monitoring bit		R/W	√	√			
SM93	Input X0 pulse capture monitoring bit		R/W	√	√			
SM94	Input X0 pulse capture monitoring bit		R/W	√	√			
SM95	Input X0 pulse capture monitoring bit		R/W	√	√			
SM96	Input X0 pulse capture monitoring bit		R/W	√	√			
SM97	Input X0 pulse capture monitoring bit		R/W	√	√			
Note:								

1. All the elements in the table are cleared when the PLC changes from STOP to RUN. The pulse capture is valid when the HCNT and SPD instructions are being executed at the same input port. For details, refer to section 6.10.9"SPD: frequency measurement instruction"and section 6.10.1 "HCNT: high-speed counter drive instruction".

10. Quad-frequency

Address	Name	function	R/W	VC1S	VC1	VC2	VC3	VC5
SM100	Switching between one/four times of AB phase input of X0 and X1	Clearing when from STOP－>RUN;	R/W	√	√	√	√	√
SM101	Switching between one/four times of AB phase input of X2 and X3		R/W			√	√	√
SM102	Switching between one/four times of AB phase input of X3 and X4		R/W	√	√	√	√	√
SM103	Switching between one/four times of AB phase input of X4 and X5		R/W			√	√	√

11. Free port (PORT0)

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM110	PORT0 transmission enable flag bit	This bit is set when the XMT instruction is used, and cleared after the transmission is over. You can manually clear this bit to suspend the current transmission at PORT0. The transmission can continue when the energy flow is on again	R/W	√	√	√	√	√
SM111	PORT0 receiving enable flag bit	This bit is set when the RCV instruction is used, and cleared after the transmission is over. You can manually clear this bit to suspend the current transmission at PORT0. The transmission can continue when the energy flow is on again	R/W	√	√	√	√	√
SM112	PORT0 transmission completion flag bit	This bit is set after the transmission is over	R/W	√	√	√	√	√
SM113	PORT0 receiving completion flag bit	This bit is set after the receiving is over	R/W	√	√	√	√	√
SM114	PORT0 idle flag bit	This bit is set when the port is idle	R	√	√	√	√	√

📖 Note

SM110-SM114 are flagsfor receiving, completion and idle states in all communication protocols that are supported by PORT0. For example, PORT0 of VC1 series PLCs supports Modbus protocol. No matter what protocol is used, the functions of M110-SM114 remain the same.

---

12. Free port (PORT1)

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM120	PORT1 transmission enable flag bit	This bit is set when the XMT instruction is used, and cleared after the transmission is over. You can manually clear this bit to suspend the current transmission at PORT1. The transmission can continue when the energy flow is on again	R/W	√	√	√	√	√

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM121	PORT1 receiving enable flag bit	This bit is set when the RCV instruction is used, and cleared after the transmission is over. You can manually clear this bit to suspend the current transmission at PORT1. The transmission can continue when the energy flow is on again	R/W	√	√	√	√	√
SM122	PORT1 transmission completion flag bit	This bit is set after the transmission is over	R/W	√	√	√	√	√
SM123	PORT1 receiving completion flag bit	This bit is set after the receiving is over	R/W	√	√	√	√	√
SM124	PORT1 idle flag bit	This bit is set when the port is idle	R	√	√	√	√	√
SM125	PORT1 MODBUS communication completion	This bit is set when the communication is completed	R		√	√	√	√
SM126	PORT1 MODBUS communication error	This bit is set when communication error	R		√	√	√	√

📖 Note

SM120-SM124 are flags for receiving, completion and idle states in all communication protocols that are supported by PORT1. For example, PORT1 of VC1 series PLCs supports N:N, Modbus and Free-port protocols. No matter what protocol is used, the functions of M120-SM124 remain the same.

13. Extended free port (PORT2)

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM130	PORT2 transmission enable flag bit	This bit is set when the XMT instruction is used, and cleared after the transmission is over. You can manually clear this bit to suspend the current transmission at PORT2. The transmission can continue when the energy flow is on again	R/W		√	√	√	√
SM131	PORT1 receiving enable flag bit	This bit is set when the RCV instruction is used, and cleared after the transmission is over. You can manually clear this bit to suspend the current transmission at PORT2. The transmission can continue when the energy flow is on again	R/W		√	√	√	√
SM132	PORT2 transmission completion flag bit	This bit is set after the transmission is over	R/W		√	√	√	√
SM133	PORT2 receiving completion flag bit	This bit is set after the receiving is over	R/W		√	√	√	√
SM134	PORT2 idle flag bit	This bit is set when the port is idle	R		√	√	√	√
SM135	PORT2 MODBUS communication completion	This bit is set when the communication is completed	R		√	√	√	√
SM136	PORT2 MODBUS communication error	This bit is set when communication error	R		√	√	√	√

🕮 Note

SM130-SM134 are flags for receiving, completion and idle states in all communication protocols that are supported by PORT2.

14. N:N communication

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM140	Station 0 communication error flag		R	√	√	√	√	√
SM141	Station 1 communication error flag		R	√	√	√	√	√
SM142	Station 2 communication error flag		R	√	√	√	√	√
SM143	Station 3 communication error flag		R	√	√	√	√	√
SM144	Station 4 communication error flag		R	√	√	√	√	√
SM145	Station 5 communication error flag		R	√	√	√	√	√
SM146	Station 6 communication error flag		R	√	√	√	√	√
SM147	Station 7 communication error flag		R	√	√	√	√	√
SM148	Station 8 communication error flag		R	√	√	√	√	√
SM149	Station 9 communication error flag		R	√	√	√	√	√
SM150	Station 10 communication error flag		R	√	√	√	√	√
SM151	Station 11 communication error flag		R	√	√	√	√	√
SM152	Station 12 communication error flag		R	√	√	√	√	√
SM153	Station 13 communication error flag		R	√	√	√	√	√
SM154	Station 14 communication error flag		R	√	√	√	√	√
SM155	Station 15 communication error flag		R	√	√	√	√	√
SM156	Station 16 communication error flag		R	√	√	√	√	√
SM157	Station 17 communication error flag		R	√	√	√	√	√
SM158	Station 18 communication error flag		R	√	√	√	√	√
SM159	Station 19 communication error flag		R	√	√	√	√	√
SM160	Station 20 communication error flag		R	√	√	√	√	√
SM161	Station 21 communication error flag		R	√	√	√	√	√
SM162	Station 22 communication error flag		R	√	√	√	√	√
SM163	Station 23 communication error flag		R	√	√	√	√	√
SM164	Station 24 communication error flag		R	√	√	√	√	√
SM165	Station 25 communication error flag		R	√	√	√	√	√
SM166	Station 26 communication error flag		R	√	√	√	√	√
SM167	Station 27 communication error flag		R	√	√	√	√	√
SM168	Station 28 communication error flag		R	√	√	√	√	√
SM169	Station 29 communication error flag		R	√	√	√	√	√
SM170	Station 30 communication error flag		R	√	√	√	√	√
SM171	Station 31 communication error flag		R	√	√	√	√	√

15. System bus error flag

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM190	Main module bus error flag bit	1. This bit is reset upon correct powering-on addressing 2. This bit is reset when the PLC changes from STOP to RUN 3. This bit is reset when a new program is downloaded 4. This bit incurs system stop	R	√	√	√	√	√
SM191	General module bus error flag bit	1. This bit is set and the system raises an alarm when a general module bus operation error occurs 2. This bit is reset automatically when the system error is removed	R	√	√	√	√	√
SM192	Special module bus error flag bit	1. This bit is set and the system raises an alarm when a special module bus operation error occurs 2. This bit is reset automatically when the system error is removed	R	√	√	√	√	√

16. Real-time clock error flag

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM193	R/W real-time clock error	This bit is set upon a real-time clock error This bit is automatically cleared if the system error is removed	R		√	√	√	√

17. Enable flag of integrated analog channel

Address no.	Corresponding counter address no.	Function	R/W	VC1S	VC1	VC2	VC3	VC2
SM200	C200	When SM2 _ _ is ON, its corresponding C2_ _ becomes a decrement counter When SM2 _ _ is OFF, its corresponding C2_ _becomes an increment counter	R/W	√	√	√	√	√
SM201	C201		R/W	√	√	√	√	√
SM202	C202		R/W	√	√	√	√	√
SM203	C203		R/W	√	√	√	√	√
SM204	C204		R/W	√	√	√	√	√
SM205	C205		R/W	√	√	√	√	√
SM206	C206		R/W	√	√	√	√	√
SM207	C207		R/W	√	√	√	√	√
SM208	C208		R/W	√	√	√	√	√
SM209	C209		R/W	√	√	√	√	√
SM210	C210		R/W	√	√	√	√	√
SM211	C211		R/W	√	√	√	√	√
SM212	C212		R/W	√	√	√	√	√
SM213	C213		R/W	√	√	√	√	√
SM214	C214		R/W	√	√	√	√	√
SM215	C215		R/W	√	√	√	√	√
SM216	C216		R/W	√	√	√	√	√
SM217	C217		R/W	√	√	√	√	√
SM218	C218		R/W	√	√	√	√	√
SM219	C219		R/W	√	√	√	√	√
SM220	C220		R/W	√	√	√	√	√
SM221	C221		R/W	√	√	√	√	√
SM222	C222		R/W	√	√	√	√	√
SM223	C223		R/W	√	√	√	√	√
SM224	C224		R/W	√	√	√	√	√
SM225	C225		R/W	√	√	√	√	√
SM226	C226	When SM2 _ _ is ON, its corresponding C2_ _ becomes a decrement counter When SM2 _ _ is OFF, its corresponding C2_ _becomes an increment counter	R/W	√	√	√	√	√
SM227	C227		R/W	√	√	√	√	√
SM228	C228		R/W	√	√	√	√	√
SM229	C229		R/W	√	√	√	√	√
SM230	C230		R/W	√	√	√	√	√
SM231	C231		R/W	√	√	√	√	√
SM232	C232		R/W	√	√	√	√	√
SM233	C233		R/W	√	√	√	√	√
SM234	C234		R/W	√	√	√	√	√
SM235	C235		R/W	√	√	√	√	√

18. Count direction and monitoring of high-speed counters

Differentiation	Address no.	Name	Register content	R/W	VC1S	VC1	VC2	VC3	VC5
Single-phase one point count input	SM236	C236	The ON and low OFF of SM2 _ _correspond to counting up and down of the counter respectively When C2 _ _ of single-phase bidirectional count input counter and two-phase count input counter is in	R/W	√	√	√	√	√
	SM237	C237		R/W	√	√	√	√	√
	SM238	C238		R/W	√	√	√	√	√
	SM239	C239		R/W	√	√	√	√	√
	SM240	C240		R/W	√	√	√	√	√
	SM241	C241		R/W	√	√	√	√	√

Differentiation	Address no.	Name	Register content	R/W	VC1S	VC1	VC2	VC3	VC5
	SM242	C242	down mode, its corresponding SM2 _ _ changes to ON. When it is in up mode, its corresponding SM2 _ _ changes to OFFl.	R/W	√	√	√	√	√
	SM243	C243		R/W	√	√	√	√	√
	SM244	C244		R/W	√	√	√	√	√
	SM245	C245		R/W	√	√	√	√	√
	SM246	C246		R/W	√	√	√	√	√
	SM247	C247		R/W	√	√	√	√	√
Single-phase bidirectional count input	SM248	C248	Register content The ON and OFF of SM2 _ _correspond to counting up and down of the counter respectively	R	√	√	√	√	√
	SM249	C249		R	√	√	√	√	√
	SM250	C250		R	√	√	√	√	√
	SM251	C251		R	√	√	√	√	√
	SM252	C252		R	√	√	√	√	√
	SM253	C253		R	√	√	√	√	√
	SM254	C254		R	√	√	√	√	√
	SM255	C255		R	√	√	√	√	√
Two-phase count input	SM256	C256		R	√	√	√	√	√
	SM257	C257		R	√	√	√	√	√
	SM258	C258		R	√	√	√	√	√
	SM259	C259		R	√	√	√	√	√
	SM260	C260		R	√	√	√	√	√
	SM261	C261		R	√	√	√	√	√
	SM262	C262		R	√	√	√	√	√
	SM263	C263		R	√	√	√	√	√

19. High speed output and positioning instruction

1) Y0 Correlation flag bit

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM270	Y0 pulse output stop instruction	When this bit is ON, Y0 pulse is disabled	R/W		√	√	√	
SM271	Y0 pulse output monitoring (busy/ready)	Use fo monitoring the status of Y0. Busy is ON, ready is OFF.	R/W		√	√	√	
SM272	PWM output microseconds	When this bit is ON, The PWM instruction of Y0 outputs in microseconds.	R/W		√	√		
SM273	The interrupt drive pulse output valid	When this bit is ON, the interrupt program and subroutine can call the PLSY instruction of Y0, and the call in the main program will be continuously and repeatedly driven with the energy stream.	R/W		√	√	√	
SM274	The envelope executes in a loop	When this bit is ON , The PLS instruction loop of Y0 is executed repeatedly	R/W		√	√	√	
SM275	PLSV progressive frequency conversion	When this bit is ON ,Y0 PSLV instruction frequency changes gradually, enabling acceleration and deceleration function.	R/W		√		√	
SM276	Enabling the clear function	The CLR signal output function Y0 of the ZRN instruction is valid.	R/W	√	√	√	√	
SM277	The element specified by the clear signal is valid	Y element Y (N) corresponding to the value N in SD175 is used to indicate the clear signal. If not specified, Y0 is Y10 by default, which is applicable to DSZR and ZRN.	R	√	√	√	√	
SM278	The ZRN direction	Y0 is applicable to DSZR	R		√	√	√	
SM279	FWD limit	Y0 is applicable to DSZR/DVIT	R		√	√	√	

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM280	REV limit	Y0 is applicable to DSZR/DVIT	R		√	√	√	
SM281	Logic reversal of the DOG signal	Y0 is applicable to DSZR	R		√	√	√	
SM282	Logic reversal of the zero signal	Y0 is applicable to DSZR	R		√	√	√	
SM283	Logic reversal of the interrupt signal	Y0 is applicable to DVIT, but not applicable to the user interrupt input instruction	R/W		√	√		
SM284	The positioning instruction is driving	Y0 is applicable to DSZR/DVIT	R/W		√	√	√	
SM285	User interrupt input instruction	Y0 is applicable to DVIT	R/W		√			

2) Y1 Correlation flag bit

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM290	Y1 pulse output stop instruction	When this bit is ON, Y1 pulse is disabled	R/W		√	√	√	
SM291	Y1 pulse output monitoring (busy/ready)	Use fo monitoring the status of Y1. Busy is ON, ready is OFF.	R/W		√	√	√	
SM292	PWM output microseconds	When this bit is ON, The PWM instruction of Y1 outputs in microseconds.	R/W		√	√		
SM293	The interrupt drive pulse output valid	When this bit is ON, the interrupt program and subroutine can call the PLSY instruction of Y1, and the call in the main program will be continuously and repeatedly driven with the energy stream.	R/W		√	√	√	
SM294	The envelope executes in a loop	When this bit is ON , The PLS instruction loop of Y1 is executed repeatedly	R/W		√	√	√	
SM295	PLSV progressive frequency conversion	When this bit is ON ,Y1 PSLV instruction frequency changes gradually, enabling acceleration and deceleration function.	R/W		√		√	
SM296	Enabling the clear function	The CLR signal output function Y1 of the ZRN instruction is valid.	R/W	√	√	√	√	
SM297	The element specified by the clear signal is valid	Y element Y (N) corresponding to the value N in SD195 is used to indicate the clear signal. If not specified, Y1 is Y11 by default, which is applicable to DSZR and ZRN.	R	√	√	√	√	
SM298	The ZRN direction	Y1 is applicable to DSZR	R		√	√	√	
SM299	FWD limit	Y1 is applicable to DSZR/DVIT	R		√	√	√	
SM300	REV limit	Y1 is applicable to DSZR/DVIT	R		√	√	√	
SM301	Logic reversal of the DOG signal	Y1 is applicable to DSZR	R		√	√	√	
SM302	Logic reversal of the zero signal	Y1 is applicable to DSZR	R		√	√	√	
SM303	Logic reversal of the interrupt signal	Y1 is applicable to DVIT, but not applicable to the user interrupt input instruction	R/W		√	√		
SM304	The positioning instruction is driving	Y1 is applicable to DSZR/DVIT	R/W		√	√	√	
SM305	User interrupt input instruction	Y1 is applicable to DVIT	R/W		√			

3) Y2 Correlation flag bit

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM310	Y2 pulse output stop instruction	When this bit is ON, Y2 pulse is disabled	R/W		√	√	√	
SM311	Y2 pulse output monitoring (busy/ready)	Use fo monitoring the status of Y2. Busy is ON, ready is OFF.	R/W		√	√	√	
SM312	PWM output microseconds	When this bit is ON, The PWM instruction of Y2 outputs in microseconds.	R/W		√	√		
SM313	The interrupt drive pulse output valid	When this bit is ON, the interrupt program and subroutine can call the PLSY instruction of Y2, and the call in the main program will be continuously and repeatedly driven with the energy stream.	R/W		√	√	√	
SM314	The envelope executes in a loop	When this bit is ON , The PLS instruction loop of Y2 is executed repeatedly	R/W		√	√	√	
SM315	PLSV progressive frequency conversion	When this bit is ON ,Y2 PSLV instruction frequency changes gradually, enabling acceleration and deceleration function.	R/W		√		√	
SM316	Enabling the clear function	The CLR signal output function Y2 of the ZRN instruction is valid.	R/W	√	√	√	√	
SM317	The element specified by the clear signal is valid	Y element Y (N) corresponding to the value N in SD215 is used to indicate the clear signal. If not specified, Y2 is Y12 by default, which is applicable to DSZR and ZRN.	R	√	√	√	√	
SM318	The ZRN direction	Y2 is applicable to DSZR	R		√	√	√	
SM319	FWD limit	Y2 is applicable to DSZR/DVIT	R		√	√	√	
SM320	REV limit	Y2 is applicable to DSZR/DVIT	R		√	√	√	
SM321	Logic reversal of the DOG signal	Y2 is applicable to DSZR	R		√	√	√	
SM322	Logic reversal of the zero signal	Y2 is applicable to DSZR	R		√	√	√	
SM323	Logic reversal of the interrupt signal	Y2 is applicable to DVIT, but not applicable to the user interrupt input instruction	R/W		√	√		
SM324	The positioning instruction is driving	Y2 is applicable to DSZR/DVIT	R/W		√	√	√	
SM325	User interrupt input instruction	Y2 is applicable to DVIT	R/W		√			

4) Y3 Correlation flag bit

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM330	Y3 pulse output stop instruction	When this bit is ON, Y3 pulse is disabled	R/W		√	√	√	
SM331	Y3 pulse output monitoring (busy/ready)	Use fo monitoring the status of Y3. Busy is ON, ready is OFF.	R/W		√	√	√	
SM332	PWM output microseconds	When this bit is ON, The PWM instruction of Y3 outputs in microseconds.	R/W		√	√		
SM333	The interrupt drive pulse output valid	When this bit is ON, the interrupt program and subroutine can call the PLSY instruction of Y3, and the call in the main program will be continuously	R/W		√	√	√	

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
		and repeatedly driven with the energy stream.						
SM334	The envelope executes in a loop	When this bit is ON , The PLS instruction loop of Y3 is executed repeatedly	R/W		√	√	√	
SM335	PLSV progressive frequency conversion	When this bit is ON ,Y3 PSLV instruction frequency changes gradually, enabling acceleration and deceleration function.	R/W		√		√	
SM336	Enabling the clear function	The CLR signal output function Y3 of the ZRN instruction is valid.	R/W	√	√	√	√	
SM337	The element specified by the clear signal is valid	Y element Y (N) corresponding to the value N in SD235 is used to indicate the clear signal. If not specified, Y3 is Y13 by default, which is applicable to DSZR and ZRN.	R	√	√	√	√	
SM338	The ZRN direction	Y3 is applicable to DSZR	R		√	√	√	
SM339	FWD limit	Y3 is applicable to DSZR/DVIT	R		√	√	√	
SM340	REV limit	Y3 is applicable to DSZR/DVIT	R		√	√	√	
SM341	Logic reversal of the DOG signal	Y3 is applicable to DSZR	R		√	√	√	
SM342	Logic reversal of the zero signal	Y3 is applicable to DSZR	R		√	√	√	
SM343	Logic reversal of the interrupt signal	Y3 is applicable to DVIT, but not applicable to the user interrupt input instruction	R/W		√	√		
SM344	The positioning instruction is driving	Y3 is applicable to DSZR/DVIT	R/W		√	√	√	
SM345	User interrupt input instruction	Y3 is applicable to DVIT	R/W		√			

5) Y4 Correlation flag bit

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM350	Y4 pulse output stop instruction	When this bit is ON, Y4 pulse is disabled	R/W		√	√	√	
SM351	Y4 pulse output monitoring (busy/ready)	Use fo monitoring the status of Y4. Busy is ON, ready is OFF.	R/W		√	√	√	
SM352	PWM output microseconds	When this bit is ON, The PWM instruction of Y4 outputs in microseconds.	R/W		√	√		
SM353	The interrupt drive pulse output valid	When this bit is ON, the interrupt program and subroutine can call the PLSY instruction of Y4, and the call in the main program will be continuously and repeatedly driven with the energy stream.	R/W		√	√	√	
SM354	The envelope executes in a loop	When this bit is ON , The PLS instruction loop of Y4 is executed repeatedly	R/W		√	√	√	
SM355	PLSV progressive frequency conversion	When this bit is ON ,Y4 PSLV instruction frequency changes gradually, enabling acceleration and deceleration function.	R/W		√		√	
SM356	Enabling the clear function	The CLR signal output function Y4 of the ZRN instruction is valid.	R/W	√	√	√	√	
SM357	The element specified by the clear signal is valid	Y element Y (N) corresponding to the value N in SD255 is used to indicate the clear signal. If not specified, Y4 is	R	√	√	√	√	

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
		Y14 by default, which is applicable to DSZR and ZRN.						
SM358	The ZRN direction	Y4 is applicable to DSZR	R		√	√	√	
SM359	FWD limit	Y4 is applicable to DSZR/DVIT	R		√	√	√	
SM360	REV limit	Y4 is applicable to DSZR/DVIT	R		√	√	√	
SM361	Logic reversal of the DOG signal	Y4 is applicable to DSZR	R		√	√	√	
SM362	Logic reversal of the zero signal	Y4 is applicable to DSZR	R		√	√	√	
SM363	Logic reversal of the interrupt signal	Y4 is applicable to DVIT, but not applicable to the user interrupt input instruction	R/W		√	√		
SM364	The positioning instruction is driving	Y4 is applicable to DSZR/DVIT	R/W		√	√	√	
SM365	User interrupt input instruction	Y4 is applicable to DVIT	R/W		√			

6) Y5 Correlation flag bit

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM370	Y5 pulse output stop instruction	When this bit is ON, Y5 pulse is disabled	R/W		√	√	√	
SM371	Y5 pulse output monitoring (busy/ready)	Use fo monitoring the status of Y5. Busy is ON, ready is OFF.	R/W		√	√	√	
SM372	PWM output microseconds	When this bit is ON, The PWM instruction of Y5 outputs in microseconds.	R/W		√	√		
SM373	The interrupt drive pulse output valid	When this bit is ON, the interrupt program and subroutine can call the PLSY instruction of Y5, and the call in the main program will be continuously and repeatedly driven with the energy stream.	R/W		√	√	√	
SM374	The envelope executes in a loop	When this bit is ON , The PLS instruction loop of Y5 is executed repeatedly	R/W		√	√	√	
SM375	PLSV progressive frequency conversion	When this bit is ON ,Y5 PSLV instruction frequency changes gradually, enabling acceleration and deceleration function.	R/W		√		√	
SM376	Enabling the clear function	The CLR signal output function Y5 of the ZRN instruction is valid.	R/W	√	√	√	√	
SM377	The element specified by the clear signal is valid	Y element Y (N) corresponding to the value N in SD275 is used to indicate the clear signal. If not specified, Y5 is Y15 by default, which is applicable to DSZR and ZRN.	R	√	√	√	√	
SM378	The ZRN direction	Y5 is applicable to DSZR	R		√	√	√	
SM379	FWD limit	Y5 is applicable to DSZR/DVIT	R		√	√	√	
SM380	REV limit	Y5 is applicable to DSZR/DVIT	R		√	√	√	
SM381	Logic reversal of the DOG signal	Y5 is applicable to DSZR	R		√	√	√	
SM382	Logic reversal of the zero signal	Y5 is applicable to DSZR	R		√	√	√	
SM383	Logic reversal of the interrupt signal	Y5 is applicable to DVIT, but not applicable to the user interrupt input instruction	R/W		√	√		
SM384	The positioning instruction is driving	Y5 is applicable to DSZR/DVIT	R/W		√	√	√	

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM385	User interrupt input instruction	Y5 is applicable to DVIT	R/W		√			

7) Y6 Correlation flag bit

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM390	Y6 pulse output stop instruction	When this bit is ON, Y6 pulse is disabled	R/W		√	√	√	
SM391	Y6 pulse output monitoring (busy/ready)	Use fo monitoring the status of Y6. Busy is ON, ready is OFF.	R/W		√	√	√	
SM392	PWM output microseconds	When this bit is ON, The PWM instruction of Y6 outputs in microseconds.	R/W		√	√		
SM393	The interrupt drive pulse output valid	When this bit is ON, the interrupt program and subroutine can call the PLSY instruction of Y6, and the call in the main program will be continuously and repeatedly driven with the energy stream.	R/W		√	√	√	
SM394	The envelope executes in a loop	When this bit is ON , The PLS instruction loop of Y6 is executed repeatedly	R/W		√	√	√	
SM395	PLSV progressive frequency conversion	When this bit is ON ,Y6 PSLV instruction frequency changes gradually, enabling acceleration and deceleration function.	R/W		√		√	
SM396	Enabling the clear function	The CLR signal output function Y6 of the ZRN instruction is valid.	R/W	√	√	√	√	
SM397	The element specified by the clear signal is valid	Y element Y (N) corresponding to the value N in SD295 is used to indicate the clear signal. If not specified, Y6 is Y16 by default, which is applicable to DSZR and ZRN.	R	√	√	√	√	
SM398	The ZRN direction	Y6 is applicable to DSZR	R		√	√	√	
SM399	FWD limit	Y6 is applicable to DSZR/DVIT	R		√	√	√	
SM400	REV limit	Y6 is applicable to DSZR/DVIT	R		√	√	√	
SM401	Logic reversal of the DOG signal	Y6 is applicable to DSZR	R		√	√	√	
SM402	Logic reversal of the zero signal	Y6 is applicable to DSZR	R		√	√	√	
SM403	Logic reversal of the interrupt signal	Y6 is applicable to DVIT, but not applicable to the user interrupt input instruction	R/W		√	√		
SM404	The positioning instruction is driving	Y6 is applicable to DSZR/DVIT	R/W		√	√	√	
SM405	User interrupt input instruction	Y6 is applicable to DVIT	R/W		√			

8) Y7 Correlation flag bit

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM410	Y7 pulse output stop instruction	When this bit is ON, Y7 pulse is disabled	R/W		√	√	√	
SM411	Y7 pulse output monitoring (busy/ready)	Use fo monitoring the status of Y7. Busy is ON, ready is OFF.	R/W		√	√	√	
SM412	PWM output microseconds	When this bit is ON, The PWM instruction of Y7 outputs in microseconds.	R/W		√	√		

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM413	The interrupt drive pulse output valid	When this bit is ON, the interrupt program and subroutine can call the PLSY instruction of Y7, and the call in the main program will be continuously and repeatedly driven with the energy stream.	R/W		√	√	√	
SM414	The envelope executes in a loop	When this bit is ON , The PLS instruction loop of Y7 is executed repeatedly	R/W		√	√	√	
SM415	PLSV progressive frequency conversion	When this bit is ON ,Y7 PSLV instruction frequency changes gradually, enabling acceleration and deceleration function.	R/W		√		√	
SM416	Enabling the clear function	The CLR signal output function Y7 of the ZRN instruction is valid.	R/W	√	√	√	√	
SM417	The element specified by the clear signal is valid	Y element Y (N) corresponding to the value N in SD315 is used to indicate the clear signal. If not specified, Y7 is Y17 by default, which is applicable to DSZR and ZRN.	R	√	√	√	√	
SM418	The ZRN direction	Y7 is applicable to DSZR	R		√	√	√	
SM419	FWD limit	Y7 is applicable to DSZR/DVIT	R		√	√	√	
SM420	REV limit	Y7 is applicable to DSZR/DVIT	R		√	√	√	
SM421	Logic reversal of the DOG signal	Y7 is applicable to DSZR	R		√	√	√	
SM422	Logic reversal of the zero signal	Y7 is applicable to DSZR	R		√	√	√	
SM423	Logic reversal of the interrupt signal	Y7 is applicable to DVIT, but not applicable to the user interrupt input instruction	R/W		√	√		
SM424	The positioning instruction is driving	Y7 is applicable to DSZR/DVIT	R/W		√	√	√	
SM425	User interrupt input instruction	Y7 is applicable to DVIT	R/W		√			

20. Timed output instruction

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM430	Timer clock output 1	For use by the DUTY instruction	R/W			√	√	√
SM431	Timer clock output 2	For use by the DUTY instruction	R/W			√	√	√
SM432	Timer clock output 3	For use by the DUTY instruction	R/W			√	√	√
SM433	Timer clock output 4	For use by the DUTY instruction	R/W			√	√	√
SM434	Timer clock output 5	For use by the DUTY instruction	R/W			√	√	√

21. Signal alarm

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM435	Signal alarm is valid	After setting SM400 to ON, the following SM401 and SD401 work	R/W			√	√	√
SM436	Signal alarm action	Setting SM401 to ON when there is any action in state S900-S999	R/W			√	√	√

22. CANopen instruction

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SM440	CANopen instruction completed		R/W			√	√	√
SM441	CANopen instruction is wrong		R/W			√	√	√

SM442	CANopen instruction is running		R			√	√	√

Appendix B Special data register

Note

1. The reserved SD and SM elements are not listed in the table. The reserved SD elements are by default read only.

1. PLC working state data

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD00	PLC type	100 indicates VC1S 112 indicates VC1, 40 indicates VC2, 30 indicates VC3 50 indicates VC5	R	√	√	√	√	√	
SD01	Version no.	For example, 100 indicates 1.00	R	√	√	√	√	√	
SD02	User program capacity	For example, 8 indicates an 8 k step program	R	√	√	√	√	√	
SD03	System error code	Storing the occurred system error codes	R	√	√	√	√	√	
SD04	Battery voltage value	Taking 0.1 V as unit, and 3.6V indicates 36	R		√	√	√	√	
SD07	Number of extension I/O modules		R	√	√	√	√	√	
SD08	Number of special modules		R	√	√	√	√	√	
SD09	Setting the input points for operation control. Decimal (X0 is displayed as 0 and X10 is displayed as 8. Maximum: 15) (Configurable through the system block only)		R	√	√	√	√		√
SD10	Number of main module I/O points	MSB: input; LSB: output	R	√	√	√	√	√	
SD11	Number of extension module I/O points	MSB: input; LSB: output	R	√	√	√	√	√	
SD12	Number of main module analog I/O points	MSB: input; LSB: output	R	√		√	√	√	

2. Operation error code FIFO area

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD20	Reserved operation error code 0	In the order of arrival, the latest five operation error codes are reserved. SD20 always stores the latest error codes.	R	√	√	√	√	√	
SD21	Reserved operation error code 1		R	√	√	√	√	√	
SD22	Reserved operation error code 2		R	√	√	√	√	√	
SD23	Reserved operation error code 3		R	√	√	√	√	√	

SD24	Reserved operation error code 4		R	√	√	√	√	√	

3. FROM/TO error

Address	Name	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD25	Special module's numbering is wrong .	R			√	√	√	Initial value: 255
SD26	The I/O chip's numbering is wrong (starts from 0) when I/O is refreshed	R			√	√	√	Initial value: 255

4. Scan time

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD30	Current scan value	Current scan time (unit: 0.1ms)	R	√	√	√	√	√	
SD31	Min. scan time	Min. scan time (unit: 0.1ms)	R	√	√	√	√	√	
SD32	Max. scan time	Max. scan time (unit: 0.1ms)	R	√	√	√	√	√	
SD33	Set value of constant scan time	Initial value: 0ms (configurable only through the system block), Unit: 1 ms. When the constant scan time is longer than the set value of user monitoring timeout, the user program timeout alarm is raised. When a scan cycle of the user program is longer than the constant scan time, the constant scan mode of this cycle is invalid automatically and no alarm is raised. SD33 is regarded as 1000 ms when it is set bigger than 1000 ms.	R	√	√	√	√	√	0–1000ms
SD34	Set value of user program timeout	Initial value:200 ms (configurable only through the system block). SD34 is regarded as 100 when it is set smaller than 100. SD34 is regarded as 1000 when it is set smaller than 1000.	R	√	√	√	√	√	100–1000ms

Note

1. The error tolerance of SD30, SD31 and SD32 is 1ms.

2. It is recommended to set the user program overtime (SD34) at least 5ms bigger than the constant scan time (SD33). Otherwise,when the value of SD33 closes to SD34, the system is apt to report user program overtime errordue to the influence of the system operation and user program.

5. Input filtering constant setup

Address	Name	Action and function	R/W	VC1S	VC1	VC2	VC3	VC5
SD35	Input filtering constant	(configurable only through the system block)	R/W	√	√			
SD36	Input filtering constant	(configurable only through the system block)	R/W	√	√			
SD37	Input filtering constant	(configurable only through the system block)	R/W	√	√			
SD38	Input filtering constant	(configurable only through the system block)	R/W	√	√			
SD39	Input filtering constant	(configurable only through the system block)	R/W	√	√			

SD40	Input filtering constant	(configurable only through the system block)	R/W	√	√			
SD41	Input filtering constant	(configurable only through the system block)	R/W	√	√			
SD42	Input filtering constant	(configurable only through the system block)	R/W	√	√			

6. Timed interrupt cycle

Address	Name	Register content	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD47	Cycle of timed interrupt 0	The interrupt is not triggered when the value is not within 1–32767	R/W	√	√	√	√	√	1–32767 ms
SD48	Cycle of timed interrupt 1	The interrupt is not triggered when the value is not within 1–32767	R/W	√	√	√	√	√	1–32767 ms
SD49	Cycle of timed interrupt 2	The interrupt is not triggered when the value is not within 1–32767	R/W	√	√	√	√	√	1–32767 ms

An error of ±1ms may occur when the system processes a user timed interrupt. To ensure the normal operation of the timed interrupt, it is recommended to set the cycle of timed interrupts to be bigger or equal to 5ms.

7. Real-time clock

Address	Name	Register content	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD60	Year	For real-time clock	R	√	√	√	√	√	2000–2099
SD61	Month	For real-time clock	R	√	√	√	√	√	1–12 month
SD62	Day	For real-time clock	R	√	√	√	√	√	1–31 day
SD63	Hour	For real-time clock	R	√	√	√	√	√	0–23 hour
SD64	Minute	For real-time clock	R	√	√	√	√	√	0–59 minute
SD65	Second	For real-time clock	R	√	√	√	√	√	0–59 second
SD66	Week	For real-time clock	R	√	√	√	√	√	0 (Sunday)–6 (Saturday)
You can set these elements only with the TWR instruction or through the host computer									

8. Usage of DHSP and DHST instructions

Address	Name	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD86	MSB of DHSP table comparison output data	R/W	√	√	√	√	√	
SD87	LSB of DHSP table comparison output data	R/W	√	√	√	√	√	
SD88	MSB of DHST or DHSP table comparison data	R/W	√	√	√	√	√	
SD89	LSB of DHST or DHSP table comparison data	R/W	√	√	√	√	√	
SD90	Record no. of the table being executed	R/W	√	√	√	√	√	

9. Free-port receiving control and state (PORT0)

Address	Name		Register content	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD110	Free-port 0 mode state word	SD100.0–SD100.2 Port baud rate	b2, b1, b0 000: 38,400 baud rate 001: 19,200 baud rate 010: 9,600 baud rate 011: 4,800 baud rate 100: 2,400 baud rate 101: 1,200 baud rate 110: 57,600 baud rate 111: 115,200 baud rate	R	√	√	√	√	√	
		SD100.3 Stop bit	0: 1 stop bit 1: 2 stop bits							
		SD100.4 Parity check	0: even parity; 1: odd parity							
		SD100.5 Parity check enabling	0: no parity check; 1: parity check							
		SD100.6 Character data bit	Data bit of each character 0: 8-bit character 1: 7-bit character							
		SD100.7 Free-port receiving start character mode	1: has specific start character 0: no specific start character							
		SD100.8 Free-port receiving end character mode	1: has specific end character 0: no specific end character							
		SD100.9 Free-port intercharacter timout enabling	1: has valid intercharacter timeout 0: no valid intercharacter timeout							
		SD100.10 Free-port interframe timeout enabling	1: has interframe timeout 0: no interframe timeout							
		SD100.11	Reserved							
		SD100.12 MSB/LSB valid	0: LSB of the word element is valid 1: MSB/LSB of the word element is valid							
		SD100.13 – SD100.15	Reserved							
SD101	Start character			R/W	√	√	√	√	√	√
SD102	End character			R/W	√	√	√	√	√	√

Address	Name	Register content	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD103	Intercharacter timeout	Default: 0 ms (intercharacter timeout is omitted)	R/W	√	√	√	√	√	√
SD104	Frame timeout	Default: 0 ms (frame timeout is omitted)	R/W	√	√	√	√	√	√
SD105	Receiving completion message code	Bit 0: set when the receiving ends Bit 1: set when the specified end word is received Bit 2: set when the max. character number is received Bit 3: set upon intercharacter timeout Bit 4: set upon frame timout Bit 5: set upon the parity check error Bits 6-15: reserved	R	√	√	√	√	√	√
SD106	Currently received characters		R	√	√	√	√	√	√
SD107	Total number of currently received characters		R	√	√	√	√	√	√
SD108	Currently transmitted characters		R	√	√	√	√	√	√
SD109	PORT 0 Station number setting		R/W	√	√	√	√	√	√
SD110	PORT0 maximum timeout setting (after sending and before receiving) ECBUS additional delay.		R/W	√	√	√	√	√	√
SD111	PORT 0 retry number		R/W	√	√	√	√	√	√
SD112	N:N network refresh mode (reserved by PORT 0)		R/W	√	√	√	√	√	√
SD113	Error code for Modbus master station (PORT 0)		R	√	√	√	√	√	√

10. Free-port receiving control and state (PORT1)

Address	Name		Register content	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD120	Free-port 1 mode state word	SD120.0 – SD120.2 Port baud rate	b2, b1, b0 000: 38,400 baud rate 001: 19,200 baud rate 010: 9,600 baud rate 011: 4,800 baud rate 100: 2,400 baud rate 101: 1,200 baud rate 110: 57,600 baud rate 111: 115,200 baud rate	R/W	√	√	√	√	√	

Address	Name		Register content	R/W	VC1S	VC1	VC2	VC3	VC5	Range
		SD120.3 Stop bit	0: 1 stop bit 1: 2 stop bits							
		SD120.4 Parity check	0: even parity 1: odd parity							
		SD120.5 Parity check enabling	0: no parity check 1: parity check							
		SD120.6 Character data bit	Data bit of each character 0: 8-bit character 1: 7-bit character							
		SD120.7 Free-port receiving start character mode	1: has specific start character 0: no specific start character							
		SD120.8 Free-port receiving end character mode	1: has specific end character 0: no specific end character							
		SD120.9 Free-port intercharacter timout enabling	1: has valid intercharacter timeout 0: no valid intercharacter timeout							
		SD120.10 Free-port interframe timeout enabling	1: has interframe timeout 0: no interframe timeout							
		SD120.11	Reserved							
		SD120.12 MSB/LSB valid	0: LSB of the word element is valid 1: MSB/LSB of the word element is valid							
		SD120.13–SD120.15	Reserved							
SD121	Start character			R/W	√	√	√	√	√	
SD122	End character			R/W	√	√	√	√	√	
SD123	Intercharacter timeout		Default: 0 ms (intercharacter timeout is omitted)	R/W	√	√	√	√	√	
SD124	Frame timeout		Default: 0 ms (frame timeout is omitted)	R/W	√	√	√	√	√	

Address	Name	Register content	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD125	Receiving completion message code	Bit 0: set when the receiving ends Bit 1: set when the specified end word is received Bit 2: set when the max. character number is received Bit 3: set upon intercharacter timeout Bit 4: set upon frame timout Bit 5: set upon the parity check error Bits 6-15: reserved	R	√	√	√	√	√	
SD126	Currently received characters		R	√	√	√	√	√	
SD127	Total number of currently received characters		R	√	√	√	√	√	
SD128	Currently transmitted characters		R	√	√	√	√	√	
SD129	PORT 1 Station number setting		R/W	√	√	√	√	√	√
SD130	PORT1 maximum timeout setting (after sending and before receiving) ECBUS additional delay.		R/W	√	√	√	√	√	√
SD131	PORT 1 retry number		R/W	√	√	√	√	√	√
SD132	N:N network refresh mode (reserved by PORT 1)		R/W	√	√	√	√	√	√
SD133	Error code for Modbus master station (PORT 1)		R	√	√	√	√	√	√
SD134	Error code for Modbus master station (PORT 1)	The value is 0 if there is no error in communication	R	√	√	√	√	√	√

11. Free-port receiving control and state (PORT2)

Address	Name		Register content	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD140	Free-port 2 mode state word	SD140.0 – SD140.2 Port baud rate	b2, b1, b0 000: 38,400 baud rate 001: 19,200 baud rate 010: 9,600 baud rate 011: 4,800 baud rate 100: 2,400 baud rate 101: 1,200 baud rate 110: 57,600 baud rate 111: 115,200 baud rate	R/W	√	√	√	√	√	

Address	Name		Register content	R/W	VC1S	VC1	VC2	VC3	VC5	Range
		SD140.3 Stop bit	0: 1 stop bit 1: 2 stop bits							
		SD140.4 Parity check	0: even parity 1: odd parity							
		SD140.5 Parity check enabling	0: no parity check 1: parity check							
		SD140.6 Character data bit	Data bit of each character 0: 8-bit character 1: 7-bit character							
		SD140.7 Free-port receiving start character mode	1: has specific start character 0: no specific start character							
		SD140.8 Free-port receiving end character mode	1: has specific end character 0: no specific end character							
		SD140.9 Free-port intercharacter timout enabling	1: has valid intercharacter timeout 0: no valid intercharacter timeout							
		SD140.10 Free-port interframe timeout enabling	1: has interframe timeout 0: no interframe timeout							
		SD140.11	Reserved							
		SD140.12 MSB/LSB valid	0: LSB of the word element is valid 1: MSB/LSB of the word element is valid							
		SD140.13–SD140.15	Reserved							
SD141	Start character			R/W	√	√	√	√	√	
SD142	End character			R/W	√	√	√	√	√	
SD143	Intercharacter timeout		Default: 0 ms (intercharacter timeout is omitted)	R/W	√	√	√	√	√	
SD144	Frame timeout		Default: 0 ms (frame timeout is omitted)	R/W	√	√	√	√	√	

Address	Name	Register content	R/W	VC1S	VC1	VC2	VC3	VC5	Range
SD145	Receiving completion message code	Bit 0: set when the receiving ends Bit 1: set when the specified end word is received Bit 2: set when the max. character number is received Bit 3: set upon intercharacter timeout Bit 4: set upon frame timout Bit 5: set upon the parity check error Bits 6-15: reserved	R	√	√	√	√	√	
SD146	Currently received characters		R	√	√	√	√	√	
SD147	Total number of currently received characters		R	√	√	√	√	√	
SD148	Currently transmitted characters		R	√	√	√	√	√	
SD149	PORT 2 Station number setting		R/W	√	√	√	√	√	√
SD150	PORT 2 maximum timeout setting (after sending and before receiving) ECBUS additional delay.		R/W	√	√	√	√	√	√
SD151	PORT 2 retry number		R/W	√	√	√	√	√	√
SD152	N:N network refresh mode (reserved by PORT 2)		R/W	√	√	√	√	√	√
SD153	Error code for Modbus master station (PORT 2)		R	√	√	√	√	√	√
SD154	Error code for Modbus master station (PORT 2)	The value is 0 if there is no error in communication	R	√	√	√	√	√	√

12. High-speed pulse output and positioning

1) Y0 Correlation register

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD160 SD161	The cumulative number of Y0 pulse output.	0	R/W	√	√	√	√	
SD162 SD163	Current position of Y0 output positioning instruction.	0	R/W	√	√	√	√	
SD164		0	R	√	√	√	√	√

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD165	Current frequency of Y0 output positioning instruction.							
SD166	Max. speed at which the ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (10- 100000) (Y0)	100000	R/W	√	√	√	√	
SD167								
SD168	Base speed when ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (1/10 of max. speed) (Y0)	5000	R/W	√	√	√	√	
SD169	Acceleration/deceleration time for accelerating from base speed (SD204) to max. speed (SD202, SD203) when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed (Y0) (50ms－5000ms)	1000	R/W	√	√	√	√	
SD170	Deceleration time for decelerating from max. speed to base speed when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed Y0 (50 ms－5000 ms)	1000	R/W		√			
SD171	Crawling speed Y0 is applicable to DSZR	1000	R/W	√	√	√	√	
SD172	The zero return speed Y0 is applicable to DSZR	50000	R/W	√	√	√	√	
SD173								
SD174	Number of segments to which the PLS output instruction is currently executed (Y0 applicable)	0	R/W	√	√	√	√	
SD175	Y0 is specified as the soft element of the clear signal	0	R/W	√	√	√	√	
SD176	DVIT interrupt signal soft element specification (applicable Y0)	0	R/W	√	√	√	√	

2) Y1 Correlation register

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD180	The cumulative number of Y1 pulse output.	0	R/W	√	√	√	√	
SD181								
SD182	Current position of Y1 output positioning instruction.	0	R/W	√	√	√	√	
SD183								
SD184	Current frequency of Y1 output positioning instruction.	0	R	√	√	√	√	√
SD185								
SD186	Max. speed at which the ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (10- 100000) (Y1)	100000	R/W	√	√	√	√	
SD187								
SD188	Base speed when ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (1/10 of max. speed) (Y1)	5000	R/W	√	√	√	√	
SD189	Acceleration/deceleration time for accelerating from base speed (SD204) to max. speed (SD202, SD203) when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed (Y1) (50ms－5000ms)	1000	R/W	√	√	√	√	

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD190	Deceleration time for decelerating from max. speed to base speed when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed Y1 (50 ms－5000 ms)	1000	R/W		√			
SD191	Crawling speed Y1 is applicable to DSZR	1000	R/W	√	√	√	√	
SD192	The zero return speed Y1 is applicable to DSZR	50000	R/W	√	√	√	√	
SD193								
SD194	Number of segments to which the PLS output instruction is currently executed (Y1 applicable)	0	R/W	√	√	√	√	
SD195	Y1 is specified as the soft element of the clear signal	0	R/W	√	√	√	√	
SD196	DVIT interrupt signal soft element specification (applicable Y1)	0	R/W	√	√	√	√	

3) Y2 Correlation register

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD200	The cumulative number of Y2 pulse output.	0	R/W	√	√	√	√	
SD201								
SD202	Current position of Y2 output positioning instruction.	0	R/W	√	√	√	√	
SD203								
SD204	Current frequency of Y2 output positioning instruction.	0	R	√	√	√	√	√
SD205								
SD206	Max. speed at which the ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (10- 100000) (Y2)	100000	R/W	√	√	√	√	
SD207								
SD208	Base speed when ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (1/10 of max. speed) (Y2)	5000	R/W	√	√	√	√	
SD209	Acceleration/deceleration time for accelerating from base speed (SD204) to max. speed (SD202, SD203) when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed (Y2) (50ms－5000ms)	1000	R/W	√	√	√	√	
SD210	Deceleration time for decelerating from max. speed to base speed when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed Y2 (50 ms－5000 ms)	1000	R/W		√			
SD211	Crawling speed Y2 is applicable to DSZR	1000	R/W	√	√	√	√	
SD212	The zero return speed Y2 is applicable to DSZR	50000	R/W	√	√	√	√	
SD213								
SD214	Number of segments to which the PLS output instruction is currently executed (Y2 applicable)	0	R/W	√	√	√	√	
SD215	Y2 is specified as the soft element of the clear signal	0	R/W	√	√	√	√	

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD216	DVIT interrupt signal soft element specification (applicable Y2)	0	R/W	√	√	√	√	

4) Y3 Correlation register

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD220	The cumulative number of Y3 pulse output.	0	R/W	√	√	√	√	
SD221								
SD222	Current position of Y3 output positioning instruction.	0	R/W	√	√	√	√	
SD223								
SD224	Current frequency of Y3 output positioning instruction.	0	R	√	√	√	√	√
SD225								
SD226	Max. speed at which the ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (10- 100000) (Y3)	100000	R/W	√	√	√	√	
SD227								
SD228	Base speed when ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (1/10 of max. speed) (Y3)	5000	R/W	√	√	√	√	
SD229	Acceleration/deceleration time for accelerating from base speed (SD204) to max. speed (SD202, SD203) when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed (Y3) (50ms－5000ms)	1000	R/W	√	√	√	√	
SD230	Deceleration time for decelerating from max. speed to base speed when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed Y3 (50 ms－5000 ms)	1000	R/W		√			
SD231	Crawling speed Y3 is applicable to DSZR	1000	R/W	√	√	√	√	
SD232	The zero return speed Y3 is applicable to DSZR	50000	R/W	√	√	√	√	
SD233								
SD234	Number of segments to which the PLS output instruction is currently executed (Y3 applicable)	0	R/W	√	√	√	√	
SD235	Y3 is specified as the soft element of the clear signal	0	R/W	√	√	√	√	
SD236	DVIT interrupt signal soft element specification (applicable Y3)	0	R/W	√	√	√	√	

5) Y4 Correlation register

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD240	The cumulative number of Y4 pulse output.	0	R/W	√	√	√	√	
SD241								
SD242	Current position of Y4 output positioning instruction.	0	R/W	√	√	√	√	
SD243								
SD244	Current frequency of Y4 output positioning instruction.	0	R	√	√	√	√	√
SD245								

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD246 SD247	Max. speed at which the ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (10- 100000) (Y4)	100000	R/W	√	√	√	√	
SD248	Base speed when ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (1/10 of max. speed) (Y4)	5000	R/W	√	√	√	√	
SD249	Acceleration/deceleration time for accelerating from base speed (SD204) to max. speed (SD202, SD203) when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed (Y4) (50ms－5000ms)	1000	R/W	√	√	√	√	
SD250	Deceleration time for decelerating from max. speed to base speed when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed Y4 (50 ms－5000 ms)	1000	R/W		√			
SD251	Crawling speed Y4 is applicable to DSZR	1000	R/W	√	√	√	√	
SD252 SD253	The zero return speed Y4 is applicable to DSZR	50000	R/W	√	√	√	√	
SD254	Number of segments to which the PLS output instruction is currently executed (Y4 applicable)	0	R/W	√	√	√	√	
SD255	Y4 is specified as the soft element of the clear signal	0	R/W	√	√	√	√	
SD256	DVIT interrupt signal soft element specification (applicable Y4)	0	R/W	√	√	√	√	

6) Y5 Correlation register

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD260 SD261	The cumulative number of Y5 pulse output.	0	R/W	√	√	√	√	
SD262 SD263	Current position of Y5 output positioning instruction.	0	R/W	√	√	√	√	
SD264 SD265	Current frequency of Y5 output positioning instruction.	0	R	√	√	√	√	√
SD266 SD267	Max. speed at which the ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (10- 100000) (Y5)	100000	R/W	√	√	√	√	
SD268	Base speed when ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (1/10 of max. speed) (Y5)	5000	R/W	√	√	√	√	
SD269	Acceleration/deceleration time for accelerating from base speed (SD204) to max. speed (SD202, SD203) when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed (Y5) (50ms－5000ms)	1000	R/W	√	√	√	√	

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD270	Deceleration time for decelerating from max. speed to base speed when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed Y5 (50 ms－5000 ms)	1000	R/W		√			
SD271	Crawling speed Y5 is applicable to DSZR	1000	R/W	√	√	√	√	
SD272	The zero return speed Y5 is applicable to DSZR	50000	R/W	√	√	√	√	
SD273								
SD274	Number of segments to which the PLS output instruction is currently executed (Y5 applicable)	0	R/W	√	√	√	√	
SD275	Y5 is specified as the soft element of the clear signal	0	R/W	√	√	√	√	
SD276	DVIT interrupt signal soft element specification (applicable Y5)	0	R/W	√	√	√	√	

7) Y6 Correlation register

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD280	The cumulative number of Y6 pulse output.	0	R/W	√	√	√	√	
SD281								
SD282	Current position of Y6 output positioning instruction.	0	R/W	√	√	√	√	
SD283								
SD284	Current frequency of Y6 output positioning instruction.	0	R	√	√	√	√	√
SD285								
SD286	Max. speed at which the ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (10- 100000) (Y6)	100000	R/W	√	√	√	√	
SD287								
SD288	Base speed when ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (1/10 of max. speed) (Y6)	5000	R/W	√	√	√	√	
SD289	Acceleration/deceleration time for accelerating from base speed (SD204) to max. speed (SD202, SD203) when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed (Y6) (50ms－5000ms)	1000	R/W	√	√	√	√	
SD290	Deceleration time for decelerating from max. speed to base speed when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed Y6 (50 ms－5000 ms)	1000	R/W		√			
SD291	Crawling speed Y6 is applicable to DSZR	1000	R/W	√	√	√	√	
SD292	The zero return speed Y6 is applicable to DSZR	50000	R/W	√	√	√	√	
SD293								
SD294	Number of segments to which the PLS output instruction is currently executed (Y6 applicable)	0	R/W	√	√	√	√	
SD295	Y6 is specified as the soft element of the clear signal	0	R/W	√	√	√	√	

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD296	DVIT interrupt signal soft element specification (applicable Y6)	0	R/W	√	√	√	√	

8) Y7 Correlation register

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD300	The cumulative number of Y7 pulse output.	0	R/W	√	√	√	√	
SD301								
SD302	Current position of Y7 output positioning instruction.	0	R/W	√	√	√	√	
SD303								
SD304	Current frequency of Y7 output positioning instruction.	0	R	√	√	√	√	√
SD305								
SD306	Max. speed at which the ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (10- 100000) (Y7)	100000	R/W	√	√	√	√	
SD307								
SD308	Base speed when ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions are executed (1/10 of max. speed) (Y7)	5000	R/W	√	√	√	√	
SD309	Acceleration/deceleration time for accelerating from base speed (SD204) to max. speed (SD202, SD203) when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed (Y7) (50ms－5000ms)	1000	R/W	√	√	√	√	
SD310	Deceleration time for decelerating from max. speed to base speed when the ZRN, DRVI, DRVA, DSZR and DVIT instructions are executed Y7 (50 ms－5000 ms)	1000	R/W		√			
SD311	Crawling speed Y7 is applicable to DSZR	1000	R/W	√	√	√	√	
SD312	The zero return speed Y7 is applicable to DSZR	50000	R/W	√	√	√	√	
SD313								
SD314	Number of segments to which the PLS output instruction is currently executed (Y7 applicable)	0	R/W	√	√	√	√	
SD315	Y7 is specified as the soft element of the clear signal	0	R/W	√	√	√	√	
SD316	DVIT interrupt signal soft element specification (applicable Y7)	0	R/W	√	√	√	√	

13. Timed output instruction

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD330	Scan times used by the timed clock output 1		R/W			√	√	√
SD331	Scan times used by the timed clock output 2		R/W			√	√	√
SD332	Scan times used by the timed clock output 3		R/W			√	√	√
SD333	Scan times used by the timed clock output 4		R/W			√	√	√
SD334	Scan times used by the timed clock output 5		R/W			√	√	√

14. Signal alarm instruction

Address	Action and function	Initial value	R/W	VC1S	VC1	VC2	VC3	VC5
SD339	Keeping the min. No. of actions in S900-S999 ignal alarm instruction	0	R/W				√	√

Appendix C Reserved elements

Description	VC1S/VC1/VC2/VC3/VC5	
N:N network sharing area	D7700	D7763
N:N enhanced mode (mode 14-18) network sharing area	D7500	D7755
N:N network sharing area	M1400	M1911

Appendix D Modbus communication error codes

Abnormal code	Meaning of abnormal code
0x01	Illegal function code
0x02	Illegal register address
0x03	Data number error
0x10	Communication timeout, namely communication time exceeds the max communication time set by the user
0x11	Receiving data frame error
0x12	Parameter error, setting parameter (mode or master/slave) error
0x13	The station no. is the same as that set by the instruction, and an error occurred
0x14	Element address overflows (the data quantity received or sent is beyond the element storage space)
0x15	Instruction execution failed
0x16	The received slave address does not match the requested one, and detailed error code element stores the received slave address
0x17	The received function code does not match the requested one, and detailed error code element stores the received function code
0x18	Information frame error: the start address of element does not match, and detailed error code element stores the received element start address
0x19	The received data length does not match the one specified by the protocol or the element number exceeds the max. number specified by this function code.
0x20	CRC/LRC parity error
0x21	Reserved
0x22	Start address of instruction parameter element is set wrong
0x23	Instruction parameter sets unsupported function code or illegal function code
0x24	The number of instruction parameter elements are set wrong
0x25	Reserved
0x26	Parameter is non-modifiable during running.
0x27	Parameter is protected by a password

Appendix E System error codes

Error code	Meaning	Error type	Description
0	No error occurred		
1–9	Reserved		
10	SRAM error	System error	User program stops ERR indicator turns on. To remove this error, it is required to power off the PLC and check the hardware.
11	FLASH error	System error	User program stops ERR indicator turns on. To remove this error, it is required to power off the PLC and check the hardware.
12	Communication port error	System error	User program stops ERR indicator turns on. To remove this error, it is required to power off the PLC and check the hardware.
13	Real-time clock error	System error	User program stops ERR indicator turns on. To remove this error, it is required to power off the PLC and check the hardware.
14	I2C error	System error	User program stops ERR indicator turns on. To remove this error, it is required to power off the PLC and check the hardware.
20	Serious local I/O error	System error	User program stops ERR indicator turns on. To remove this error, it is required to power off the PLC and check the hardware.
21	Serious extension I/O error	System error	ERR indicator blinks The alarm is cleared automatically upon the removal of the error
22	Serious special module error	System error	ERR indicator blinks The alarm is cleared automatically upon the removal of the error
23	Refreshing error of real-time clock (incorrect time is read during the system refreshing)	System error	ERR indicator blinks The alarm is cleared automatically upon the removal of the error
24	EEPROM read/read operation error	System error	ERR indicator blinks The alarm is cleared automatically upon the removal of the error
25	Local analog signal error	System error	ERR indicator blinks The alarm is cleared automatically upon the removal of the error
26	System special module configuration error	System error	ERR indicator blinks The alarm is cleared automatically upon the removal of the error
40	User program file error	System error	User program stops, and ERR indicator turns on To remove this error, it is required to download new program or format the disk
41	System configuration file error	System error	User program stops ERR indicator turns on To remove this error, it is required to download new system configuration files or format the disk

Error code	Meaning	Error type	Description
42	Data block file error	System error	User program stops, and ERR indicator turns on To remove this error, it is required to download new data block file or format the disk
43	Battery-backed data loss error	System error	User program keeps running, and ERR indicator blinks To remove this error, it is required to clear the elements, format the disk, or reset.
44	Forcing table loss error	System error	User program keeps running, and ERR indicator blinks To remove this error, it is required to clear the elements, force, format the disk, or reset.
45	User information file error	System error	User program stops, and ERR indicator turns on To remove this error, it is required to download new program and data block files or format the disk
46–59	Reserved		
60	User program compilation error	Execution error	User program stops, and ERR indicator turns on.
61	User program operation timeout	Execution error	User program stops, and ERR indicator turns on.
62	Illegal user program instruction execution error	Execution error	User program stops, and ERR indicator turns on.
63	Illegal element type of instruction operand	Execution error	User program stops, and ERR indicator turns on.
64	Illegal instruction operand value	Execution error	User program keeps running, and ERR indicator keeps off. The corresponding error code is prompted in SD20
65	Outside the instruction operand element range	Execution error	
66	Subprogram stack overflow	Execution error	
67	User interrupt request queue overflow	Execution error	
68	Illegal label jump or subprogram call	Execution error	
69	Divide by 0 error	Execution error	
70	Definition error of the stack operated	Execution error	When the stack size, or number of stack elements are smaller than zero, or number of the stack elements exceed the limit of the stack size
71	Reserved	Execution error	
72	Undefined user subprogram or interrupt subprogram	Execution error	
73	Invalid special module address	Execution error	
74	Error occurred when assess the special module	Execution error	
75	I/O immediate refreshing error	Execution error	
76	Clock setup error	Execution error	
77	PLSR instruction parameter error	Execution error	
78	BFM buffer of assessed special module exceeds the range	Execution error	

Appendix F Modbus communication protocols

1. Overview of Modbus communication protocol

VC series micro-PLCS are configured with two or three communication ports: PORT0(also as a programming port) supports the Modbus slave station, PORT1 and PORT2 supports Modbus master and slave stations (configurable through the background software).

1. Using RS485 or RS232 interface, with RS232 3-line system as the physical interface.
2. Supporting RTU modes
3. Being as an Modbus slave station with the address ranging from 1 to 31.
4. Supportingthe broadcast mode. The broadcast is effective for write and sub-function codes of diagnosis.
5. Supporting baud rates including 38,400 bps, 19,200 bps, 9,600 bps, 4,800 bps, 2,400 bps and 1,200 bps.(Default: 19200, 8 bits, 1 stop bit, and even check)
6. Supporting data field 2 × 252 bytes (ASII mode) or 252 bytes (RTU mode).

2. Supported Modbus function code and element addressing mode

The slave station supports function codes 01, 02, 03, 05, 06, 08, 15, and 16 (decimal).

- Relationship between read/write element function code and the element

Function code	Name of function code	Modicon data addresss	Type of operable element	Note
01	Read coil	0[Note 1]:xxxx	Y, X, M, SM, S, T, and C	Bit read
02	Read discrete input	1[Note 2]:xxxx	X	Bit read
03	Read register	4[Note 3]:xxxx[Note 4]	D, SD, Z, T, and C	Word read
05	Write single coil	0:xxxx	Y, M, SM, S, T, and C	Bit write
06	Write single register	4:xxxx	D, SD, Z, T, and C	Word write
15	Write multiple coils	0:xxxx	Y, M, SM, S, T, and C	Bit write
16	Write multiple registers	4:xxxx	D, SD, Z, T, and C	Word write

Notes:
1. 0 indicates "coil".
2. 1 indicates "discrete input".
3. 4 indicates "register".
4. xxxx indicates range "1–9999". Each type has an independent logic address range of 1 to 9999 (protocol address starts from 0).
5. 0, 1 and 4 do not have the physical meaning and are not involved in actual addressing.
6. You shall not write data in X element with function codes 05 and 15; otherwise, the system does not feed back the error information if the written operand and data are correct, but the system does not perform any operation on the write instruction.

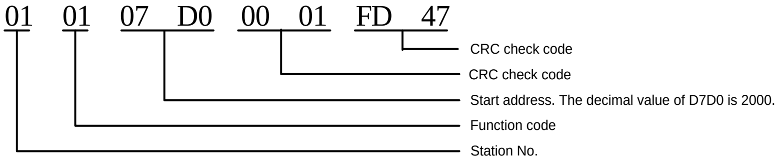
- Relationship between PLC element and Modbus communication protocol address

Element	Type	Physical element	Protocol address	Supported function code	Note
Y	Bit	Y0–Y777 (octal code) 512 points in total	0000–0511	01, 05, and 15	Output state, element code: Y0–Y7 and Y10–Y17
X	Bit	X0–X777 (octal code) 512 points in total	1200–01711	01, 05, and 15 02	Input state. It supports two kinds of addresses, and the element code is same as the above
M	Bit	M0–M2047	2000–4047	01, 05, and 15	

Element	Type	Physical element	Protocol address	Supported function code	Note
		M2048–M10240	12000-20191		
SM	Bit	SM0–SM255 SM256–SM1023	4400–4655 30000-30767	01, 05, and 15	
S	Bit	S0–S1023 S1024–S4095	6000-7023 31000-34071	01, 05, and 15	
T	Bit	T0–T255 T256–T511	8000–8255 11000-11255	01, 05, and 15	State of T element
C	Bit	C0–C255 C256–C511	9200–9455 10000-10511	01, 05, and 15	State of C element
D	Word	D0–D7999	0000–7999	03, 06, and 16	
SD	Word	SD0–SD255 SD256–SD1023	8000–8255 12000-12767	03, 06, and 16	
Z	Word	Z0–Z15	8500–8515	03, 06, and 16	
T	Word	T0–T255 T256–T511	9000–9255 11000-11255	03, 06, and 16	Current value of T element
C	Word	C0–C199	9500–9699	03, 06, and 16	Current value of C element (INT)
C	Double word	C200–C255	9700–9811	03 and 16	Current value of C element (DINT)
C	Double word	C256–C306	10000-10101	03 and 16	Current value of C element (DINT)
R	Word	R0–R32767	13000-45767	03, 06, and 16	

Note:

The protocol address is the address used on data transmission and corresponds with the logic address of Modicon data. The protocol address starts from 0 while the logic address of Modicon data starts from 1, that is, protocol address + 1 = logic address of Modicon data. For example, if M0 protocol address is 2000, and its corresponding logic address of Modicon data is 0:2001. In fact, the read and write of M0 is completed through the protocol address, for example: read M0 element frame (sent from the master station).



- Abnormal response description:

Abnormal code	Definition
0x01	Error function code
0x02	Error register address
0x03	Error data

- Notes

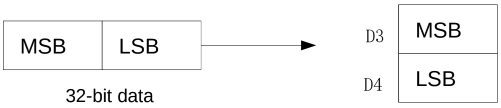
1. X and Y elements use the octal system. There are 256 points in total from X0 to X377, 256 points from Y0 to Y377, with the combinations ofY0–Y7, Y10–Y17, Y20–Y27, etc.

2. Two addressing methods are available for X element. One is the protocol address of 1200-1455 with corresponding function codes of 01, 05 and 15; the other is the protocol address of 0-255 with function code 02.

3. Processing of double-word elements: C element is a counter. It has its state and current value. C200–C255 are 32-bit elements, but each C element in the range occupies two protocol addresses during the protocol addressing. For example: The protocol address of C200 is 9700–9701. When reading the elements though Modbus, both the startprotocol address and the number of the read elements are even number.

4. For most SM and SD elements, the real value cannot be written through Modbus, but PLC salve station still returns "OK" to indicate the completion of write operation, which is allowable.

5. In addition, the Modbus communication protocol of the VC series supports the read and write of double word element, LONGINT variable and floating-point number. In VC series PLCs, 32-bit data are stored with high bits at high address. For example, a 32-bit data is stored in two D elements (D3 and D4), with MSB in D3 and LSB in D4, as shownin the following figure: (Refer to the description for the specific example)



3. Modbus function code description

3.1 Reading the coil state (0x01)

A maximum of 256 bit elements can be read in the VC series PLCs.

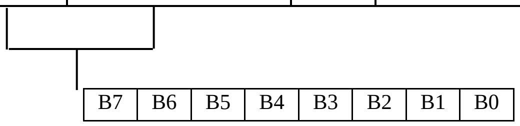
1. Request frame

Address	Function code (01H)	Start address		Number of elements		
		H	L	H	L	

2. Response frame

If the read address is not a multiple of 8, the remaining bits are filled with 0 (starting from the high bits).

Address	Function code (01H)	Number of read elements (n) (Number of bytes)	Read data No.1	……	Read data No.n	Check code (CRC or LRC)



3.2 (0x02)Reading the discrete input state (0x02)

In the VC series PLC, it specially refers to X element. The function code only supports the read function of X element with the max. read number of 256.

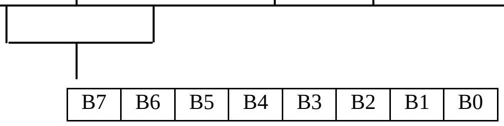
1. Request frame

Address	Function code (02H)	Start address		Number of elements		Check code (CRC or LRC)
		H	L	H	L	

2. Response frame

If the read address is not a multiple of 8, the remaining bits are filled with 0 (starting from the high bits).

Address	Function code (02H)	Number of read elements (n) (Number of bytes)	Read dataNo.1	……	Read data No.n	Check code (CRC or LRC)



3.3 Reading holding registers (0x03)

Reading holding registers refers to reading the value of data (word) register at the slave station, with a maximum of 125 registers to be read. It does not support the broadcast.

1. Request frame

Address	Function code	Start address	Number of elements	Check code

	(03H)	H	L	H	L	(CRC or LRC)

2. Response frame

Address	Function code (03H)	Number of read elements (n) (Number of bytes)	Read dataNo.1		……	Read data No.n		Check code (CRC or LRC)
			H	L		H	L	

3.4 Forcing (writing) single coil (0x05)

Forcing (writing) single coil writes the bit element value to the slave station and supports broadcast, that is, writing the same element to all slave stations. It supports a maximum of 1 bit element.

1. Request frame

Address	Function code (05H)	Start address		Written element value		Check code (CRC or LRC)
		H	L	H	L	

Note: The written value of the element is 0xFF00(ON,1) or 0x0000(OFF,0).

2. Response frame

Response frame is the repetition of request frame.

Address	Function code (05H)	Start address		Written element value		Check code (CRC or LRC)
		H	L	H	L	

3.5 Presetting (writing) single register (0x06)

Presetting (writing) single register (0x06) writes the word element value to the slave station andsupports broadcast, that is, writing the same element to all slave stations. It supports a maximum of 1 word element.

1. Request frame

Address	Function code (06H)	Start address		Written element value		Check code (CRC or LRC)
		H	L	H	L	

2. Response frame

Response frame is the repetition of request frame.

Address	Function code (06H)	Start address		Written element value		Check code (CRC or LRC)
		H	L	H	L	

3.6 Returning diagnostic check (0x08)

Diagnostic registers and communication error information can be obtained through the returning diagnostic check.

Diagnostic code	Meaning
0x00	Returning the request frame
0x 01	Restarting the communication option
0x 04	Listen only mode of the slave station
0x0a	Clearing counters and diagnostic registers
0x0b	Returning the bus message count
0x0c	Returning the bus CRC error count
0x0d	Returning the slave error count
0x0e	Returning the slave message count
0x0f	Returning the slave no response count
0x12	Returning the bus character overrun count

The frame description of sub-function code is as follows.

- Returning the request frame(0x00):

  1. Request frame

Address	Function code (0x08H)	Function word code		Any character		Check code (CRC or LRC)
		(0x00H)	(0x00H)	H	L	

2. Response frame

Returning the request frame intact.

Address	Function code	Function word code		Any character		Check code
	(0x08H)	(0x00H)	(0x00H)	H	L	(CRC or LRC)

- Restarting the communication option (0x01):

After receiving the frame, the PLC exit from the listen only mode (Broadcast frame is supported).

1. Request frame

The normal data field is 0x00 00 or 0xff 00.

Address	Function code	Function word code		Data field		Check code
	(0x08H)	0x00H	0x01H	H	L	(CRC or LRC)

2. Response frame

Address	Function code	Function word code		Data field		Check code
	(0x08H)	0x00H	0x01H	H	L	(CRC or LRC)

- Listen only mode of the slave station(0x04):

The slave station enters the listen only mode. None of the instruction is executed or responded. The slave station can only recognize the restarting communication option instruction and enters the online mode after receiving the instruction(Broadcast frame is supported).

1. Request frame

Address	Function code	Function word code		Data field		Check code
	(0x08H)	(0x00H)	(0x04H)	0x00H	0x00H	(CRC or LRC)

2. Response frame

No return.

- Clearing counters and diagnostic registers(0x0A):

Clearing all counters (Broadcast frame is supported).

1. Request frame

Address	Function code	Function word code		Data field		Check code
	(0x08H)	(0x00H)	(0x0AH)	0x00H	0x00H	(CRC or LRC)

2. Response frame

Address	Function code	Function word code		Data field		Check code
	(0x08H)	(0x00H)	(0x0AH)	0x00H	0x00H	(CRC or LRC)

- Returning the bus message count(0x0B):

Recording the total number of the messages to all master stations from the slave stations since the last starting, clearing and power-on of counters, which excludes the message of CRC error.

1. Request frame

Address	Function code	Function word code		Data field		Check code
	(0x08H)	(0x00H)	(0x0BH)	0x00H	0x00H	(CRC or LRC)

2. Response frame

Address	Function code	Function word code		Data field		Check code
	(0x08H)	(0x00H)	(0x0BH)	H	L	(CRC or LRC)

- CRC error count (0x0C):

Recording the number of CRC errors received by the slave station since the last starting, clearing and power-on of counters.

1. Request frame

Address	Function code (0x08H)	Function word code		Data field		Check code (CRC or LRC)
		(0x00H)	(0x0CH)	0x00H	0x00H	

2. Response frame

Address	Function code (0x08H)	Function word code		Data field		Check code (CRC or LRC)
		(0x00H)	(0x0CH)	H	L	

- Returningthe slave error count (0x0D):

Recording the number of errors that are detected by the slave station since the last starting, clearing and power-onof counters, which includes the error detected in the broadcast message.

1. Request frame

Address	Function code (0x08H)	Function word code		Data field		Check code (CRC or LRC)
		(0x00H)	(0x0DH)	0x00H	0x00H	

2. Response frame

Address	Function code (0x08H)	Function word code		Data field		Check code (CRC or LRC)
		(0x00H)	(0x0DH)	H	L	

- Returning the slave message count(0x0E):

Recording the number of the addressing messages received by the slave station since the last starting, clearing andpower-on of counters.

1. Request frame

Address	Function code (0x08H)	Function word code		Data field		Check code (CRC or LRC)
		(0x00H)	(0x0EH)	0x00H	0x00H	

2. Response frame

Address	Function code (0x08H)	Function word code		Data field		Check code (CRC or LRC)
		(0x00H)	(0x0EH)	H	L	

- Returning the slave no response count(0x0F):

Recording the number of messages that have not returned to the slave station since the last starting, clearing andpower-on of counters.

1. Request frame

Address	Function code (0x08H)	Function word code		Data field		Check code (CRC or LRC)
		(0x00H)	(0x0FH)	0x00H	0x00H	

2. Response frame

Address	Function code (0x08H)	Function word code		Data field		Check code (CRC or LRC)
		(0x00H)	(0x0FH)	H	L	

- Returning thebus character overrun count (0x12)

Recording the number of the messages that cannot be addressed due to the character overrun since the last starting,clearing and power-on of counters.

1. Request frame

Address	Function code (0x08H)	Function word code		Data field		Check code (CRC or LRC)
		(0x00H)	(0x12H)	0x00H	0x00H	

2. Response frame

Address	Function code (0x08H)	Function word code		Data field		Check code (CRC or LRC)
		(0x00H)	(0x12H)	H	L	

3.7 Forcing (Writing) multiple coils(0x0F)

A maximum of 1968 bit elements (0x07b0) can be written and the number is changeable according to the defined range.

1. Request frame

Address	Function code (0FH)	Start address		Number of elements		Number of bytes (n)	Written element value No.1	……	Written element value No.N	Check code (CRC or LRC)
		H	L	H	L					

B7	B6	B5	B4	B3	B2	B1	B0

2. Response frame

Address	Function code (0FH)	Start address		Number of elements		Check code (CRC or LRC)
		H	L	H	L	

3.8 Presetting (writing) multiple registers (0x10 Hex)

A maximum of 120 registers can be written (0x78).

1. Request frame

Address	Function code (0x10H)	Start address		Number of elements		Number of bytes (n)	Written element value No.1		……	Written element value No.N		Check code (CRC or LRC)
		H	L	H	L		H	L		H	L	

2. Response frame

Address	Function code (0x10H)	Start address		Number of elements		Check code (CRC or LRC)
		H	L	H	L	

3.9 Fault response frame (0x80+function code)

Response frame:

Address	Function code	Error code (Refer to 2. "Supported Modbus function code and element addressing mode"of appendix G	Check code (CRC or LRC)

Function code refers to the function code of the captured request frame ＋0x80.

3.10 Note

1. Refer to the address classification of elements, the read soft elements each time shall be of the same type. For example, X and Y elements cannot be read in one frame.

2. The address and data range of the element shall be within the range specified by the protocol. For example:

For Y element, the protocol address range is 0000﹣0255 (Y0-Y377):

- If the read start address is 1 and 256 elements are read, address error is returned (error code 02),because there are only 255 Y elements that start from 1.
- If the read start address is 0 and 257 elements are read, data error is returned (error code 03), becausethe actual defined number of Y elements is only 256.
- If the read start address is 0 and 256 elements are read, the states of 256 elements are returned.

In other words, the number of the read elements must be within the actually defined range. It is true for read/write ofbit/word elements.

4. Example of Modbus communication control

Modbus slave station responds to the message from the master station according to the received message of local addressing, rather than sending out message actively. The slave station only supports Modbus function codes 01, 02, 03, 05, 06, 08, 15, and 16, and the other codes are "illegal function codes"(except broadcast frame).

- Reading and writing elements

All supported function codes, except 08, are used for reading and writing elements. In principle, in oneframe, there are 2000 bit elements and 125 word elements for reading, 1968 bit elements and 120 word elements for writing at most. However, theactual protocol addresses for different types of elements are different and discontinuous (for example, Y377's protocol addressis 255 while X0's protocol address is 1200). Therefore, when reading or writing an element, the element read at one timecan only be the same type, and the max. number of the read/written elements depends on the number of the elements thatare actually defined. For example, when reading Y elements(Y0 – Y377), the protocol addressranges from 0 to 255, the corresponding logic address of Modicon data is from 1 to 256, and the max. number of the Y elements that can be readis 256.The examples are as follows:

1. Sending from the master station: 0101 000001003D 9A

01 – address; 01- function code; 00 00 – start address; 01 00 – number of the read elements; 3D 9A – check

Response of the slave station: providing correct response

2. Sending from the master station: 01 01 00 00 01 01 FC 5A

The master station reads 01 01 (257) elements, which is beyond thedefined number of Y elements.

Response of the slave station: 01 81 03 00 51

The response of the slave station is illegal address, because 257>256, and 256 is the allowed max. number of Y elements.

3. Sending from the master station: 01 01 00 64 00 A0 7D AD

00 64 (decimal: 100) is the start address for the master station to read, and 00 A0 (decimal: 160) is the number of the elements.

Response of the slave station: 01 81 02 C1 91

The response of the slave station is illegal address, because the protocol address 300 has no definition of bit elements.

The response of the slave station is illegal address, because there are only 156 Y elements which are defined to start from 100, and 160 Y elements have exceeded the specified number.

4. Sending from the master station: 01 01 01 2C 00 0A 7C 38

The master station read 10 bit elements of 01 2C (decimal: 300).

Response of slave station: 01 81 02 C1 91

The response of the slave station is illegal address, because the protocol address 300 has no definition of bit elements.

5. Sending from the master station: 01 04 00 02 00 0A D1 CD

The master station sends the frame of function code 04

Response of the slave station: 01 84 01 82 C0

The response of the slave station is illegal function code.

6. Sending from the master station: 01 02 00 00 00 0A F8 0D

The master station reads 10 (X0–X9) input elements (X elements) from the start address.

Response of the slave station:01 02 02 00 00 B9 B8

The slave station responds with correct information, which has 02 bytes, and the content is 00 00.

7. Sending from the master station: 01 01 04 B0 00 0A BC DA

The master station reads 10 bit elements (X0–X9) starting from 04 B0 (demical:1200).

Response of the slave station:01 01 02 00 00 B9 FC

The slave station responds with 02 bytes, and the content is 00 00.

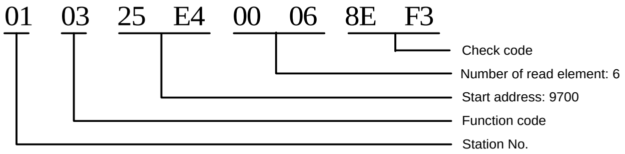
Note

1. The address of the slave station is 01, the last two bytes are CRC check codes and the second byte is function code.
2. X element does not support the write operation.

- Processing of double word elements

1. The current count value of C element is word or double word element. The values from C200 to C263 are double word elements, which are read and written through the function codes (03 and 16) of read/write registers. The address of every two registers corresponds to one C double word element, and the registers can only be read or written in pair.

For example, reading the RTU frame of three C double word elements from C200 to C202.



In the returned data, two addresses 9700 and 9701 indicate the contents of C200. 9700 is MSB, and 9701 is LSB.

2. When reading a double word element, if the start address for reading is not an even number, then the system responds with error code of illegal address. For example:

Sending from the master station: 0103 25E500 045E F2

The start address for the reading sent by the master station is 25 E5 (four word elements, decimal:9701).

Response of the slave station: 01 8302 C0 F1

Response of the slave station: Illegal data address

3. If the number of the read registers is not an even number, the system responds with error code of illegal data. For example:

Sending from the master station: 01 03 25 E4 00 05 CE F2

The start address for the master station reading is 25 E4 (5 word elements).

Response of the slave station: 01 83 03 01 31

The slave station returns the illegal data.

- Processing of LONG INT data

Based on the storage method of VC series PLCs, one LONG INT data can be saved in two D elements. For example, if a LONG INT data is saved in D3 and D4, VEICHI PLC think that MSB are stored in D3 and LSB are stored in D4. When the master station reads the LONG INT data through Modbus, the 32-bit data shall be regrouped based on the LONG INTstorage principle of VEICHI PLC after reading the data.

The storage principle of FLOAT is the same as the storage principle of LONG INT.

5. Description of broadcast

The slave station supports broadcast but not all the function codes. The slave station supports function codes 01, 02,03, 05, 06, 08, 15 and 16 (decimal), among which, 01, 02 and 03 can read element but do not support broadcast, noresponse is gotten after sending out the broadcast; 05, 06, 15 and 16 are can write element and support broadcast,no response is gotten after sending out the broadcast, but the slave station processes the received data; 08 is thediagnostic function code, it does not support the broadcast except its sub-function codes 0x01, 0x04 and 0x0A(hex).

Appendix G ASCII code character encoding table

ASCII HEX code		Most significant 3 bits							
		0	1	2	3	4	5	6	7
Least significant 4 bits	0	NUL	DLE	SPACE	0	@	P	、	p
	1	SOH	DC1	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	ETX	DC3	#	3	C	S	c	s
	4	EOT	DC4	$	4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
	6	ACK	SYN	&	6	F	V	f	v
	7	BEL	ETB	’	7	G	W	g	w
	8	BS	CAN	(	8	H	X	h	x
	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	{
	C	FF	FS	,	<	L		l	|
	D	CR	GS	-	=	M	]	m	}
	E	SO	RS	.	>	N	^	n	–
	F	SI	US	/	?	O	_	o	DEL

Appendix H Instruction order index table

Instruction		Instructionfunction	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
A	ACOS	Instruction for obtaining ACOS of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	115
	ADD	Integer addition instruction	7	Zero, carry, and borrow	√	√	√	√	√	98
	ANB	Energy flow block and instruction	1		√	√	√	√	√	61
	AND	NO contact and instruction	1		√	√	√	√	√	59
	AND<	Integer comparison AND< instruction	5		√	√	√	√	√	200
	AND<=	Integer comparison AND<= instruction	5		√	√	√	√	√	200
	AND<>	Integer comparison AND<> instruction	5		√	√	√	√	√	200
	AND=	Integer comparison AND= instruction	5		√	√	√	√	√	200
	AND>	Integer comparison AND> instruction	5		√	√	√	√	√	200
	AND>=	Integer comparison AND>= instruction	5		√	√	√	√	√	200
	ANDD<	Long integer comparison AND< instruction	7		√	√	√	√	√	203
	ANDD<=	Long integer comparison AND<= instruction	7		√	√	√	√	√	203
	ANDD<>	Long integer comparison AND<> instruction	7		√	√	√	√	√	203
	ANDD=	Long integer comparison AND= instruction	7		√	√	√	√	√	203
	ANDD>	Long integer comparison AND> instruction	7		√	√	√	√	√	203
	ANDD>=	Long integer comparison AND>= instruction	7		√	√	√	√	√	203
	ANDR<	Floating-point number comparison AND> instruction	7		√	√	√	√	√	206
	ANDR<=	Floating-point number comparison AND<= instruction	7		√	√	√	√	√	206
	ANDR<>	Floating-point number comparison AND<> instruction	7		√	√	√	√	√	206
	ANDR=	Floating-point number comparison AND= instruction	7		√	√	√	√	√	206
	ANDR>	Floating-point number comparison AND> instruction	7		√	√	√	√	√	206
	ANDR>=	Floating-point number comparison AND>= instruction	7		√	√	√	√	√	206
	ANI	NC contact and instruction	1		√	√	√	√	√	60
	ANR	Signal alarm reset instruction	1	Zero, carry, and borrow				√	√	249
	ANS	Signal alarm set instruction	7	Zero, carry, and borrow				√	√	248
	ASC	ASCII code conversion instruction	19	Zero, carry, and borrow	√	√	√	√	√	123
	ASIN	Instruction for obtaining ASIN of a floating-point number	7	Zero, carry, and borrow				√	√	115
	ATI	Instruction for converting an ASCII code to a 16-bit hex data	7	Zero, carry, and borrow	√	√	√	√	√	124
	ATAN	Instruction for obtaining ATAN of a floating-point number	7	Zero, carry, and borrow				√	√	115
	ALT	Alternate output instruction	11	Zero, carry, and borrow	√	√	√	√	√	184

Instruction		Instructionfunction	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	ABSD	Cam absolute control instruction	9	Zero, carry, and borrow			√	√	√	181
B	BAND	Word bit contact AND instruction	5		√	√	√	√	√	196
	BANI	Word bit contact ANI instruction	5		√	√	√	√	√	197
	BCD	Instruction for converting a word to a 16-bit BCD code	5	Zero, carry, and borrow	√	√	√	√	√	119
	BIN	BIN: Instruction for converting a 16-bit BCD code to a word	5	Zero, carry, and borrow	√	√	√	√	√	120
	BITS	Instruction for counting on bit in word	5		√	√	√	√	√	194
	BKADD	Bulk data addition operation instruction	9	Zero, carry, and borrow				√	√	210
	BKCMP=, >,<,<>,<=, >=	Bulk data comparison instruction	9					√	√	211
	BKSUB	Bulk data subtraction operation instruction	9	Zero, carry, and borrow				√	√	211
	BLD	Word bit contact LD instruction	5		√	√	√	√	√	195
	BLDI	Word bit contact LDI instruction	5		√	√	√	√	√	196
	BMOV	Block data transmission instruction	7		√	√	√	√	√	92
	BON	Instruction for judging on bit in word	7					√	√	195
	BOR	Word bit contact OR instruction	5		√	√	√	√	√	197
	BORI	Word bit contact ORI instruction	5		√	√	√	√	√	198
	BOUT	Word bit coil output instruction	5		√	√	√	√	√	198
B	BRST	Word bit coil reset instruction	5		√	√	√	√	√	199
	BSET	Word bit coil set instruction	5		√	√	√	√	√	198
	BTOW	Byte-unit data combination instruction	7	Zero, carry, and borrow				√	√	245
C	CALL	Instruction for calling the user subprogram			√	√	√	√	√	90
	CCITT	CCITT check instruction	7		√	√	√	√	√	191
	CCW	Counterclockwise arc trajectory interpolation	12	Zero, carry, and borrow				√	√	238
	CFEND	Instruction for conditional return of user main program	1		√	√	√	√	√	88
	CIRET	Instruction for conditional return of user interrupt program	1		√	√	√	√	√	89
	CJ	Conditional jump instruction	3		√	√	√	√	√	88
	COS	Instruction for obtaining COS of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	112
	CRC16	CRC16 check instruction	7		√	√	√	√	√	191
	CSRET	Instruction for conditional return of user subprogram	1		√	√	√	√	√	90
	CTR	16-bit cyclic counting instruction	5		√	√	√	√	√	76
	CTU	16-bit increment counter instruction	5		√	√	√	√	√	76
	CMP	Instruction for setting integer comparison to ON	7				√	√	√	209
	CW	Clockwise arc trajectory interpolation	12	Zero, carry, and borrow				√	√	237
D	DADD	Long integer addition instruction	10	Zero, carry, and borrow	√	√	√	√	√	102
	DBAND	Deadband control instruction	9	Zero, carry, and borrow				√	√	213

Instruction	Instructionfunction	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
DBCD	Instruction for converting a double word to a 32-bit BCD code	7	Zero, carry, and borrow	√	√	√	√	√	120
DBIN	Instruction for converting a 32-bit BCD code to a double word	7		√	√	√	√	√	121
DBITS	Instruction for counting on bit in double word	6		√	√	√	√	√	195
DCMP<	Date < comparison instruction	7			√	√	√	√	153
DCMP<=	Date <= comparison instruction	7			√	√	√	√	153
DCMP<>	Date <> comparison instruction	7			√	√	√	√	153
DCMP=	Date = comparison instruction	7			√	√	√	√	153
DCMP>	Date > comparison instruction	7			√	√	√	√	153
DCMP>=	Date >= comparison instruction	7			√	√	√	√	153
DCNT	32-bit increment and decrement counting instruction	7		√	√	√	√	√	77
DDEC	Long integer minus one instruction	4	Zero, carry, and borrow	√	√	√	√	√	105
DDIV	Long integer division instruction	10	Zero, carry, and borrow	√	√	√	√	√	103
DEC	Integer minus one instruction	3	Zero, carry, and borrow	√	√	√	√	√	101
DEG	Instruction for floating-point number radian-angle conversion	7	Zero, carry, and borrow				√	√	116
DECO	Decoding instruction	5		√	√	√	√	√	193
DFLT	Instruction for converting a long integer to a floating-point number	7	Zero, carry, and borrow			√	√	√	118
DFMOV	Data block double word fill instruction	9		√	√	√	√	√	93
DFROM	Instruction for reading double words from a special module buffer register	10				√	√	√	141
DGBIN	Instruction for converting a 32-bit gray code to a double word	7	Zero, carry, and borrow	√	√	√	√	√	122
DGRY	Instruction for converting a double word to a 32-bit gray code	7	Zero, carry, and borrow	√	√	√	√	√	121
DHSCI	Instruction for triggering interrupt based on comparison of high-speed count	10		√	√	√	√	√	157
DHSCR	Instruction for resetting the high-speed count comparison	10		√	√	√	√	√	159
DHSCS	Instruction of setting the high-speed count comparison	10		√	√	√	√	√	156
DHSP	Instruction for pulse output based on high-speed count table comparison	10		√	√	√	√	√	164
DHSPI	Instruction for triggering interrupt based on comparison of absolute high-speed output positions	10				√	√	√	159
DHST	High-speed count table comparison instruction	10		√	√	√	√	√	162
DHSZ	High-speed count range comparison instruction	13		√	√	√	√	√	161
DI	Disable interrupt instruction	1		√	√	√	√	√	89
DINC	Long integer plus one instruction	4	Zero, carry, and borrow	√	√	√	√	√	104

Instruction		Instructionfunction	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	DINT	Instruction for convert a floating-point number to a long integer	7	Zero, carry, and borrow	√	√	√	√	√	119
D	DIS	Instruction for separating 4 bitsof 16-bit data	7	Zero, carry, and borrow				√	√	247
	DIV	Integer division instruction	7		√	√	√	√	√	99
	DMOV	Double word data transmission instruction	7		√	√	√	√	√	91
	DMUL	Long integer multiplication instruction	10	Zero, carry, and borrow	√	√	√	√	√	103
	DNEG	Long integer negation instruction	7	Zero, carry, and borrow	√	√	√	√	√	106
	DRCL	Instruction for 32-bit rotate left with carry flag bit	9	Carry	√	√	√	√	√	136
	DRCR	Instruction for 32-bit rotate right with carry flag bit	9	Carry	√	√	√	√	√	135
	DROL	32-bit rotate left instruction	9	Carry	√	√	√	√	√	135
	DROR	32-bit rotate right instruction	9	Carry	√	√	√	√	√	134
	DRVA	Absolute position control instruction	11	Zero, carry, and borrow	√	√	√	√	√	230
	DRVI	Relative position control instruction	11	Zero, carry, and borrow	√	√	√	√	√	229
	DSHL	32-bit shift left instruction	9		√	√	√	√	√	138
	DSHR	32-bit shift right instruction	9		√	√	√	√	√	138
	DSQT	Instruction for extracting the square root of a long integer	7	Zero, carry, and borrow	√	√	√	√	√	104
	DSUB	Long integer subtraction instruction	10	Zero, carry, and borrow	√	√	√	√	√	102
	DSUM	Long integer accumulation instruction	9	Zero, carry, and borrow	√	√	√	√	√	108
	DTI	Instruction for converting a long integer to an integer	6	Zero, carry, and borrow	√	√	√	√	√	117
	DTO	Instruction for writing double words from a special module buffer register	10				√	√	√	143
	DUTY	Instruction for generating timed pulses	7					√	√	250
	DVABS	Instruction for obtaining the absolute value of a long integer	7	Zero, carry, and borrow	√	√	√	√	√	105
	DWAND	Double word AND instruction	10		√	√	√	√	√	130
	DWINV	Double word negation instruction	10		√	√	√	√	√	131
	DWOR	Double word OR instruction	10		√	√	√	√	√	130
	DWXOR	Double word XOR instruction	10		√	√	√	√	√	131
	DXCH	Double word swop instruction	7		√	√	√	√	√	94
	DABSD	Cam absolute control instruction	11	Zero, carry, and borrow			√	√	√	183
	DSZR	Instruction for zero return with DOG	9	Zero, carry, and borrow		√	√	√	√	231
	DVIT	Interrupt positioning instruction	11	Zero, carry, and borrow		√				232
E	ED	Falling edge detection instruction	1		√	√	√	√	√	64
	EI	Enable interrupt instruction	1		√	√	√	√	√	89
	ENCO	Encoding instruction	5		√	√	√	√	√	194
	EROMWR	EEPROM write instruction	7							145
	EU	Rising edge detection instruction	2		√	√	√	√	√	64

Instruction		Instructionfunction	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	EXP	Instruction for obtaining the natural number power of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	114
F	FIFO	First-in-first-out instruction	7		√	√	√	√	√	95
	FLT	Instruction for converting an integer to a floating-point number	6	Zero, carry, and borrow	√	√	√	√	√	118
	FMOV	Data block fill instruction	7		√	√	√	√	√	92
	FOR	Cycle instruction	3		√	√	√	√	√	85
	FROM	Instruction for reading words from a special module buffer register	9				√	√	√	141
G	GBIN	Instruction for converting a 16-bit gray code to a word	5	Zero, carry, and borrow	√	√	√	√	√	122
	GRY	Instruction for converting a word to a 16-bit gray code	5	Zero, carry, and borrow	√	√	√	√	√	121
H	HACKLE	Sawtooth wave signal output instruction	12		√	√	√	√	√	179
	HCNT	Instruction for driving the high-speed counter	7		√	√	√	√	√	156
	HOUR	Chronograph instruction	8		√	√	√	√	√	152
	HTOS	Instruction for converting hour-minute-second data to seconds	5					√	√	155
I	INC	Integer plus one instruction	3	Zero, carry, and borrow	√	√	√	√	√	100
	INITR	Instruction for initializing an extension register	5	Zero, carry, and borrow						223
	INT	Instruction for converting a floating-point number to an integer	6	Zero, carry, and borrow	√	√	√	√	√	118
	INV	Energy flow negation instruction	1		√	√	√	√	√	157
	ITA	Instruction for converting a 16-bit hex data to an ASCII code	7	Zero, carry, and borrow	√	√	√	√	√	124
	ITD	Instruction for converting an integer to a long integer	6	Zero, carry, and borrow	√	√	√	√	√	117
L	LBL	Jump label definition instruction	3		√	√	√	√	√	86
	LCNV	Project conversion instruction	9	Zero, carry, and borrow			√	√	√	125
	LD	NO contact instruction	1		√	√	√	√	√	199
	LD<	Integer comparison LD<instruction	5		√	√	√	√	√	199
	LD<=	Integer comparison LD<=instruction	5		√	√	√	√	√	199
	LD<>	Integer comparison LD<>instruction	5		√	√	√	√	√	199
	LD=	Integer comparison LD=instruction	5		√	√	√	√	√	199
	LD>	Integer comparison LD=instruction	5		√	√	√	√	√	199
	LD>=	Integer comparison LD>=instruction	5		√	√	√	√	√	199
	LDD<	Long integer comparison LD<instruction	7		√	√	√	√	√	202
	LDD<=	Long integer comparison LD<=instruction	7		√	√	√	√	√	202
	LDD<>	Long integer comparison LD<>instruction	7		√	√	√	√	√	202

Instruction		Instructionfunction	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	LDD=	Long integer comparison LD=instruction	7		√	√	√	√	√	202
	LDD>	Long integer comparison LD>instruction	7		√	√	√	√	√	202
	LDD>=	Long integer comparison LD>=instruction	7		√	√	√	√	√	202
	LDI	NC contact instruction	1		√	√	√	√	√	59
	LDR<	Floating-point number comparison LD<instruction	7		√	√	√	√	√	205
	LDR<=	Floating-point number comparison LD<=instruction	7		√	√	√	√	√	205
	LDR<>	Floating-point number comparison LD<>instruction	7		√	√	√	√	√	205
	LDR=	Floating-point number comparison LD=instruction	7		√	√	√	√	√	205
	LDR>	Floating-point number comparison LD>instruction	7		√	√	√	√	√	205
	LDR>=	Floating-point number comparison LD>=instruction	7		√	√	√	√	√	205
	LIFO	Last-in-first-out instruction	7		√	√	√	√	√	96
	LIMIT	Upper/lower limit control instruction	9	Zero, carry, and borrow				√	√	212
	LIN	Linear trajectory interpolation instruction	12	Zero, carry, and borrow				√	√	234
	LN	Instruction for obtaining the natural logarithm of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	113
	LOADR	Instruction for reading data from an extension file register	5	Zero, carry, and borrow						221
	LOG	Instruction for obtaining the common logarithm of a floating-point number	7	Zero, carry, and borrow				√	√	116
	LOGR	Instruction for logging on an extension register	11	Zero, carry, and borrow						224
	LRC	LRC check instruction	7		√	√	√	√	√	192
M	MC	Main control instruction	3		√	√	√	√	√	70
	MCR	Main control reset instruction	1		√	√	√	√	√	70
	MEAN	Mean instruction	7	Zero, carry, and borrow				√	√	244
	Modbus	Modbus master station communication instruction	8		√	√	√	√	√	184
	MOV	Word data transmission instruction	5		√	√	√	√	√	91
	MOVLINK	Synchronous control instruction	17	Zero, carry, and borrow						238
	MPP	Output energy flow stack pop instruction	1		√	√	√	√	√	63
	MPS	Output energy flow push instruction	1		√	√	√	√	√	62
	MRD	Instruction for reading output energy flow stack top value	1		√	√	√	√	√	63
	MUL	Integer negation instruction	8	Zero, carry, and borrow	√	√	√	√	√	99
	MODRW	Modbus read/write instruction	14		√	√	√	√	√	187
N	NEG	Integer negation instruction	5	Zero, carry, and borrow	√	√	√	√	√	102
	NEXT	Cycle return instruction	1		√	√	√	√	√	85
	NOP	No operation instruction	1		√	√	√	√	√	70
O	OR	NO contact or instruction	1		√	√	√	√	√	60

Instruction		Instructionfunction	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	OR<	Integer comparison OR<instruction	5		√	√	√	√	√	201
	OR<=	Integer comparison OR<=instruction	5		√	√	√	√	√	201
	OR<>	Integer comparison OR<>instruction	5		√	√	√	√	√	201
	OR=	Integer comparison OR=instruction	5		√	√	√	√	√	201
	OR>	Integer comparison OR>instruction	5		√	√	√	√	√	201
	OR>=	Integer comparison OR>=instruction	5		√	√	√	√	√	201
	ORB	Energy flow block or instruction	1		√	√	√	√	√	62
	ORD<	Long integer comparison OR<instruction	7		√	√	√	√	√	204
	ORD<=	Long integer comparison OR<=instruction	7		√	√	√	√	√	204
	ORD<>	Long integer comparison OR<>instruction	7		√	√	√	√	√	204
	ORD=	Long integer comparison OR=instruction	7		√	√	√	√	√	204
	ORD>	Long integer comparison OR>instruction	7		√	√	√	√	√	204
	ORD>=	Long integer comparison OR>=instruction	7		√	√	√	√	√	204
	ORI	NO contact or instruction	1		√	√	√	√	√	61
	ORR<	Floating-point number comparison OR>instruction	7		√	√	√	√	√	207
	ORR<=	Floating-point number comparison OR<=instruction	7		√	√	√	√	√	207
	ORR<>	Floating-point number comparison OR<>instruction	7		√	√	√	√	√	207
	ORR=	Floating-point number comparison OR=instruction	7		√	√	√	√	√	207
	ORR>	Floating-point number comparison OR>instruction	7		√	√	√	√	√	207
	ORR>=	Floating-point number comparison OR>=instruction	7		√	√	√	√	√	207
	OUT	Coil output instruction	1		√	√	√	√	√	61
	OUT Sxx	SFC state jump instruction	3		√	√	√	√	√	72
P	PID	PID function instruction	9		√	√	√	√	√	174
	PLS	Envelop line pulse output instruction	7			√	√	√	√	171
	PLSR	Instruction for count pulse output with acceleration/deceleration	10		√	√	√	√	√	168
	PLSV	Variable speed pulse output instruction	8	Zero, carry, and borrow	√	√	√	√	√	228
	PLSY	High-speed pulse output instruction	9		√	√	√	√	√	167
	POWER	Instruction for exponentiation of a floating-point number	10	Zero, carry, and borrow	√	√	√	√	√	113
	PUSH	Data push instruction	7		√	√	√	√	√	94
	PWM	Pulse output instruction	7		√	√	√	√	√	172
R	RAD	Instruction for floating-point number angle-radian conversion	7	Zero, carry, and borrow			√	√	√	116

Instruction		Instructionfunction	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	RADD	Floating-point number addition instruction	10	Zero, carry, and borrow	√	√	√	√	√	108
	RAMP	Ramp signal output instruction	12		√	√	√	√	√	178
	RCL	Instruction for 16-bit rotate left with carry flag bit	7	Carry	√	√	√	√	√	134
	RCR	Instruction for 16-bit rotate right with carry flag bit	7	Carry	√	√	√	√	√	133
	RCV	Free-port receiving instruction	7		√	√	√	√	√	186
	RDIV	Floating-point number division instruction	10	Zero, carry, and borrow	√	√	√	√	√	110
	REF	Instruction for immediately refreshing I/O	5		√	√	√	√	√	144
	REFF	Instruction for setting input filtering constant	3		√	√	√	√	√	144
	RET	SFC program segment end instruction	1		√	√	√	√	√	73
	RLCNV	Floating-point project conversion instruction	12	Zero, carry, and borrow			√	√	√	126
	RMOV	Floating-point data transmission instruction	7		√	√	√	√	√	92
	RMUL	Floating-point number multiplication instruction	10	Zero, carry, and borrow	√	√	√	√	√	109
	RND	Instruction for generating random numbers	3	Zero				√	√	249
	RNEG	Floating-point number negation instruction	7	Zero, carry, and borrow	√	√	√	√	√	111
	ROL	16-bit rotate left instruction	7	Carry	√	√	√	√	√	132
	ROR	16-bit rotate right instruction	7	Carry	√	√	√	√	√	132
	RSQT	Instruction for extracting the square root of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	110
	RST	Coil reset instruction	1		√	√	√	√	√	69
	RST Sxx	SFC state reset instruction	3		√	√	√	√	√	72
	RSUB	Floating-point number subtraction instruction	10	Zero, carry, and borrow	√	√	√	√	√	109
	RSUM	Floating-point number accumulation instruction	9	Zero, carry, and borrow	√	√	√	√	√	114
	RVABS	Instruction for obtaining the absolute value of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	111
	RCMP	Instruction for setting floating-point number comparison to ON	9				√	√	√	210
S	SAVER	Instruction for writing data to an extension file register	7	Zero, carry, and borrow						222
	SCL	Coordinate setting instruction	7	Zero, carry, and borrow				√	√	214
	SEG	Instruction for converting a word to a 7-segment code	5	Zero, carry, and borrow	√	√	√	√	√	123
	SER	Data search instruction	9	Zero, carry, and borrow				√	√	215
	SET	Coil set instruction	1		√	√	√	√	√	69
	SET Sxx	SFC state transition instruction	3		√	√	√	√	√	72
	SFTL	Bit string shift left instruction	9		√	√	√	√	√	139
	SFTR	Bit string shift right instruction	9		√	√	√	√	√	139
	SHL	16-bit shift left instruction	7		√	√	√	√	√	137
	SHR	16-bit shift right instruction	7		√	√	√	√	√	136
	SIN	Instruction for obtaining SIN of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	111

Instruction		Instructionfunction	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	SPD	Frequency measuring instruction	7		√	√	√	√	√	166
	SQT	Instruction for extracting the square root of an integer	5	Zero, carry, and borrow	√	√	√	√	√	100
	STL	SFC state loading instruction	3		√	√	√	√	√	71
	STOH	Instruction for converting seconds to hour-minute-second data	5					√	√	155
	STOP	Instruction for stopping the user program	1		√	√	√	√	√	89
	STRADD	String combination instruction	7	Zero, carry, and borrow				√	√	216
	STRINSTR	String search instruction	9	Zero, carry, and borrow				√	√	220
	STRLEFT	Instruction for reading a string from left	7	Zero, carry, and borrow				√	√	218
	STRLEN	Instruction for detecting the string length	5	Zero, carry, and borrow				√	√	217
	STRMIDR	Instruction for reading any characters of a string	7	Zero, carry, and borrow				√	√	219
	STRMIDW	Instruction for replacing any characters of a string	7	Zero, carry, and borrow				√	√	219
	STRMOV	String transmission instruction	5	Zero, carry, and borrow				√	√	221
	STRRIGHT	Instruction for reading a string from right	7	Zero, carry, and borrow				√	√	217
	SUB	Integer subtraction instruction	7	Zero, carry, and borrow	√	√	√	√	√	99
	SUM	Integer accumulation instruction	8	Zero, carry, and borrow	√	√	√	√	√	106
	SWAP	MSB/LSB swop instruction	3		√	√	√	√	√	93
T	TADD	Clock addition instruction	7	Zero and carry		√	√	√	√	150
	TAN	Instruction for obtaining TAN of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	112
	TCMP<	Time < comparison instruction	7			√	√	√	√	154
	TCMP<=	Time >= comparison instruction	7			√	√	√	√	154
	TCMP<>	Time <> comparison instruction	7			√	√	√	√	154
	TCMP=	Time = comparison instruction	7			√	√	√	√	154
	TCMP>	Time > comparison instruction	7			√	√	√	√	154
	TCMP>=	Time >= comparison instruction	7			√	√	√	√	154
	TKY	Numeric key input instruction	7					√	√	146
	TMON	Non-retriggerable monostable timing instruction	5		√	√	√	√	√	75
	TO	Instruction for writing words from a special module buffer register	9				√	√	√	142
	TOF	Turn-off delay timing instruction	5		√	√	√	√	√	74
	TON	Turn-on delay timing instruction	5		√	√	√	√	√	73
	TONR	Memory-type turn-on delay timing instruction	5		√	√	√	√	√	73
	TRD	Real-time clock read instruction	3			√	√	√	√	148
	TRIANGLE	Triangle wave signal output instruction	12		√	√	√	√	√	180
	TSUB	Clock subtraction instruction	7	Zero and borrow		√	√	√	√	151
	TWR	Real-time clock write instruction	3			√	√	√	√	149
U	UNI	Instruction for combining 4 bits of 16-bit data	7	Zero, carry, and borrow				√	√	246

Instruction		Instructionfunction	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
V	VABS	Instruction for obtaining the absolute value of an integer	5	Zero, carry, and borrow	√	√	√	√	√	101
	VRRD	Instruction for reading the value of an analog potentiometer	5							143
W	WAND	Word AND instruction	7		√	√	√	√	√	127
	WDT	Instruction for watchdog reset of user program	1		√	√	√	√	√	88
	WINV	Word INV instruction	5		√	√	√	√	√	130
	WOR	Word OR instruction	7		√	√	√	√	√	129
	WSFL	Word string shift left instruction	9	Zero, carry, and borrow	√	√	√	√	√	97
	WSFR	Word string shift right instruction	9	Zero, carry, and borrow	√	√	√	√	√	96
	WTOB	Byte-unit data separation instruction	7	Zero, carry, and borrow				√	√	244
	WXOR	Word XOR instruction	7		√	√	√	√	√	129
X	XCH	Word swop instruction	5		√	√	√	√	√	94
	XMT	Free-port sending instruction	7		√	√	√	√	√	185
Z	ZONE	Zone control instruction	9	Zero, carry, and borrow				√	√	213
	ZRN	Zero return instruction	11	Zero, carry, and borrow	√	√	√	√	√	227
	ZRST	Instruction for resetting bits to 0 in batch	5		√	√	√	√	√	193
	ZSET	Instruction for resetting bits in batch	5		√	√	√	√	√	193
Note:										

Appendix I Instruction classification index table

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
Basic instructions	LD	NO contact instruction	1		√	√	√	√	√	59
	LDI	NC contact instruction	1		√	√	√	√	√	59
	AND	NO contact and instruction	1		√	√	√	√	√	59
	ANI	NC contact and instruction	1		√	√	√	√	√	60
	OR	NO contact or instruction	1		√	√	√	√	√	60
	ORI	NC contact or instruction	1		√	√	√	√	√	61
	OUT	Coil output instruction	1		√	√	√	√	√	61
	SET	Coil set instruction	1		√	√	√	√	√	69
	RST	Coil reset instruction	1		√	√	√	√	√	69
	ANB	Energy flow block and instruction	1		√	√	√	√	√	61
	ORB	Energy flow block or instruction	1		√	√	√	√	√	62
	INV	Energy flow negation instruction	1		√	√	√	√	√	69
	NOP	No operation instruction	1		√	√	√	√	√	70
	MPS	Output energy flow push instruction	1		√	√	√	√	√	62
	MRD	Instruction for reading output energy flow stack top value	1		√	√	√	√	√	63
	MPP	Output energy flow stack pop instruction	1		√	√	√	√	√	63
	MC	Main control instruction	3		√	√	√	√	√	70
	MCR	Main control reset instruction	1		√	√	√	√	√	70
	EU	Rising edge detection instruction	2		√	√	√	√	√	64
	ED	Falling edge detection instruction	2		√	√	√	√	√	64
	TON	Turn-on delay timing instruction	5		√	√	√	√	√	73
	TOF	Turn-off delay timing instruction	5		√	√	√	√	√	74
	TMON	Non-retriggerable monostable timing instruction	5		√	√	√	√	√	75
	TONR	Memory-type turn-on delay timing instruction	5		√	√	√	√	√	73
	CTU	16-bit increment counter instruction	5		√	√	√	√	√	76
	CTR	16-bit cyclic counting instruction	5		√	√	√	√	√	76

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	DCNT	32-bit increment and decrement counting instruction	7		√	√	√	√	√	77
Program flow control instructions	LBL	Jump label definition instruction	3		√	√	√	√	√	86
	CJ	Conditional jump instruction	3		√	√	√	√	√	88
	CALL	Instruction for calling the user subprogram			√	√	√	√	√	90
	CSRET	Instruction for conditional return of user subprogram	1		√	√	√	√	√	90
	CFEND	Instruction for conditional return of user main program	1		√	√	√	√	√	88
	CIRET	Instruction for conditional return of user interrupt program	1		√	√	√	√	√	89
	FOR	Cycle instruction	3		√	√	√	√	√	85
	NEXT	Cycle return instruction	1		√	√	√	√	√	85
	WDT	Instruction for watchdog reset of user program	1		√	√	√	√	√	88
	STOP	Instruction for stopping the user program	1		√	√	√	√	√	89
	EI	Enable interrupt instruction	1		√	√	√	√	√	89
	DI	Disable interrupt instruction	1		√	√	√	√	√	89
SFC instructions	STL	SFC state loading instruction	3		√	√	√	√	√	71
	SET Sxx	SFC state transition instruction	3		√	√	√	√	√	72
	OUT Sxx	SFC state jump instruction	3		√	√	√	√	√	72
	RST Sxx	SFC state reset instruction	3		√	√	√	√	√	72
	RET	SFC program segment end instruction	1		√	√	√	√	√	73
Data transmission instructions	MOV	Word data transmission instruction	5		√	√	√	√	√	91
	DMOV	Double word data transmission instruction	7		√	√	√	√	√	91
	RMOV	Floating-point data transmission instruction	7		√	√	√			92
	BMOV	Block data transmission instruction	7		√	√	√	√	√	92
	SWAP	MSB/LSB swop instruction	3		√	√	√	√	√	93
Data flow	XCH	Word swop instruction	5		√	√	√	√	√	94

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	DXCH	Double word swop instruction	7		√	√	√	√	√	94
	FMOV	Data block fill instruction	7		√	√	√	√	√	92
	DFMOV	Data block double word fill instruction	9		√	√	√	√	√	93
	WSFR	Word string shift right instruction	9	Zero, carry, and borrow	√	√	√	√	√	96
	WSFL	Word string shift left instruction	9	Zero, carry, and borrow	√	√	√	√	√	97
	PUSH	Data push instruction	7		√	√	√	√	√	94
	FIFO	First-in-first-out instruction	7		√	√	√	√	√	95
	LIFO	Last-in-first-out instruction	7		√	√	√	√	√	96
Integer/long integer arithmetic operation instructions	ADD	Integer addition instruction	7	Zero, carry, and borrow	√	√	√	√	√	98
	DADD	Long integer addition instruction	10	Zero, carry, and borrow	√	√	√	√	√	102
	SUB	Integer subtraction instruction	7	Zero, carry, and borrow	√	√	√	√	√	99
	DSUB	Long integer subtraction instruction	10	Zero, carry, and borrow	√	√	√	√	√	102
	INC	Integer plus one instruction	3	Zero, carry, and borrow	√	√	√	√	√	100
	DINC	Long integer plus one instruction	4	Zero, carry, and borrow	√	√	√	√	√	104
	DEC	Integer minus one instruction	3	Zero, carry, and borrow	√	√	√	√	√	101
	DDEC	Long integer minus one instruction	4	Zero, carry, and borrow	√	√	√	√	√	105
	MUL	Integer multiplication instruction	8	Zero, carry, and borrow	√	√	√	√	√	99
	DMUL	Long integer multiplication instruction	10	Zero, carry, and borrow	√	√	√	√	√	103
	DIV	Integer division instruction	7		√	√	√	√	√	99
	DDIV	Long integer division instruction	10	Zero, carry, and borrow	√	√	√	√	√	103
	VABS	Instruction for obtaining the absolute value of an integer	5	Zero, carry, and borrow	√	√	√	√	√	101
	DVABS	Instruction for obtaining the absolute value of a long integer	7	Zero, carry, and borrow	√	√	√	√	√	105
	NEG	Integer negation instruction	5	Zero, carry, and borrow	√	√	√	√	√	102
	DNEG	Long integer negation instruction	7	Zero, carry, and borrow	√	√	√	√	√	106
	SQT	Instruction for extracting the square root of an integer	5	Zero, carry, and borrow	√	√	√	√	√	100
	DSQT	Instruction for extracting the square root of a long integer	7	Zero, carry, and borrow	√	√	√	√	√	104

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	SUM	Integer accumulation instruction	8	Zero, carry, and borrow	√	√	√	√	√	106
	DSUM	Long integer accumulation instruction	9	Zero, carry, and borrow	√	√	√	√	√	108
Floating-point arithmetic operation instructions	RADD	Floating-point number addition instruction	10	Zero, carry, and borrow	√	√	√	√	√	108
	RSUB	Floating-point number subtraction instruction	10	Zero, carry, and borrow	√	√	√	√	√	109
	RMUL	Floating-point number multiplication instruction	10	Zero, carry, and borrow	√	√	√	√	√	109
	RDIV	Floating-point number division instruction	10	Zero, carry, and borrow	√	√	√	√	√	110
	RVABS	Instruction for obtaining the absolute value of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	111
	RNEG	Floating-point number negation instruction	7	Zero, carry, and borrow	√	√	√	√	√	111
	RSQT	Instruction for extracting the square root of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	110
	SIN	Instruction for obtaining SIN of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	111
	COS	Instruction for obtaining COS of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	112
	TAN	Instruction for obtaining TAN of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	112
	LN	Instruction for obtaining the natural logarithm of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	113
	EXP	Instruction for obtaining the natural number power of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	114
	POWER	Instruction for exponentiation of a floating-point number	10	Zero, carry, and borrow	√	√	√	√	√	113
Floating-point arithmetic operation instructions	RSUM	Floating-point number accumulation instruction	9	Zero, carry, and borrow	√	√	√	√	√	114
	ASIN	Instruction for obtaining ASIN of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	115
	ACOS	Instruction for obtaining ACOS of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	115
	ATAN	Instruction for obtaining ATAN of a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	115

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	RAD	Instruction for floating-point number angle-radian conversion	7	Zero, carry, and borrow			√	√	√	116
	DEG	Instruction for floating-point number radian-angle conversion	7	Zero, carry, and borrow			√	√	√	116
	LOG	Instruction for obtaining the common logarithm of a floating-point number	7	Zero, carry, and borrow			√	√	√	116
Word/double word logic operation instructions	WAND	Word AND instruction	7		√	√	√	√	√	127
	DWAND	Double word AND instruction	10		√	√	√	√	√	130
	WOR	Word OR instruction	7		√	√	√	√	√	129
	DWOR	Double word OR instruction	10		√	√	√	√	√	130
	WXOR	Word XOR instruction	7		√	√	√	√	√	130
	DWXOR	Double word XOR instruction	10		√	√	√	√	√	131
	WINV	Word INV instruction	5		√	√	√	√	√	130
	DWINV	Double word negation instruction	7		√	√	√	√	√	131
Bit shift and rotate instructions	ROR	16-bit rotate right instruction	7	Carry	√	√	√	√	√	132
	DROR	32-bit rotate right instruction	9	Carry	√	√	√	√	√	134
	ROL	16-bit rotate left instruction	7	Carry	√	√	√	√	√	132
	DROL	32-bit rotate left instruction	9	Carry	√	√	√	√	√	135
	RCR	Instruction for 16-bit rotate right with carry flag bit	7	Carry	√	√	√	√	√	133
	DRCR	Instruction for 32-bit rotate right with carry flag bit	9	Carry	√	√	√	√	√	135
	RCL	Instruction for 16-bit rotate left with carry flag bit	7	Carry	√	√	√	√	√	134
	DRCL	Instruction for 32-bit rotate left with carry flag bit	9	Carry	√	√	√	√	√	136
	SHR	16-bit shift right instruction	7		√	√	√	√	√	136
	DSHR	32-bit shift right instruction	9		√	√	√	√	√	138
	SHL	16-bit shift left instruction	7		√	√	√	√	√	137
	DSHL	32-bit shift left instruction	9		√	√	√	√	√	138
	SFTL	Bit string shift left instruction	9		√	√	√	√	√	139
	SFTR	Bit string shift right instruction	9		√	√	√	√	√	139
	DECO	Decoding instruction	5		√	√	√	√	√	193

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
Enhanced bit processing instructions	ENCO	Encoding instruction	5		√	√	√	√	√	194
	BITS	Instruction for counting on bit in word	5		√	√	√	√	√	194
	DBITS	Instruction for counting on bit in double word	6		√	√	√	√	√	195
	ZRST	Instruction for resetting bits to 0 in batch	5		√	√	√	√	√	193
	ZSET	Instruction for resetting bits in batch	5		√	√	√	√	√	193
	BON	Instruction for judging on bit in word	7					√	√	195
High-speed I/O instructions	HCNT	High-speed counter drive instruction	7		√	√	√	√	√	156
	DHSCS	High-speed counting comparison set instruction	10		√	√	√	√	√	156
	DHSCR	High-speed counting comparison reset instruction	10		√	√	√	√	√	159
	DHSCI	High-speed counting comparison interrupt trigger instruction	10		√	√	√	√	√	157
	DHSZ	High-speed count range comparison instruction	13		√	√	√	√	√	161
	DHST	High-speed count table comparison instruction	10		√	√	√	√	√	162
	DHSP	Instruction for pulse output based on high-speed count table comparison	10		√	√	√	√	√	164
	SPD	Frequency measuring instruction	7		√	√	√	√	√	166
	PLSY	High-speed pulse output instruction	9		√	√	√	√	√	167
	PLSR	Instruction for count pulse output with acceleration/deceleration	10		√	√	√	√	√	168
	PWM	PWM pulse output instruction	7		√	√	√	√	√	172
	PLS	Envelop line pulse output instruction	7			√	√	√	√	172
Control calculation instructions	PID	Function instruction	9		√	√	√	√	√	174
	RAMP	Ramp signal output instruction	12		√	√	√	√	√	178
	TRIANGLE	Triangle wave signal output instruction	12		√	√	√	√	√	180
	HACKLE	Sawtooth wave signal output instruction	12		√	√	√	√	√	179
	ABSD	Cam absolute control instruction	9	Zero, carry, and borrow			√	√	√	181

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	DABSD	Double word cam absolute control instruction	11	Zero, carry, and borrow			√	√	√	183
	ALT	Alternate output instruction	3	Zero, carry, and borrow	√	√	√	√	√	184
Peripheral instructions	FROM	Instruction for reading words from a special module buffer register	9				√	√	√	141
	DFROM	Instruction for reading double words from a special module buffer register	10				√	√	√	141
	TO	Instruction for writing words from a special module buffer register	9				√	√	√	142
	DTO	Instruction for writing double words from a special module buffer register	10				√	√	√	143
	VRRD	Instruction for reading the value of an analog potentiometer	5							143
	REFF	Instruction for setting input filtering constant	3		√	√	√	√	√	144
	REF	Instruction for immediately refreshing I/O	5		√	√	√	√	√	144
	EROMWR	EEPROM write instruction	7							145
	PR	Printing instruction	5				√	√	√	145
	TKY	Numeric key input instruction	7				√	√	√	146
Positioning instructions	ZRN	Zero return instruction	11	Zero, carry, and borrow	√	√	√	√	√	227
	PLSV	Variable speed pulse output instruction	8	Zero, carry, and borrow	√	√	√	√	√	228
	DRVI	Relative position control instruction	11	Zero, carry, and borrow	√	√	√	√	√	229
	DRVA	Absolute position control instruction	11	Zero, carry, and borrow	√	√	√	√	√	230
	DSZR	Instruction for zero return with DOG	9	Zero, carry, and borrow		√	√	√	√	231
	DVIT	Interrupt positioning instruction	11	Zero, carry, and borrow		√				232
	LIN	Linear trajectory interpolation instruction	12	Zero, carry, and borrow				√	√	234
	CW	Clockwise arc trajectory interpolation	12	Zero, carry, and borrow				√	√	237
	CCW	Counterclockwise arc trajectory interpolation instruction	12	Zero, carry, and borrow				√	√	238

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	MOVLINK	Synchronous control instruction	17	Zero, carry, and borrow						238
Real-time clock instructions	TRD	Real-time clock read instruction	3			√	√	√	√	148
	TWR	Real-time clock write instruction	3			√	√	√	√	149
	TADD	Clock addition instruction	7	Zero and carry		√	√	√	√	150
	TSUB	Clock subtraction instruction	7	Zero and borrow		√	√	√	√	151
	HOUR	Chronograph instruction	8			√	√	√	√	152
	HTOS	Instruction for converting hour-minute-second data to seconds	5				√	√	√	155
	STOH	Instruction for converting seconds to hour-minute-second data	5				√	√	√	155
Comparison contact instructions	LD=	Integer comparison LD= instruction	5		√	√	√	√	√	199
	LDD=	Long integer comparison LD= instruction	7		√	√	√	√	√	202
	LDR=	Floating-point number comparison LD= instruction	7		√	√	√	√	√	205
	LD>	Integer comparison LD= instruction	5		√	√	√	√	√	199
	LDD>	Long integer comparison LD> instruction	7		√	√	√	√	√	202
	LDR>	Floating-point number comparison LD> instruction	7		√	√	√	√	√	205
	LD>=	Integer comparison LD>= instruction	5		√	√	√	√	√	199
	LDD>=	Long integer comparison LD>= instruction	7		√	√	√	√	√	202
Comparison contact instructions	LD<	Integer comparison LD<instruction	5		√	√	√	√	√	199
	LDD<	Long integer comparison LD< instruction	7		√	√	√	√	√	202
	LDR<	Floating-point number comparison LD< instruction	7		√	√	√	√	√	205
	LD<=	Integer comparison LD<= instruction	5		√	√	√	√	√	199
	LDD<=	Long integer comparison LD<= instruction	7		√	√	√	√	√	202
	LDR<=	Floating-point number comparison LD<= instruction	7		√	√	√	√	√	205
	LD<>	Integer comparison LD<> instruction	5		√	√	√	√	√	199

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	LDD<>	Long integer comparison LD<> instruction	7		√	√	√	√	√	202
	LDR<>	Floating-point number comparison LD<> instruction	7		√	√	√	√	√	205
	AND=	Integer comparison AND= instruction	5		√	√	√	√	√	200
	ANDD=	Long integer comparison AND= instruction	7		√	√	√	√	√	203
	ANDR=	Floating-point number comparison AND= instruction	7		√	√	√	√	√	206
	AND>	Integer comparison AND> instruction	5		√	√	√	√	√	200
	ANDD>	Long integer comparison AND> instruction	7		√	√	√	√	√	203
	ANDR>	Floating-point number comparison AND> instruction	7		√	√	√	√	√	206
	AND>=	Integer comparison AND>= instruction	5		√	√	√	√	√	200
	ANDD>=	Long integer comparison AND>= instruction	7		√	√	√	√	√	203
	ANDR>=	Floating-point number comparison AND>= instruction	7		√	√	√	√	√	206
	AND<	Integer comparison AND< instruction	5		√	√	√	√	√	200
	ANDD<	Long integer comparison AND< instruction	7		√	√	√	√	√	203
	ANDR<	Floating-point number comparison AND> instruction	7		√	√	√	√	√	206
	AND<=	Integer comparison AND<= instruction	5		√	√	√	√	√	200
	ANDD<=	Long integer comparison AND<= instruction	7		√	√	√	√	√	203
	ANDR<=	Floating-point number comparison AND<= instruction	7		√	√	√	√	√	206

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
Comparison contact instructions	AND<>	Integer comparison AND<> instruction	5		√	√	√	√	√	200
	ANDD<>	Long integer comparison AND<> instruction	7		√	√	√	√	√	203
	ANDR<>	Floating-point number comparison AND<> instruction	7		√	√	√	√	√	206
	OR=	Integer comparison OR= instruction	5		√	√	√	√	√	201
	ORD=	Long integer comparison OR= instruction	7		√	√	√	√	√	204
	ORR=	Floating-point number comparison OR= instruction	7		√	√	√	√	√	207
	OR>	Integer comparison OR> instruction	5		√	√	√	√	√	201
	ORD>	Long integer comparison OR> instruction	7		√	√	√	√	√	204
	ORR>	Floating-point number comparison OR> instruction	7		√	√	√	√	√	207
	OR>=	Integer comparison OR>= instruction	5		√	√	√	√	√	201
	ORD>=	Long integer comparison OR>= instruction	7		√	√	√	√	√	204
	ORR>=	Floating-point number comparison OR>= instruction	7		√	√	√	√	√	207
	OR<	Integer comparison OR< instruction	5		√	√	√	√	√	201
	ORD<	Long integer comparison OR< instruction	7		√	√	√	√	√	204
	ORR<	Floating-point number comparison OR> instruction	7		√	√	√	√	√	207
	OR<=	Integer comparison OR<= instruction	5		√	√	√	√	√	201
	ORD<=	Long integer comparison OR<= instruction	7		√	√	√	√	√	204
	ORR<=	Floating-point number comparison OR<= instruction	7				√	√	√	207
	OR<>	Integer comparison OR<> instruction	5		√	√	√	√	√	201
	ORD<>	Long integer comparison OR<> instruction	7		√	√	√	√	√	204
	ORR<>	Floating-point number comparison OR<> instruction	7		√	√	√	√	√	207
	CMP	Instruction for setting integer comparison to ON	7				√	√	√	209

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	LCMP	Instruction for setting long integer comparison to ON	9				√	√	√	209
	RCMP	Instruction for setting floating-point number comparison to ON	9				√	√	√	210
Value conversion instructions	ITD	Instruction for converting an integer to a long integer	6	Zero, carry, and borrow	√	√	√	√	√	117
	DTI	Instruction for converting a long integer to an integer	6	Zero, carry, and borrow	√	√	√	√	√	117
	FLT	Instruction for converting an integer to a floating-point number	6	Zero, carry, and borrow	√	√	√	√	√	118
	DFLT	Instruction for converting a long integer to a floating-point number	7	Zero, carry, and borrow	√	√	√	√	√	118
	INT	Instruction for converting a floating-point number to an integer	6	Zero, carry, and borrow	√	√	√	√	√	118
	DINT	Instruction for convert a floating-point number to a long integer	7	Zero, carry, and borrow	√	√	√	√	√	119
	BCD	Instruction for converting a word to a 16-bit BCD code	5	Zero, carry, and borrow	√	√	√	√	√	119
	DBCD	Instruction for converting a double word to a 32-bit BCD code	7	Zero, carry, and borrow	√	√	√	√	√	120
	BIN	Instruction for converting a 16-bit BCD code to a word	5	Zero, carry, and borrow	√	√	√	√	√	120
	DBIN	Instruction for converting a 32-bit BCD code to a double word	7	Zero, carry, and borrow	√	√	√	√	√	121
	GRY	Instruction for converting a word to a 16-bit gray code	5	Zero, carry, and borrow	√	√	√	√	√	121
	DGRY	Instruction for converting a double word to a 32-bit gray code	7	Zero, carry, and borrow	√	√	√	√	√	121
	GBIN	Instruction for converting a 16-bit gray code to a word	5	Zero, carry, and borrow	√	√	√	√	√	122
	DGBIN	Instruction for converting a 32-bit gray code to a double word	7	Zero, carry, and borrow	√	√	√	√	√	122
	SEG	Instruction for converting a word to a 7-segment code	5	Zero, carry, and borrow	√	√	√	√	√	123

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	ASC	ASCII code conversion instruction	19	Zero, carry, and borrow	√	√	√	√	√	123
	ITA	Instruction for converting a 16-bit hex data to an ASCII code	7	Zero, carry, and borrow	√	√	√	√	√	124
	ATI	Instruction for converting an ASCII code to a 16-bit hex data	7	Zero, carry, and borrow	√	√	√	√	√	124
	LCNV	Project conversion instruction	9	Zero, carry, and borrow			√	√	√	125
	RLCNV	Floating-point project conversion instruction	12	Zero, carry, and borrow			√	√	√	126
Word contact instructions	BLD	Word bit contact LD instruction	5		√	√	√	√	√	195
	BLDI	Word bit contact LDI instruction	5		√	√	√	√	√	196
	BAND	Word bit contact AND instruction	5		√	√	√	√	√	196
	BANI	Word bit contact ANI instruction	5		√	√	√	√	√	197
	BOR	Word bit contact OR instruction	5		√	√	√	√	√	197
	BORI	Word bit contact ORI instruction	5		√	√	√	√	√	198
	BSET	Word bit coil set instruction	5		√	√	√	√	√	198
	BRST	Word bit coil reset instruction	5		√	√	√	√	√	199
	BOUT	Word bit coil output instruction	5		√	√	√	√	√	198
Communication instructions	Modbus	Master station communication instruction	8		√	√	√	√	√	184
	XMT	Free-port sending instruction	7		√	√	√	√	√	185
	RCV	Free-port receiving instruction	7		√	√	√	√	√	186
	MODRW	Modbus read/write instruction	14		√	√	√	√	√	187
Check instructions	CCITT	CCITT check instruction	7		√	√	√	√	√	191
	CRC16	CRC16 check instruction	7		√	√	√	√	√	191
	LRC	LRC check instruction	7		√	√	√	√	√	192
Data comparison instructions	DCMP=	Date = comparison instruction	7			√	√	√	√	153
	DCMP>	Date > comparison instruction	7			√	√	√	√	153
	DCMP<	Date < comparison instruction	7			√	√	√	√	153
	DCMP>=	Date >= comparison instruction	7			√	√	√	√	153
	DCMP<=	Date <= comparison instruction	7			√	√	√	√	153

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	DCMP<>	Date <> comparison instruction	7			√	√	√	√	153
Time comparison instructions	TCMP=	Time = comparison instruction	7			√	√	√	√	154
	TCMP>	Time > comparison instruction	7			√	√	√	√	154
	TCMP<	Time < comparison instruction	7			√	√	√	√	154
	TCMP>=	Time >= comparison instruction	7			√	√	√	√	154
	TCMP<=	Time <= comparison instruction	7			√	√	√	√	154
	TCMP<>	Time <> comparison instruction	7			√	√	√	√	154
Data processing instructions	MEAN	Mean instruction	7	Zero, carry, and borrow				√	√	244
	WTOB	Byte-unit data separation instruction	7	Zero, carry, and borrow				√	√	244
	BTOW	Byte-unit data combination instruction	7	Zero, carry, and borrow				√	√	245
	UNI	Instruction for combining 4 bits of 16-bit data	7	Zero, carry, and borrow				√	√	246
	DIS	Instruction for separating 4 bitsof 16-bit data	7	Zero, carry, and borrow				√	√	247
	ANS	Signal alarm set instruction	7	Zero, carry, and borrow				√	√	248
	ANR	Signal alarm reset instruction	1	Zero, carry, and borrow				√	√	249
Bulk data processing instructions	BKADD	Bulk data addition operation instruction	9	Zero, carry, and borrow				√	√	210
	BKSUB	Bulk data subtraction operation instruction	9	Zero, carry, and borrow				√	√	211
	BKCMP=,>,<,<>,<=,>=	Bulk data comparison instruction	9	Zero, carry, and borrow				√	√	211
Datasheet instructions	LIMIT	Upper/lower limit control instruction	9	Zero, carry, and borrow				√	√	212
	DBAND	Deadband control instruction	9	Zero, carry, and borrow				√	√	213
	ZONE	Zone control instruction	9	Zero, carry, and borrow				√	√	213
	SCL	Coordinate setting instruction	7	Zero, carry, and borrow				√	√	214
	SER	Data search instruction	9	Zero, carry, and borrow				√	√	215
Character string processing instructions	STRADD	String combination instruction	7	Zero, carry, and borrow				√	√	216
	STRLEN	Instruction for detecting the string length	5	Zero, carry, and borrow				√	√	217
	STRRIGHT	Instruction for reading a string from right	7	Zero, carry, and borrow				√	√	217
	STRLEFT	Instruction for reading a string from left	7	Zero, carry, and borrow				√	√	218

Instruction		Instruction function	Step length	Influenced flag bit	VC1S	VC1	VC2	VC3	VC5	Page
	STRMIDR	Instruction for reading any characters of a string	7	Zero, carry, and borrow				√	√	219
	STRMIDW	Instruction for replacing any characters of a string	7	Zero, carry, and borrow				√	√	219
	STRINSTR	String search instruction	9	Zero, carry, and borrow				√	√	220
	STRMOV	String transmission instruction	5	Zero, carry, and borrow				√	√	221
Extension file register instruction	LOADR	Instruction for reading data from an extension file register	5	Zero, carry, and borrow				√	√	221
	SAVER	Instruction for writing data to an extension file register	7	Zero, carry, and borrow				√	√	222
	INITR	Instruction for initializing an extension register	5	Zero, carry, and borrow				√	√	223
	LOGR	Instruction for logging on an extension register	11	Zero, carry, and borrow				√	√	224
	INITER	Instruction for initializing an extension file register	5	Zero, carry, and borrow				√	√	225
Other instruction	RND	Instruction for generating random numbers	3	Zero				√	√	249
	DUTY	Instruction for generating timed pulses	7					√	√	250