Smarter. Greener. Together.

Industrial Automation Headquarters

Delta Electronics, Inc.
Taoyuan Technology Center
No.18, Xinglong Rd., Taoyuan City,
Taoyuan County 33068, Taiwan
TEL: 886-3-362-6301 / FAX: 886-3-371-6301

Asia

Delta Electronics (Jiangsu) Ltd.
Wujiang Plant 3
1688 Jiangxing East Road,
Wujiang Economic Development Zone
Wujiang City, Jiang Su Province, P.R.C. 215200
TEL: 86-512-6340-3008 / FAX: 86-769-6340-7290

Delta Greentech (China) Co., Ltd.
238 Min-Xia Road, Pudong District,
ShangHai, P.R.C. 201209
TEL: 86-21-58635678 / FAX: 86-21-58630003

Delta Electronics (Japan), Inc.
Tokyo Office
2-1-14 Minato-ku Shibadaimon,
Tokyo 105-0012, Japan
TEL: 81-3-5733-1111 / FAX: 81-3-5733-1211

Delta Electronics (Korea), Inc.
1511, Byucksan Digital Valley 6-cha, Gasan-dong,
Geumcheon-gu, Seoul, Korea, 153-704
TEL: 82-2-515-5303 / FAX: 82-2-515-5302

Delta Electronics Int'l (S) Pte Ltd.
4 Kaki Bukit Ave 1, #05-05, Singapore 417939
TEL: 65-6747-5155 / FAX: 65-6744-9228

Delta Electronics (India) Pvt. Ltd.
Plot No 43 Sector 35, HSIIDC
Gurgaon, PIN 122001, Haryana, India
TEL : 91-124-4874900 / FAX : 91-124-4874945

Americas

Delta Products Corporation (USA)
Raleigh Office
P.O. Box 12173,5101 Davis Drive,
Research Triangle Park, NC 27709, U.S.A.
TEL: 1-919-767-3800 / FAX: 1-919-767-8080

Delta Greentech (Brasil) S.A.
Sao Paulo Office
Rua Itapeva, 26 - 3° andar Edificio Itapeva One-Bela Vista
01332-000-São Paulo-SP-Brazil
TEL: 55 11 3568-3855 / FAX: 55 11 3568-3865

Europe

Delta Electronics (Netherlands) B.V.
Eindhoven Office
De Witbogt 20, 5652 AG Eindhoven, The Netherlands
TEL : +31 (0)40-8003800 / FAX : +31 (0)40-8003898

AS-0249720-01

*We reserve the right to change the information in this manual without prior notice.

2016/11/30

AS **Series Programming Manual**

AS **Series Programming Manual**

www.deltaww.com

Smarter. Greener. Together.

AS Series Programming Manual

Revision History

Version	Revision	Date
1st	The first version was published.	2016/11/30

AS Series Programming Manual

Table of Contents

Chapter 1 Introduction

1.1 Overview 1-2
1.1.1 Related Manuals 1-2
1.1.2 Model Description 1-2

1.2 Software 1-6
1.2.1 Program Editor 1-6
1.2.2 Program Organization Units and Tasks 1-8

Chapter 2 Devices

2.1 Introduction of Devices 2-2
2.1.1 Device Table 2-2
2.1.2 Basic Structure of I/O Storages 2-3
2.1.3 Relation Between the PLC Action and the Device Type 2-3
2.1.4 Latched Areas in the Device Range 2-4

2.2. Functions of Devices 2-5
2.2.1 Values and Constants 2-5
2.2.2 Floating-point Numbers 2-7
2.2.2.1 Single-precision Floating-point Numbers 2-7
2.2.2.2 Decimal Floating-point Numbers 2-8
2.2.3 Strings 2-8
2.2.4 Input Relays (X) 2-9
2.2.5 Output Relays (Y) 2-10
2.2.6 Auxiliary Relays (M) 2-10
2.2.7 Special Auxiliary Relays (SM) 2-11
2.2.8 Refresh Time of Special Auxiliary Relays 2-43
2.2.9 Stepping Relays (S) 2-50
2.2.10 Timers (T) 2-50
2.2.11 Counters 2-52
2.2.12 32-bit Counters (HC) 2-54
2.2.13 Data Registers (D) 2-56
2.2.14 Special Data Registers (SR) 2-56
2.2.15 Refresh Time of Special Data Registers 2-83
2.2.16 Additional Remarks on Special Auxiliary Relays and Special Data Registers 2-86
2.2.17 Index Register (E) 2-97
2.2.18 File Registers (FR) 2-97

Chapter 3 Instruction Tables

3.1 Instructions 3-2
3.1.1 Basic Instructions 3-2
3.1.2 Applied Instructions 3-2

3.2 Instruction Tables 3-3
3.2.1 Basic Instructions 3-3
3.2.2 Applied Instructions (Sorted numerically) 3-4
3.2.3 Applied Instructions (Sorted Alphabetically) 3-5
3.2.4 Device Tables 3-6

3.3 Lists of Basic Instructions 3-7

3.4 Lists of Applied Instructions 3-9
3.4.1 Applied Instructions (Sorted numerically) 3-9
3.4.2 Applied Instructions (Sorted Alphabetically) 3-37

Chapter 4 Instruction Structure

4.1 API Composition of Applied Instructions 4-2

4.2 Operand Usage Description 4-5

4.3 Restrictions on the Use of the Instructions 4-6

4.4 Index Registers 4-7

4.5 Pointer Registers 4-9

4.6 Pointer Registers of Timers 4-11

4.7 Pointer Registers of 16-bit Counters 4-12

4.8 Pointer Registers of 32-bit Counters 4-14

4.9 File Register 4-15

Chapter 5 Basic Instructions

5.1 List of Basic Instructions 5-2

5.2 Basic Instructions 5-3

Chapter 6 Applied Instructions

6.1 Comparison Instructions 6-4

6.1.1 List of Comparison Instructions 6-4
6.1.2 Explanation of Comparison Instructions 6-7

6.2 Arithmetic Instructions 6-46
6.2.1 List of Arithmetic Instructions 6-46
6.2.2 Explanation of Arithmetic Instructions 6-47

6.3 Data Conversion Instructions 6-78
6.3.1 List of Data Conversion Instructions 6-78
6.3.2 Explanation of Data Conversion Instructions 6-79

6.4 Data Transfer Instructions 6-117
6.4.1 List of Data Transfer Instructions 6-117
6.4.2 Explanation of Data Transfer Instructions 6-118

6.5 Jump Instructions 6-145
6.5.1 List of Jump Instructions 6-145
6.5.2 Explanation of Jump Instructions 6-146

6.6 Program Execution Instructions 6-154
6.6.1 List of Program Execution Instructions 6-154
6.6.2 Explanation of Program Execution Instructions 6-155

6.7 IO Refreshing Instructions 6-167
6.7.1 List of IO Refreshing Instructions 6-167
6.7.2 Explanation of IO Refreshing Instructions 6-168

6.8 Convenience Instructions 6-170
6.8.1 List of Convenience Instructions 6-170
6.8.2 Explanation of Convenience Instructions 6-171

6.9 Logic Instructions 6-218
6.9.1 List of Logic Instructions 6-218
6.9.2 Explanation of Logic Instructions 6-219

6.10 Rotation Instructions 6-240
6.10.1 List of Rotation Instructions 6-240
6.10.2 Explanation of Rotation Instructions 6-241

6.11 Timer and Counter Instructions 6-252
6.11.1 List of Timer and Counter Instructions 6-252
6.11.2 Explanation of Timer and Counter Instructions 6-253

6.12 Shift Instructions 6-285
6.12.1 List of Shift Instructions 6-285
6.12.2 Explanation of Shift Instructions 6-286

6.13 Data Processing Instructions 6-323
6.13.1 List of Data Processing Instructions 6-323

6.13.2 Explanation of Data Processing Instructions 6-324

6.14 Structure Creation Instructions 6-377
6.14.1 List of Structure Creation Instructions 6-377
6.14.2 Explanation of Structure Creation Instructions 6-378

6.15 Module Instructions 6-386
6.15.1 List of Module Instructions 6-386
6.15.2 Explanation of Module Instructions 6-387

6.16 Floating-point Number Instructions 6-393
6.16.1 List of Floating-point Number Instructions 6-393
6.16.2 Explanation of Floating-point Number Instructions 6-394

6.17 Real-time Clock Instructions 6-429
6.17.1 List of Real-time Clock Instructions 6-429
6.17.2 Explanation of Real-time Clock Instructions 6-430

6.18 Peripheral Instructions 6-453
6.18.1 List of Peripheral Instructions 6-453
6.18.2 Explanation of Peripheral Instructions 6-454

6.19 Communication Instructions 6-470
6.19.1 List of Communication Instructions 6-470
6.19.2 Explanation of Communication Instructions 6-471
6.19.3 Descriptions on the Communication-related Flags and Registers 6-539

6.20 Other Instructions 6-542
6.20.1 List of Other Instructions 6-542
6.20.2 Explanation of Other Instructions 6-543

6.21 String Processing Instructions 6-553
6.21.1 List of String Processing Instructions 6-553
6.21.2 Explanation of String Processing Instructions 6-554

6.22 Ethernet Instructions 6-608
6.22.1 List of Ethernet Instructions 6-608
6.22.2 Explanation of Ethernet Instructions 6-609

6.23 Memory Card Instructions 6-643
6.23.1 List of Memory Card Instructions 6-643
6.23.2 Explanation of Memory Card Instructions 6-644

6.24 Task Control Instructions 6-660
6.24.1 List of Task Control Instructions 6-660
6.24.2 Explanation of Task Control Instructions 6-661

6.25 SFC Instructions 6-665
6.25.1 List of SFC Instructions 6-665

6.25.2 Explanation of SFC Instructions 6-666

6.26 High-speed Output Instructions 6-673
6.26.1 List of High-speed Output Instructions..... 6-673
6.26.2 Explanation of High-speed Output Instructions 6-675

6.27 Delta CANopen Communication Instructions 6-760
6.27.1 List of Delta CANopen Communication Instructions..... 6-760
6.27.2 Explanation of Delta CANopen Communication Instructions 6-761

Chapter 7 Troubleshooting

7.1 Troubleshooting 7-2
7.1.1 Basic troubleshooting steps 7-2
7.1.2 Clear the States of Errors 7-2
7.1.3 Troubleshooting SOP 7-3
7.1.4 System Log 7-4

7.2 Troubleshooting for CPU Modules 7-5
7.2.1 ERROR LED Indicator's Being ON 7-5
7.2.2 ERROR LED Indicator's Blinking Every 0.5 Seconds 7-5
7.2.3 ERROR LED Indicator's Raipid Blinking Every 0.2 Seconds..... 7-7
7.2.4 ERROR LED Indicator's Slow Blinking Every 3 Seconds and Lighting up for 1 Second..... 7-7
7.2.5 BAT. LOW LED Indicator's Being ON..... 7-7
7.2.6 BAT. LOW LED Indicator's Blinking Every 0.5 Seconds 7-7
7.2.7 The LED Indicators of RUN and ERROR are Blinking Every 0.5 Seconds Simultaneously..... 7-7
7.2.8 The LED Indicators of RUN and ERROR are Blinking One After Another Every 0.5 Seconds. 7-8
7.2.9 Other Errors (Without LED Indicators)..... 7-8

7.3 Troubleshooting for I/O Modules 7-14
7.3.1 Troubleshootings for Analog Modules (AD/DA/XA) and Temperature Modules (RTD/TC) 7-14
7.3.2 Troubleshootings for Load Cell Module AS02LC..... 7-15
7.3.3 Troubleshootings for Module AS00SCM as a Communication Module ... 7-15
7.3.4 Troubleshootings for Module AS00SCM as a Remote Module..... 7-17

7.4 Error Codes and LED Indicators for CPU Modules 7-18
7.4.1 Error Codes and LED Indicators for CPU Modules 7-18
7.4.2 Error Codes and LED Indicators for Analog/Temperature Modules..... 7-23
7.4.3 Error Codes and LED Indicators for Load Cell Module AS02LC 7-24

7.4.4 Error Codes and LED Indicators for Module AS00SCM as a Communication Module 7-25
7.4.5 Error Codes and LED Indicators for Module AS00SCM as a Remote Module 7-25

1

Chapter 1 Introduction

Table of Contents

1.1 Overview 1-2
1.1.1 Related Manuals 1-2
1.1.2 Model Description 1-2

1.2 Software 1-6
1.2.1 Program Editor 1-6
1.2.2 Program Organization Units and Tasks 1-8

1.1 Overview

This manual introduces the programming of the AS series programmable logic controllers, the basic instructions, and the applied instructions.

1.1.1 Related Manuals

The related manuals of the AS series programmable logic controllers are composed of the following.

- AS series Quick Start

  It guides users to use the system before they read the related manuals.

- AS series Programming Manual

  It introduces the programming of the AS series programmable logic controllers, the basic instructions, and the applied instructions.

- ISPSoft User Manual

  It introduces the use of ISPSoft, the programming languages (ladder diagrams, instruction lists, sequential function charts, function block diagrams, and structured texts), the concept of POUs, and the concept of tasks.

- AS series Hardware Manual

  It introduces electrical specifications, appearances, dimensions, and etc.

- AS series Operation Manual

  It introduces functions of CPUs, devices, module tables, troubleshooting, and etc.

- AS series Module Manual

  It introduces the use of special I/O modules. For example, network modules, analog I/O modules, temperature measurement modules, and etc.

1.1.2 Model Description

Classification	Model Name	Description
Power supply module	AS-PS02	Input: 100~240 VAC, 50/60 Hz Output: 24VDC/2A, 48W (for PLC internal use)
	AS-PS02A	Input: 100~240 VAC, 50/60 Hz Output: 24VDC/1.5A, 36W (for PLC internal use) Output: 24VDC/0.5A, 12W (for external use)
CPU module	AS332P-A	CPU module, PNP output, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, 2x function cards (optional), supporting 32 I/Os (16DI+16DO) and up to 1024 I/Os, the program capacity:128K steps
	AS332T-A	CPU module, NPN output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, 2x function cards (optional), supporting 32 I/Os (16DI+16DO) and up to 1024 I/Os, the program capacity:128K steps
	AS324MT-A	CPU module, NPN differential output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, 2x function cards (optional), supporting 24 I/Os (12DI+12DO) and up to 1016 I/Os, the program capacity:128K steps
Digital input/output module	AS08AM10N-A	24VDC 5mA 8 inputs Spring-clamp terminal block

Classification	Model Name	Description
	AS08AN01P-A	5 ~ 30VDC 0.5A 8 outputs Sourcing output Spring-clamp terminal block
	AS08AN01R-A	240VAC/24VDC 2A 8 outputs Relay Spring-clamp terminal block
	AS08AN01T-A	5 ~ 30VDC 0.5A 8 outputs Sinking output Spring-clamp terminal block
	AS16AM10N-A	24VDC 5mA 16 inputs Spring-clamp terminal block
	AS16AN01P-A	5 ~ 30VDC 0.5A 16 outputs Sourcing output Spring-clamp terminal block
	AS16AN01R-A	240VAC/24VDC 2A 16 outputs Relay Spring-clamp terminal block
	AS16AN01T-A	5 ~ 30VDC 0.5A 16 outputs Sinking output Spring-clamp terminal block
	AS16AP11P-A	24VDC 5mA 8 inputs 5 ~ 30VDC 0.5A 8 outputs Sourcing output Spring-clamp terminal block
	AS16AP11R-A	24VDC 5mA 8 inputs 240VAC/24VDC 2A 8 outputs Relay

Classification	Model Name	Description
		Spring-clamp terminal block
	AS16AP11T-A	24VDC 5mA 8 inputs 5 ~ 30VDC 0.5A 8 outputs Sinking output Spring-clamp terminal block
	AS32AM10N-A	24VDC 3.2mA 32 inputs MIL connector
	AS32AN02T-A	5 ~ 30VDC 0.1A 32 outputs Sinking output MIL connector
	AS64AM10N-A	24VDC 3.2mA 64 inputs MIL connector
	AS64AN02T-A	5 ~ 30VDC 0.1A 64 outputs Sinking output MIL connector
Analog input/output module	AS04AD-A	4-channel analog input module Hardware resolution: 16 bits 0~10V, 0/1~5V, -5~+5V, -10~+10V, 0/4~20mA, -20~+20mA Conversion time: 2ms/channel
	AS04DA-A	4-channel analog input module Hardware resolution: 12 bits -10~+10V, 0~20mA, 4~20mA Conversion time: 2ms/channel
	AS06XA-A	4-channel analog input module Hardware resolution: 16 bits 0~10V, 0/1~5V, -5~+5V, -10~+10V, 0/4~20mA, -20~+20mA Conversion time: 2 ms/channel 2-channel analog input module Hardware resolution: 12 bits -10~+10V, 0~20mA, 4~20mA Conversion time: 2ms/channel
Temperature measurement module	AS04RTD-A	4-channe, 2-wire/3-wire RTD Sensor type: Pt100 / Ni100 / Pt1000 / Ni1000 / JPt100 / LG-Ni1000 / Cu50 / Cu100 / 0~300Ω / 0~3000Ω input impedance Resolution: 0.1°C/0.1°F (16 bits) Conversion time: 200ms/channel
	AS04TC-A	4-channe thermocouple

Classification	Model Name	Description
		Sensor type: J, K, R, S, T, E, N, B and -100~+100 mV Resolution: 0.1°C/0.1°F (24 bits) Conversion time: 200ms/channel
Load cell module	AS02LC-A	2-channel, 4-wire/6-wire load cell sensor Eigenvalue applicable to a load cell: 1, 2, 4, 6, 20, 40, 80 mV/V Highest precision 1/10000 @ 50ms of the conversion time ADC Resolution : 24 bits Conversion time: 2.5 ~ 400ms (9 options to choose from)
Network module	AS00SCM-A	Serial communication module, 2x communication ports, applicable to communication cards, supporting MODBUS protocols
Remote I/O module	AS00SCM-A + AS-FCOPM	Applicable to AS-FCOPM function cards
Function cards	AS-F232	Serial communication port, RS232, functioning as a master or slave
	AS-F422	Serial communication port, RS422, functioning as a master or slave
	AS-F485	Serial communication port, RS485, functioning as a master or slave
	AS-FCOPM	CANopen communication port, supporting DS301, AS series remote modules and Delta servo systems
	AS-F2AD	2-channel analog input 0~10V (12 bits), 4~20mA (11 bits) Conversion time: 3ms/channel
	AS-F2DA	2-channel analog input 0~10V, 4~20mA (12 bits) Conversion time: 2ms/channel
Programming cable	UC-PRG015-01A (1.5M)	Used for the connection between a PLC and a PC via a mini USB port, applicable for AS332T-A, AS332P-A, and AS324MT-A
	UC-PRG030-01A (3M)	Used for the connection between a PLC and a PC via a mini USB port, applicable for AS332T-A, AS332P-A, AS324MT-A
	UC-PRG030-20A (3M)	Used for the connection between a PLC and a PC via a RJ45 port, applicable for AS332T-A, AS332P-A, AS324MT-A
I/O extension cable	UC-ET010-24B (1M) UC-ET020-24B (2M) UC-ET030-24B (3M)	MIL connector, 40Pin ↔ 40Pin, shielded, applicable for AS32AM10N-A, AS32AN02T-A, AS64AM10N-A, AS64AN02T-A
	UC-ET010-24D (1M) UC-ET020-24D (2M) UC-ET030-24D (3M)	MIL connector, 40Pin↔ 2x 20Pin, shielded, applicable for AS332T-A, AS332P-A, AS324MT-A, AS32AM10N-A, AS32AN02T-A, AS64AM10N-A, AS64AN02T-A
External terminal module	UB-10-ID16A	16 inputs/outputs, 20-Pin MIL connector, applicable for AS332T-A, AS332P-A, AS324MT-A, AS32AM10N-A, AS32AN02T-A, AS64AM10N-A, AS64AN02T-A
	UB-10-ID32A	32 inputs, 40-Pin MIL connector, applicable for AS32AM10N-A, AS64AM10N-A
	UB-10-OR16A	16 relay outputs, 20-Pin MIL connector, NPN, applicable for AS332T-A, AS32AN02T-A, AS64AN02T-A
	UB-10-OR16B	16 relay outputs, 20-Pin MIL connector, PNP, applicable for AS332P-A
	UB-10-OT32A	32 transistor outputs, 40-Pin MIL connector, NPN, applicable for AS32AN02T-A, AS64AN02T-A
Network cables	UC-CMC003-01A (0.3M)	CANopen communication cable, applicable for AS-FCOPM series
	UC-CMC005-01A (0.5M)	CANopen communication cable, applicable for AS-FCOPM series

Classification	Model Name	Description
	UC-CMC010-01A (1M)	CANopen communication cable, applicable for AS-FCOPM series
	UC-CMC015-01A (1.5M)	CANopen communication cable, applicable for AS-FCOPM series
	UC-CMC020-01A (2M)	CANopen communication cable, applicable for AS-FCOPM series
	UC-CMC030-01A (3M)	CANopen communication cable, applicable for AS-FCOPM series
	UC-CMC050-01A (5M)	CANopen communication cable, applicable for AS-FCOPM series
	UC-CMC100-01A (10M)	CANopen communication cable, applicable for AS-FCOPM series
	UC-CMC200-01A (20M)	CANopen communication cable, applicable for AS-FCOPM series

1.2 Software

1.2.1 Program Editor

The outline of program editor ISPSoft:

- There are four types of programming languages, including the structure text (ST), the ladder diagram (LD), the sequential function chart (SFC), and the Continuous Function Chart (CFC).
  NOTE: the CFC programming is only available for ISPSoft with version 3.01 or higher.

- The use of variables which allows the user to define the variable symbol to replace the device name of the PLC not only enhances the readability of the program, but also saves the user a lot of time to allocate the address of the device.

- The introduction of the POU (Program Organization Unit) framework not only divides the main program into several program units, but also replaces the traditional subroutines with functions and function blocks. The framework of the program becomes more modular, and is easier to be managed.

- The concept of tasks which is used to manage the execution order of the programs advances the program development to the level of project management. The large-scale program development becomes easier to be managed.

1.2.2 Program Organization Units and Tasks

The POUs (Program Organization Units) are the basic elements which constitute the PLC program. Differing from the traditional PLC program, the character of the program framework introduced by IEC 61131-3 lies in the fact that the large program is divided into several small units. These small units are called POUs. The POUs can be classified into three types.

1. Program (PROG): The POU of the program type plays the role of the primary program in the PLC program. The designer can define the execution of the POU of the program type as the cyclic scan or the interrupt, and arrange the scan order in the task list for the POUs of the program type.
2. Function block (FB): The meaning of the function block (FB) in itself is similar to the subroutine. The program defined within the function block is executed after the function block is called by the POU of the program type and the related parameters are entered.
3. Function (FC): The meaning of the function (FC) in itself is close to the macro instruction. That is, users can write many operation instructions or functions into the function POU, and then call them into use in the POU of the program type or the function block.

The task is a function which stipulates that programs are executed in certain order or according to certain interrupt condition. The meaning of the task lies in the fact that it provides each POU of the program type with a specific execution task, and specifies the execution order for the POUs of the program type or the way to enable them.

Basically, not all of the POUs of the program type in a project will take part in the practical execution. Whether to execute the POU of the program type or not, and how to execute it depend on the assignment of the task. If the POU of the program type is not assigned the task, it will be saved as an ordinary source code with the project instead of being compiled as an execution code of the PLC. In addition, only the POU of the program type needs to be assigned the task. The execution of the function blocks or functions depends on the superior POU of the program type which calls them. There are three types of tasks.

1. Cyclic task: The POUs of the program type assigned to the cyclic task will be scanned cyclically, and executed in order.
2. Timed interrupt task: If the time of interrupting is reached, all POUs of the program type assigned to the timed interrupt task will be executed in order.
3. Conditional interrupt task: Conditional Interrupts can be divided into several types. For example, the external interrupts, the I/O interrupts, and etc. Users have to make sure of the interrupts supported by the PLC before they create a project. If the POU of the program type is assigned to the conditional interrupt task, the function of the POU of the program type is similar to the interrupt subroutine. If the interrupt condition is satisfied, e.g. the contact of the external interrupt is triggered, all POUs of the program type assigned to the task will be executed in order.

2

Chapter 2 Devices

Table of Contents

2.1 Introduction of Devices 2-2
2.1.1 Device Table 2-2
2.1.2 Basic Structure of I/O Storages 2-3
2.1.3 Relation Between the PLC Action and the Device Type 2-3
2.1.4 Latched Areas in the Device Range 2-4

2.2. Functions of Devices 2-5
2.2.1 Values and Constants 2-5
2.2.2 Floating-point Numbers 2-7
2.2.2.1 Single-precision Floating-point Numbers 2-7
2.2.2.2 Decimal Floating-point Numbers 2-8
2.2.3 Strings 2-8
2.2.4 Input Relays (X) 2-9
2.2.5 Output Relays (Y) 2-10
2.2.6 Auxiliary Relays (M) 2-10
2.2.7 Special Auxiliary Relays (SM) 2-11
2.2.8 Refresh Time of Special Auxiliary Relays 2-43
2.2.9 Stepping Relays (S) 2-50
2.2.10 Timers (T) 2-50
2.2.11 Counters 2-52
2.2.12 32-bit Counters (HC) 2-54
2.2.13 Data Registers (D) 2-56
2.2.14 Special Data Registers (SR) 2-56
2.2.15 Refresh Time of Special Data Registers 2-83
2.2.16 Additional Remarks on Special Auxiliary Relays and Special Data Registers 2-86
2.2.17 Index Register (E) 2-97
2.2.18 File Registers (FR) 2-97

2.1 Introduction of Devices

This section gives an account of values/strings processed by the PLC. It also describes the functions of devices which include input/output/auxiliary relays, timers, counters, and data registers.

2.1.1 Device Table

Type	Device name		Number of devices	Range
Bit device	Input relay	X	1024	X0.0~X63.15
	Output relay	Y	1024	Y0.0~Y63.15
	Data register	D	48,0000	D0.0~D29999.15
		W	48,0000	W0.0~W29999.15 *4
	Auxiliary relay	M	8192	M0~M8191
	Special auxiliary relay	SM	2048	SM0~SM2047
	Stepping relay	S	2048	S0~S2047
	Timer	T	512	T0~T511
	Counter	C	512	C0~C511
	32-bit counter	HC	256	HC0~HC255
Word device	Input relay	X	64	X0~X63
	Output relay	Y	64	Y0~Y63
	Data register	D	30000	D0~D29999
		W	30000	W0~W29999 *4
	Special auxiliary relay	SR	2048	SR0~SR2047
	File register	FR	65536	FR0~FR65535
	Timer	T	512	T0~T511
	Counter	C	512	C0~C511
	32-bit counter	HC	256（512 words）	HC0~HC255
	Index register	E	10	E0~E9
			5	E10~E14 *4
Constant*1	Decimal system	K	16 bits: -32768~32767 32 bits: -2147483648~2147483647	
Constant*2	Hexadecimal system	16#	16 bits: 16#0~16#FFFF 32 bits: 16#0~16#FFFFFFFF	
	Single-precision floating-point number	F	32 bits: $\pm 1.17549435^{-38} \sim \pm 3.40282347^{+38}$	
String*3	String	“$”	1~31 characters	

*1: The decimal forms are notated by K in the device lists in Chapter 5 and Chapter 6 in AS Series Programming Manual. For example a K50 in the AS programming manual, only the number 50 should be inputted in ISPSoft.

*2: The floating-point numbers are notated by F/DF in the device lists in Chapter 5 and Chapter 6 in AS Series Programming Manual, whereas they are represented by decimal points in ISPSoft; for the floating-point F500, one should input 500.0.

*3: The strings are notated by “$” in Chapter 5 and Chapter 6 in AS Series Programming Manual, whereas they are represented by “ ” in ISPSoft; for the string of 1234, one should input “1234”in ISPSoft.

*4: Used for editing in ISPSoft only.

2.1.2 Basic Structure of I/O Storages

Device	Function	Access of bits	Access of words	Modification by ISPSoft	Forcing the bit ON/OFF
X	Input relay	OK	OK	OK	OK
Y	Output relay	OK	OK	OK	OK
M	Auxiliary relay	OK	-	OK	-
SM	Special auxiliary relay	OK	-	OK	-
S	stepping relay	OK	-	OK	-
T	Timer	OK	OK	OK	-
C	Counter	OK	OK	OK	-
HC	32-bit counter	OK	OK	OK	-
D	Data register	OK	OK	OK	OK
SR	Special data register	-	OK	OK	-
FR	File register	-	OK*1	-	-
E	Index register	-	OK	OK	-

*1: An instruction should be used for the writing of a FR.

2.1.3 Relation Between the PLC Action and the Device Type

PLC action \ Device type		Non-latched area		Latched area	
		Device Y	**Other devices**	**File register**	**Other devices**
Power: OFF→ON		Cleared	Cleared	Retained	Retained
Restore to defaults		Cleared	Cleared	Cleared	Cleared
STOP ↓ RUN*1	**The non-latched area is cleared.**	Cleared	Cleared	Retained	Retained
	The state of the non-latched area is retained.	Retained	Retained	Retained	Retained
RUN ↓ STOP*1	**The state of device Y is cleared.**	Cleared	Retained	Retained	Retained
	The state of device Y is retained.	Retained	Retained	Retained	Retained
SM204 is ON. (All non-latched areas are cleared.)		Cleared	Cleared	Retained	Retained
SM205 is ON. (All latched areas are cleared.)		Retained	Retained	Retained	Cleared

*1: For the setups of the states, please go to HWCONFIG in ISPSoft. The default of PLC STOP->RUN is “cleared not-latched area”. The default of PLC RUN->STOP is “cleared the state of device Y”.

2.1.4 Latched Areas in the Device Range

Device	Function	Device range	Latched area
X	Input relay	X0~X63	All devices are non-latched.
Y	Output relay	Y0~Y63	All devices are non-latched.
M*[1]	Auxiliary relay	M0~M8191	The default range is M6000~M8191.
SM	Special auxiliary relay	SM0~SM2047	Some devices are latched, and cannot be changed. Please refer to the list of special auxiliary relays for more information.
S*[1]	Stepping relay	S0~S1023	The default range is S512~S1023
T	Timer	T0~T511	All devices are non-latched.
C*[1]	Counter	C0~C511	The default range is C448~C511
HC*[1]	32-bit counter	HC0~HC255	The default range is HC128~HC255
D*[1]	Data register	D0~D29999	The default range is D20000~D29999
		W0~W29999	*[2]
FR	File register	FR0~FR65535	All devices are latched.
SR	Special data register	SR0~SR2047	Some are latched, and cannot be changed. Please refer to the list of special data registers for more information.
E	Index register	E0~E9	All devices are non-latched.
		E10~E14	*[2]

*1: For the setups of the latched area, please go to HWCONFIG in ISPSoft. Set the latched area and the other areas will be seen as non-latched areas. The range of latched areas cannot exceed the device range. For example, set the M600~M7000 as the latched areas and that makes M0~M5999 and M7001~M8191 as the non-latched areas.

*2: Used for editing in ISPSoft only.

2.2. Functions of Devices

Procedure for processing the program in the PLC:

- Regenerating the input signal
  1. Before the program is executed, the state of the external input signal is read into the memory of the input signal.
  2. When program is executed, the state in the memory of the input signal does not change even if the input signal changes from ON to OFF or from OFF to ON. Not until the next scan begins will the input signal be refreshed.
- Processing the program

  After the input signal is refreshed, the instructions in the program are executed in order from the start address of the program, and the results are stored in the device memories.
- Regenerating the state of the output

  After the instruction END is executed, the state in the device memory is sent to the specified output terminal.

2.2.1 Values and Constants

Name	Description
Bit	A bit is the basic unit in the binary system. Its state is either 1 or 0.
Nibble	A nibble is composed of four consecutive bits (e.g. b3~b0). Nibbles can be used to represent 0~9 in the decimal system, or 0~F in the hexadecimal system.
Byte	A byte is composed of two consecutive nibbles (i.e. 8 bits, b7~b0). Bytes can be used to represent 00~FF in the hexadecimal system.
Word	A word is composed of two consecutive bytes (i.e. 16 bits, b15~b0). Words can be used to represent 0000~FFFF in the hexadecimal system.
Double word	A double word is composed of two consecutive words (i.e. 32 bits, b31~b0). Double words can be used to represent 00000000~FFFFFFFF in the hexadecimal system.

The relation among bits, nibbles, bytes, words, and double words in the binary system is shown below.

The PLC uses four types of values to execute the operation according to different control purposes. The functions of these values are illustrated as follows:

1. Binary number (BIN)

   The PLC adopts the binary system to operate the values.

2. Decimal number (DEC)

   The decimal number in the PLC is used as;

   - the setting value of the timer (T) or the setting value of the counter (C/HC). For example, TMR C0 50 (**constant K**).
   - the device number. For example, M10 and T30 (device number)
   - the number before or after the decimal point. For example, X0.0, Y0.11, and D10.0 (device number).
   - **the constant K**: It is used as the operand in the applied instruction. For example, MOV 123 D0 **(constant K)**.

3. Binary-coded decimal (BCD)

   A decimal value is represented by a nibble or four bits, and therefore sixteen consecutive bits can represent a four-digit decimal value.

4. Hexadecimal number (HEX)

   The hexadecimal number in the PLC is used as;

   - **the constant 16#**: It is used as the operand in the applied instruction. For example, MOV 16#1A2B D0 (hexadecimal constant).

Corresponding values:

Binary Number (BIN)	**Decimal Number (DEC)**	**Binary Code Decimal (BCD)**	**Hexadecimal Number (HEX)**
PLC internal execution	**Constant** K, Device number	BCD related instruction	Constant 16#, Device number
0000	0	0000	0
0001	1	0001	1
0010	2	0010	2
0011	3	0011	3
0100	4	0100	4
0101	5	0101	5
0110	6	0110	6
0111	7	0111	7
1000	8	1000	8
1001	9	1001	9
1010	10	-	A
1011	11	-	B
1100	12	-	C
1101	13	-	D
1110	14	-	E
1111	15	-	F
10000	16	0001 0000	10
10001	17	0001 0001	11

2.2.2 Floating-point Numbers

The floating-point numbers are represented by decimal points in ISPSoft. For example, the floating-point number of 500 is 500.0. Please refer to section 2.2.2 in AS Series Programming Manual for more information.

The floating-point numbers are represented by decimal points in ISPSoft. For example, the floating-point number of 500 is 500.0.

2.2.2.1 Single-precision Floating-point Numbers

The floating-point number is represented by the 32-bit register. The representation adopts the IEEE754 standard, and the format is as follows.

Equation: $(-1)^S \times 2^{E-B} \times 1.M; B = 127$

The single-precision floating-point numbers range from ±2-126 to ±2+128, and correspond to the range from ±1.1755×10-38 to ±3.4028×10+38.

The AS series PLC uses two consecutive registers to form a 32-bit floating-point number. Take (D1, D0) for example.

Example 1:

23 is represented by the single-precision floating-point number.

Step 1: Convert 23 into the binary number, i.e. 23.0=10111.

Step 2: Normalize the binary number, i.e. 10111=1.0111 ×24 (0111 is the mantissa, and 4 is the exponent.).

Step 3: Get the value of the exponent.

∵ E-B=4→E-127=4 ∴ E=131=100000112

Step 4: Combine the sign bit, the exponent, and the mantissa to form the floating-point number.

0 100**0001**1 011**1000**0000**0000**0000**0000**$_2$=41B80000$_{16}$

Example 2:

-23 is represented by the single-precision floating-point number.

The steps of converting -23.0 into the floating-point number are the same as those of converting 23.0 into the floating-point number, except that the sign bit is 1.

1 100**0001**1 011**1000**0000**0000**0000**0000**$_2$=C1B80000$_{16}$

2.2.2.2 Decimal Floating-point Numbers

- Since single-precision floating-point numbers and double-precision floating-point numbers are not widely accepted by people, they can be converted into decimal floating-point numbers for people to make judgement. However, as to the operation of the decimal point, the PLC still uses single-precision floating-point numbers and double-precision floating-point numbers.
- A 32-bibt decimal floating-point number is represented by two consecutive registers. The constant is stored in the register whose number is smaller while the exponent is stored in the register whose number is bigger. Take (D1, D0) for example.

$$\text{Decimal floating-point number}=[\text{Constant D0}]*10^{[\text{Exponent D1}]}$$

Base number D0=±1,000~±9,999

Exponent D1=-41~+35

The base number 100 does not exist in D0 because 100 is represented by 1,000×10^{-1}. In addition, 32-bit decimal floating-point numbers range from ±1175×10^{-41} to ±402×10^{+35}.

2.2.3 Strings

What strings can process are ASCII codes (*1). A complete string begins with a start character, and ends with an ending character (NULL code). If what you enter is a string, you can enter 31 characters at most, and the ending character 16#00 will be added automatically in ISPSoft.

1. No string (NULL code) is moved.

D0=0 (NULL)

2. The string is an even number.

D0	16#62 (b)	16#61 (a)
D1	16#64 (d)	16#63 (b)
D2	0 (NULL)	

3. The string is an odd number.

D0	16#62 (b)	16#61 (a)
D1	16#64 (d)	16#63 (b)
D2	0 (NULL)	16#65 (e)

*1: ASCII code chart

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Hex	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
ASCII	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Hex	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
ASCII	SP	!	"	#	$	%	&	'	(	)	*	+	·	-	.	/
Hex	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
ASCII	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
Hex	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
ASCII	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Hex	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
ASCII	P	Q	R	S	T	U	V	W	X	Y	Z	☒	☒	☒	☒	☒
Hex	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
ASCII	`	a	b	C	d	e	f	g	h	i	j	k	l	M	n	o
Hex	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
ASCII	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	☒

Note: ☒ represents an invisible character. Please do not use it.

2.2.4 Input Relays (X)

- Function of the input

  The input is connected to the input device (e.g. external devices such as button switches, rotary switches, number switches, and etc.), and the input signal is read into the PLC. Besides, contact A or contact B of the input can be used several times in the program, and the ON/OFF state of the input varies with the ON/OFF state of the input device.

- Input number (the decimal number):

  For the PLC, the input numbers start from X0.0. The number of inputs varies with the number of inputs on the digital input/output modules, and the inputs are numbered according to the order in which the digital input/output modules are connected to the CPU module. The maximum number of inputs on the PLC can reach up to 8192, and the range

is between X0.0 and X511.15.

- Input type

  The inputs are classified into two types.

  1. Regenerated input: Before the program is executed, the data is fed into the PLC according to the states of the inputs which are regenerated. For example, LD X0.0.
  2. Direct input: During the execution of the instructions, the data is fed into the PLC according to the states of the inputs. For example, LD DX0.0.

2.2.5 Output Relays (Y)

- The function of the output

  The task of the output is sending the ON/OFF signal to drive the load connected to the output. The load can be an external signal lamp, a digital display, or an electromagnetic valve. There are four types of outputs. They are relays, transistors (NPN and PNP), and TRIACs (thyristors). Contact A or contact B of the output can be used several times in the program, but the output Y should be used only once in the program. Otherwise, according the program-scanning principle of the PLC, the state of the output depends on the circuit connected to the last output Y in the program.

- The output number (the decimal number)

  For the PLC, the input numbers start from Y0.0. The number of outputs varies with the number of outputs on the digital input/output modules, and the outputs are numbered according to the order in which the digital input/output modules are connected to the PLC. The maximum number of outputs on the PLC can reach up to 1024, and the range is between Y0.0 and Y63.15.

  The output which is not practically put to use can be used as a general device.

- The output type

  The outputs are classified into two types.

  1. Regenerated output: Not until the program executes the instruction END is the information fed out according to the states of the outputs. For example, OUT Y0.0.
  2. Direct output: When the instructions are executed, the information is fed out according to the states of the outputs. For example, OUT DY0.0.

2.2.6 Auxiliary Relays (M)

The auxiliary relay has contact A and contact B. It can be used several times in the program. Users can combine the control loops by means of the auxiliary relay, but cannot drive the external load by means of the auxiliary relay. The auxiliary relays can be divided into two types according to their attributes.

1. For general use: If an electric power cut occurs when the PLC is running, the auxiliary relay for general use will be reset to OFF. When the power supply is restored, the auxiliary relay for general use is still OFF.
2. For latched use: If an electric power cut occurs when the PLC is running, the state of the auxiliary relay for latched use will be retained. When the power supply is restored, the state remains the same as that before the power electric cut.

2.2.7 Special Auxiliary Relays (SM)

Every special auxiliary relay has its specific function. Please do not use the special auxiliary relays which are not defined.

The special auxiliary relays and their functions are listed as follows. As to the SM numbers marked "*", users can refer to the additional remarks on special auxiliary relays/special data registers. "R" in the attribute column indicates that the special auxiliary relay can read the data, whereas "R/W" in the attribute column indicates that it can read and write the data. In addition, the mark "–" indicates that the status of the special auxiliary relay does not make any change. The mark "#" indicates that the system will be set according to the status of the PLC, and users can read the setting value and refer to the related manual for more information.

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM0	Error of the operation or operand exceeding the allowed range	○	OFF	OFF	–	N	R	OFF
SM1	Error of the operation or operand exceeding the allowed range is locked.	○	OFF	OFF	–	N	R	OFF
SM5	Instruction inspection error	○	OFF	OFF	–	N	R	OFF
SM6	Data lost in the latched area	○	OFF	–	–	N	R/W	OFF
SM7	Not sufficient power supply (24V)	○	OFF	–	–	N	R	OFF
*SM8	Watchdog timer error	○	OFF	–	–	N	R	OFF
SM9	System error	○	OFF	–	–	N	R	OFF
SM10	I/O bus error	○	OFF	–	–	N	R	OFF
*SM22	Clearing the error log	○	OFF	OFF	OFF	N	R/W	OFF
SM23	Clearing the download log	○	OFF	OFF	OFF	N	R/W	OFF
SM24	Clearing the state-changing log of the PLC	○	OFF	OFF	OFF	N	R/W	OFF
SM25	The online-editing processing flag is on when the online-editing mode starts.	○	OFF	–	–	N	R	OFF
SM26	The debugging mode processing flag is on when the debugging mode starts.	○	OFF	–	–	N	R	OFF
SM28	Error occurs when the output point is the same as the output that instruction high speed used	○	OFF	OFF	OFF	N	R/W	OFF
SM30	Error occurs in the remote module	○	OFF	–	–	N	R	OFF
SM34	Wrong password	○	OFF	–	–	N	R/W	OFF
*SM36	Enable saving data to the memory card. When it is ON, the PLC will run according to the value in the SR36.	○	OFF	–	–	N	R/W	OFF
SM76	The data is sent through Function Card 1.	○	OFF	OFF	–	N	R/W	OFF
SM77	The data is sent through Function Card 2.	○	OFF	OFF	–	N	R/W	OFF
SM78	Waiting to receive the reply through Function Card 1	○	OFF	OFF	–	N	R	OFF
SM79	Waiting to receive the reply through Function Card 2	○	OFF	OFF	–	N	R	OFF
SM80	Reception through Function Card 1 is complete.	○	OFF	OFF	–	N	R/W	OFF
SM81	Reception through Function Card 2 is complete.	○	OFF	OFF	–	N	R/W	OFF
SM82	An error occurs during the reception of the data through Function Card 1 by using the instruction MODRW or the instruction RS.	○	OFF	OFF	–	N	R	OFF
SM83	An error occurs during the reception of the data through Function Card 2 by using the instruction MODRW or the	○	OFF	OFF	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	instruction RS.							
SM84	No data is received through Function Card 1 after a specified period of time.	○	OFF	OFF	–	N	R/W	OFF
SM85	No data is received through Function Card 2 after a specified period of time.	○	OFF	OFF	–	N	R/W	OFF
SM86	Choice made by Function Card 1 between the 8-bit processing mode and the 16-bit processing mode ON: The 8-bit processing mode OFF: The 16-bit processing mode	○	OFF	–	–	N	R/W	OFF
SM87	Choice made by Function Card 2 between the 8-bit processing mode and the 16-bit processing mode ON: The 8-bit processing mode OFF: The 16-bit processing mode	○	OFF	–	–	N	R/W	OFF
SM90	The communication protocol of Function Card 1 changes	○	OFF	–	–	N	R/W	OFF
SM91	The communication protocol of Function Card 2 changes	○	OFF	–	–	N	R/W	OFF
SM94	Change of the LED lighting control in COM1	○	–	–	–	H	R/W	OFF
SM95	Change of the LED lighting control in COM2	○	–	–	–	H	R/W	OFF
*SM96	The data is sent through COM1.	○	OFF	OFF	–	N	R/W	OFF
*SM97	The data is sent through COM2.	○	OFF	OFF	–	N	R/W	OFF
*SM98	Waiting to receive the reply through COM1	○	OFF	OFF	–	N	R	OFF
*SM99	Waiting to receive the reply through COM2	○	OFF	OFF	–	N	R	OFF
*SM100	Reception through COM1 is complete.	○	OFF	OFF	–	N	R/W	OFF
*SM101	Reception through COM2 is complete.	○	OFF	OFF	–	N	R/W	OFF
*SM102	An error occurs during the reception of the data through COM1 by using the instruction MODRW or the instruction RS.	○	OFF	OFF	–	N	R/W	OFF
*SM103	An error occurs during the reception of the data through COM2 by using the instruction MODRW or the instruction RS.	○	OFF	OFF	–	N	R/W	OFF
*SM104	No data is received through COM1 after a specified period of time.	○	OFF	OFF	–	N	R/W	OFF
*SM105	No data is received through COM2 after a specified period of time.	○	OFF	OFF	–	N	R/W	OFF
*SM106	Choice made by COM1 between the 8-bit processing mode and the 16-bit processing mode ON: The 8-bit processing mode OFF: The 16-bit processing mode	○	OFF	–	–	N	R/W	OFF
*SM107	Choice made by COM2 between the 8-bit processing mode and the 16-bit processing mode ON: The 8-bit processing mode OFF: The 16-bit processing mode	○	OFF	–	–	N	R/W	OFF
SM166	VR0 has been started (need to work with SR166)	○	OFF	–	–	N	R/W	OFF
SM167	VR1 has been started (need to work with SR167)	○	OFF	–	–	N	R/W	OFF
SM168	The connection for Function Card 1 has been established.	○	–	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM169	The Function Card 1 is in operation.	○	OFF	–	–	N	R	OFF
SM170	The connection for Function Card 2 has been established.	○	–	–	–	N	R	OFF
SM171	The Function Card 12 is in operation.	○	OFF	–	–	N	R	OFF
*SM204	All non-latched areas are cleared.	○	OFF	OFF	OFF	N	R/W	OFF
*SM205	All latched areas are cleared.	○	OFF	OFF	OFF	N	R/W	OFF
SM206	Inhibiting all output	○	OFF	–	–	N	R/W	OFF
*SM209	The communication protocol of COM1 changes	○	OFF	OFF	OFF	N	R/W	OFF
*SM210	Choice made by COM1 between the ASCII mode and the RTU mode ON: The RTU mode	○	–	–	–	H	R/W	OFF
*SM211	The communication protocol of COM1 changes	○	OFF	OFF	OFF	N	R/W	OFF
*SM212	Choice made by COM2 between the ASCII mode and the RTU mode ON: The RTU mode	○	–	–	–	H	R/W	OFF
SM215	Running state of the PLC	○	OFF	ON	OFF	N	R/W	OFF
SM218	Error occurs when the real-time clock is malfunction	○	–	–	–	N	R	OFF
SM219	Error occurs when the power of the battery for the real-time clock is low	○	–	–	–	N	R	OFF
*SM220	Calibrating the real-time clock within ±30 seconds	○	OFF	OFF	–	N	R/W	OFF
SM270	The flag of reversing the input direction for MPG 1 (X0.0/X0.1)	○	OFF	OFF	–	N	R/W	OFF
SM271	The flag of reversing the input direction for MPG 2 (X0.2/X0.3)	○	OFF	OFF	–	N	R/W	OFF
SM272	The flag of reversing the input direction for MPG 3 (X0.4/X0.5)	○	OFF	OFF	–	N	R/W	OFF
SM273	The flag of reversing the input direction for MPG 4 (X0.6/X0.7)	○	OFF	OFF	–	N	R/W	OFF
SM274	The flag of reversing the input direction for MPG 5 (X0.8/X0.9)	○	OFF	OFF	–	N	R/W	OFF
SM275	The flag of reversing the input direction for MPG 6 (X0.10/X0.11)	○	OFF	OFF	–	N	R/W	OFF
SM221	Enable the daylight saving time	○	–	–	–	H	R	OFF
SM281	The flag of reversing the input direction for high-speed counter 1	○	OFF	OFF	–	N	R/W	OFF
SM282	The flag of reversing the input direction for high-speed counter 2	○	OFF	OFF	–	N	R/W	OFF
SM283	The flag of reversing the input direction for high-speed counter 3	○	OFF	OFF	–	N	R/W	OFF
SM284	The flag of reversing the input direction for high-speed counter 4	○	OFF	OFF	–	N	R/W	OFF
SM285	The flag of reversing the input direction for high-speed counter 5	○	OFF	OFF	–	N	R/W	OFF
SM286	The flag of reversing the input direction for high-speed counter 6	○	OFF	OFF	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM287	The flag of reversing the input direction for high-speed counter 7	○	OFF	OFF	–	N	R/W	OFF
SM288	The flag of reversing the input direction for high-speed counter 8	○	OFF	OFF	–	N	R/W	OFF
SM291	The flag of clearing the input point for high-speed counter 1	○	OFF	OFF	–	N	R/W	OFF
SM292	The flag of clearing the input point for high-speed counter 2	○	OFF	OFF	–	N	R/W	OFF
SM293	The flag of clearing the input point for high-speed counter 3	○	OFF	OFF	–	N	R/W	OFF
SM294	The flag of clearing the input point for high-speed counter 4	○	OFF	OFF	–	N	R/W	OFF
SM295	The flag of clearing the input point for high-speed counter 5	○	OFF	OFF	–	N	R/W	OFF
SM296	The flag of clearing the input point for high-speed counter 6	○	OFF	OFF	–	N	R/W	OFF
SM300	It sets the counting mode of HC200. (HC200 counts down when SM300 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM301	It sets the counting mode of HC201. (HC201 counts down when SM301 is ON.)	○	OFF	–	–	N	R	OFF
SM302	It sets the counting mode of HC202. (HC202 counts down when SM302 is ON.)	○	OFF	–	–	N	R	OFF
SM303	It sets the counting mode of HC203. (HC203 counts down when SM303 is ON.)	○	OFF	–	–	N	R	OFF
SM304	It sets the counting mode of HC204. (HC204 counts down when SM304 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM305	It sets the counting mode of HC205. (HC205 counts down when SM305 is ON.)	○	OFF	–	–	N	R	OFF
SM306	It sets the counting mode of HC206. (HC206 counts down when SM306 is ON.)	○	OFF	–	–	N	R	OFF
SM307	It sets the counting mode of HC207. (HC207 counts down when SM307 is ON.)	○	OFF	–	–	N	R	OFF
SM308	It sets the counting mode of HC208. (HC208 counts down when SM308 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM309	It sets the counting mode of HC209. (HC209 counts down when SM309 is ON.)	○	OFF	–	–	N	R	OFF
SM310	It sets the counting mode of HC210. (HC210 counts down when SM310 is ON.)	○	OFF	–	–	N	R	OFF
SM311	It sets the counting mode of HC211. (HC211 counts down when SM311 is ON.)	○	OFF	–	–	N	R	OFF
SM312	It sets the counting mode of HC212. (HC212 counts down when SM312 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM313	It sets the counting mode of HC213. (HC213 counts down when SM313 is ON.)	○	OFF	–	–	N	R	OFF
SM314	It sets the counting mode of HC214. (HC214 counts down when SM314 is ON.)	○	OFF	–	–	N	R	OFF
SM315	It sets the counting mode of HC215. (HC215 counts down when SM315 is ON.)	○	OFF	–	–	N	R	OFF
SM316	It sets the counting mode of HC216. (HC216 counts down when SM316 is ON.)	○	OFF	OFF	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM317	It sets the counting mode of HC217. (HC217 counts down when SM317 is ON.)	○	OFF	–	–	N	R	OFF
SM318	It sets the counting mode of HC218. (HC218 counts down when SM318 is ON.)	○	OFF	–	–	N	R	OFF
SM319	It sets the counting mode of HC219. (HC219 counts down when SM319 is ON.)	○	OFF	–	–	N	R	OFF
SM320	It sets the counting mode of HC220. (HC220 counts down when SM320 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM321	It sets the counting mode of HC221. (HC221 counts down when SM321 is ON.)	○	OFF	–	–	N	R	OFF
SM322	It sets the counting mode of HC222. (HC222 counts down when SM322 is ON.)	○	OFF	–	–	N	R	OFF
SM323	It sets the counting mode of HC223. (HC223 counts down when SM323 is ON.)	○	OFF	–	–	N	R	OFF
SM332	It sets the counting mode of HC232. (HC232 counts down when SM332 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM333	It sets the counting mode of HC233. (HC233 counts down when SM333 is ON.)	○	OFF	–	–	N	R	OFF
SM334	It sets the counting mode of HC234. (HC234 counts down when SM334 is ON.)	○	OFF	–	–	N	R	OFF
SM335	It sets the counting mode of HC235. (HC235 counts down when SM335 is ON.)	○	OFF	–	–	N	R	OFF
SM336	It sets the counting mode of HC236. (HC236 counts down when SM336 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM337	It sets the counting mode of HC237. (HC237 counts down when SM337 is ON.)	○	OFF	–	–	N	R	OFF
SM338	It sets the counting mode of HC238. (HC238 counts down when SM338 is ON.)	○	OFF	–	–	N	R	OFF
SM339	It sets the counting mode of HC239. (HC239 counts down when SM339 is ON.)	○	OFF	–	–	N	R	OFF
SM340	It sets the counting mode of HC240. (HC240 counts down when SM340 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM341	It sets the counting mode of HC241. (HC241 counts down when SM341 is ON.)	○	OFF	–	–	N	R	OFF
SM342	It sets the counting mode of HC242. (HC242 counts down when SM342 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM343	It sets the counting mode of HC243. (HC243 counts down when SM343 is ON.)	○	OFF	–	–	N	R	OFF
SM344	It sets the counting mode of HC244. (HC244 counts down when SM344 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM345	It sets the counting mode of HC245. (HC245 counts down when SM345 is ON.)	○	OFF	–	–	N	R	OFF
SM346	It sets the counting mode of HC246. (HC246 counts down when SM346 is ON.)	○	OFF	OFF	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM347	It sets the counting mode of HC247. (HC247 counts down when SM347 is ON.)	○	OFF	–	–	N	R	OFF
SM348	It sets the counting mode of HC248. (HC248 counts down when SM348 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM349	It sets the counting mode of HC249. (HC249 counts down when SM349 is ON.)	○	OFF	–	–	N	R	OFF
SM350	It sets the counting mode of HC250. (HC250 counts down when SM350 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM351	It sets the counting mode of HC251. (HC251 counts down when SM351 is ON.)	○	OFF	–	–	N	R	OFF
SM352	It sets the counting mode of HC252. (HC252 counts down when SM352 is ON.)	○	OFF	OFF	–	N	R/W	OFF
SM353	It sets the counting mode of HC253. (HC253 counts down when SM353 is ON.)	○	OFF	OFF	–	N	R/W	OFF
*SM400	Normally-open contact	○	OFF	ON	OFF	N	R	OFF
*SM401	Normally-closed contact	○	OFF	OFF	ON	N	R	OFF
*SM402	The pulse is ON at the time when the PLC runs.	○	OFF	ON	OFF	N	R	OFF
*SM403	The pulse is OFF at the time when the PLC runs.	○	OFF	OFF	ON	N	R	OFF
*SM404	10 millisecond clock pulse during which the pulse is ON for 5 milliseconds and is OFF for 5 milliseconds	○	OFF	–	–	N	R	OFF
*SM405	100 millisecond clock pulse during which the pulse is ON for 50 milliseconds and is OFF for 50 milliseconds	○	OFF	–	–	N	R	OFF
*SM406	200 millisecond clock pulse during which the pulse is ON for 100 milliseconds and is OFF for 100 milliseconds	○	OFF	–	–	N	R	OFF
*SM407	One second clock pulse during which the pulse is ON for 500 milliseconds and is OFF for 500 milliseconds	○	OFF	–	–	N	R	OFF
*SM450	Whether the memory card exists ON: The memory card exists. OFF: The memory card does not exist.	○	–	–	–	N	R	OFF
*SM452	The data in the memory card is being accessed. ON: The data in the memory card is being accessed. OFF: The data in the memory card is not accessed.	○	OFF	–	–	N	R	OFF
*SM453	An error occurs during the operation of the memory card. ON: An error occurs.	○	OFF	–	–	N	R	OFF
SM454	Enabling/disabling the data logger. (ON: enable, OFF: disable)	○	OFF	–	–	N	R/W	OFF
SM455	The data logger is currently taking samples. (ON: buffer is full or in cycle)	○	OFF	–	–	N	R	OFF
*SM456	Execution of data logger and the memory card. (ON: execution by the values in SR902)	○	OFF	–	–	N	R/W	OFF
SM457	The state of the sample parameters in data logger (ON: the sample parameter is set)	○	–	–	–	N	R	OFF
SM460	Y0.0/axis 1 (Y0.0/Y0.1) is outputting.	○	OFF	OFF	–	N	R	OFF
SM461	The outputting of Y0.0/axis 1 (Y0.0/Y0.1) is complete.	○	OFF	OFF	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM462	Reversing the output direction of axis 1 (Y0.1)	○	OFF	OFF	–	N	R/W	OFF
*SM463	Stopping the output of Y0.0/axis 1 (Y0.0/Y0.1)	○	OFF	OFF	–	N	R/W	OFF
SM464	Enabling the positive maximum value for axis 1 (Y0.0/Y0.1)	○	–	–	–	Y	R/W	OFF
SM465	The alarm of the positive limit for axis 1 (Y0.0/Y0.1)	○	OFF	OFF	–	N	R/W	OFF
SM466	Enabling the negative maximum value for axis 1 (Y0.0/Y0.1)	○	–	–	–	Y	R/W	OFF
SM467	The alarm of the negative limit for axis 1 (Y0.0/Y0.1)	○	OFF	OFF	–	N	R/W	OFF
*SM468	Enabling the S curve ramp-up/down for axis 1 (Y0.0/Y0.1)	○	OFF	OFF	–	N	R/W	OFF
SM469	Enabling fixed slope ramp-up/down for axis 1 (Y0.0/Y0.1)	○	OFF	OFF	–	N	R/W	OFF
SM470	Output completion auto-reset for Y0.0/axis 1 (Y0.0/Y0.1)	○	OFF	OFF	–	N	R/W	OFF
SM471	Executing an interrupt I500 when pulse output ends for axis 1 (Y0.0/Y0.1)	○	OFF	OFF	–	N	R/W	OFF
SM472	Y0.1 is outputting.	○	OFF	OFF	–	N	R	OFF
SM473	The outputting of Y0.1 is complete.	○	OFF	OFF	–	N	R/W	OFF
*SM474	Stopping the output of Y0.1.	○	OFF	OFF	–	N	R/W	OFF
SM475	Output completion auto-reset for Y0.1	○	OFF	OFF	–	N	R/W	OFF
*SM476	The output immediately stops when the instruction is disabled or stops for Y0.0/axis 1 (Y0.0/Y0.1)	○	OFF	OFF	–	N	R/W	OFF
*SM477	The output immediately stops when the instruction is disabled or stops for Y0.1	○	OFF	OFF	–	N	R/W	OFF
SM480	Y0.2/axis 2 (Y0.2/Y0.3) is outputting.	○	OFF	OFF	–	N	R	OFF
SM481	The outputting of Y0.2/axis 2 (Y0.2/Y0.3) is complete.	○	OFF	OFF	–	N	R/W	OFF
SM482	Reversing the output direction of axis 2 (Y0.3)	○	OFF	OFF	–	N	R/W	OFF
*SM483	Stopping the output of Y0.2/axis 2 (Y0.2/Y0.3)	○	OFF	OFF	–	N	R/W	OFF
SM484	Enabling the positive maximum value for axis 2 (Y0.2/Y0.3)	○	–	–	–	Y	R/W	OFF
SM485	The alarm of the positive limit for axis 2 (Y0.2/Y0.3)	○	OFF	OFF	–	N	R/W	OFF
SM486	Enabling the negative maximum value for axis 2 (Y0.2/Y0.3)	○	–	–	–	Y	R/W	OFF
SM487	The alarm of the negative limit for axis 2 (Y0.2/Y0.3)	○	OFF	OFF	–	N	R/W	OFF
*SM488	Enabling the S curve ramp-up/down for axis 2 (Y0.2/Y0.3)	○	OFF	OFF	–	N	R/W	OFF
SM489	Enabling fixed slope ramp-up/down for axis 2 (Y0.2/Y0.3)	○	OFF	OFF	–	N	R/W	OFF
SM490	Output completion auto-reset for Y0.2/axis 2 (Y0.2/Y0.3)	○	OFF	OFF	–	N	R/W	OFF
SM491	Executing an interrupt I501 when pulse output ends for axis 2 (Y0.2/Y0.3)	○	OFF	OFF	–	N	R/W	OFF
SM492	Y0.3 is outputting.	○	OFF	OFF	–	N	R	OFF
SM493	The outputting of Y0.3 is complete.	○	OFF	OFF	–	N	R/W	OFF
*SM494	Stopping the output of Y0.3.	○	OFF	OFF	–	N	R/W	OFF
SM495	Output completion auto-reset for Y0.3	○	OFF	OFF	–	N	R/W	OFF
*SM496	The output immediately stops when the instruction is disabled or stops for Y0.2/axis 2 (Y0.2/Y0.3)	○	OFF	OFF	–	N	R/W	OFF
*SM497	The output immediately stops when the instruction is disabled or stops for Y0.3.	○	OFF	OFF	–	N	R/W	OFF
SM500	Y0.4/axis 3 (Y0.4/Y0.5) is outputting.	○	OFF	OFF	–	N	R	OFF
SM501	The outputting of Y0.4/axis 3 (Y0.4/Y0.5) is complete.	○	OFF	OFF	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM502	Reversing the output direction of axis 3 (Y0.5)	○	OFF	OFF	–	N	R/W	OFF
*SM503	Stopping the output of Y0.4/axis 3 (Y0.4/Y0.5)	○	OFF	OFF	–	N	R/W	OFF
SM504	Enabling the positive maximum value for axis 3 (Y0.4/Y0.5)	○	–	–	–	Y	R/W	OFF
SM505	The alarm of the positive limit for axis 3 (Y0.4/Y0.5)	○	OFF	OFF	–	N	R/W	OFF
SM506	Enabling the negative maximum value for axis 3 (Y0.4/Y0.5)	○	–	–	–	Y	R/W	OFF
SM507	The alarm of the negative limit for axis 3 (Y0.4/Y0.5)	○	OFF	OFF	–	N	R/W	OFF
*SM508	Enabling the S curve ramp-up/down for axis 3 (Y0.4/Y0.5)	○	OFF	OFF	–	N	R/W	OFF
SM509	Enabling fixed slope ramp-up/down for axis 3 (Y0.4/Y0.5)	○	OFF	OFF	–	N	R/W	OFF
SM510	Output completion auto-reset for Y0.4/axis 3 (Y0.4/Y0.5)	○	OFF	OFF	–	N	R/W	OFF
SM511	Executing an interrupt I502 when pulse output ends for axis 3 (Y0.4/Y0.5)	○	OFF	OFF	–	N	R/W	OFF
SM512	Y0.5 is outputting.	○	OFF	OFF	–	N	R	OFF
SM513	The outputting of Y0.5 is complete.	○	OFF	OFF	–	N	R/W	OFF
*SM514	Stopping the output of Y0.5.	○	OFF	OFF	–	N	R/W	OFF
SM515	Output completion auto-reset for Y0.5	○	OFF	OFF	–	N	R/W	OFF
*SM516	The output immediately stops when the instruction is disabled or stops for Y0.4/axis 3 (Y0.4/Y0.5)	○	OFF	OFF	–	N	R/W	OFF
*SM517	The output immediately stops when the instruction is disabled or stops for Y0.5	○	OFF	OFF	–	N	R/W	OFF
SM520	Y0.6/axis 4 (Y0.6/Y0.7) is outputting.	○	OFF	OFF	–	N	R	OFF
SM521	The outputting of Y0.6/axis 4 (Y0.6/Y0.7) is complete.	○	OFF	OFF	–	N	R/W	OFF
SM522	Reversing the output direction of axis 4 (Y0.7)	○	OFF	OFF	–	N	R/W	OFF
*SM523	Stopping the output of Y0.6/axis 4 (Y0.6/Y0.7)	○	OFF	OFF	–	N	R/W	OFF
SM524	Enabling the positive maximum value for axis 4 (Y0.6/Y0.7)	○	–	–	–	Y	R/W	OFF
SM525	The alarm of the positive limit for axis 4 (Y0.6/Y0.7)	○	OFF	OFF	–	N	R/W	OFF
SM526	Enabling the negative maximum value for axis 4 (Y0.6/Y0.7)	○	–	–	–	Y	R/W	OFF
SM527	The alarm of the negative limit for axis 4 (Y0.6/Y0.7)	○	OFF	OFF	–	N	R/W	OFF
*SM528	Enabling the S curve ramp-up/down for axis 4 (Y0.6/Y0.7)	○	OFF	OFF	–	N	R/W	OFF
SM529	Enabling fixed slope ramp-up/down for axis 4 (Y0.6/Y0.7)	○	OFF	OFF	–	N	R/W	OFF
SM530	Output completion auto-reset for Y0.6/axis 4 (Y0.6/Y0.7)	○	OFF	OFF	–	N	R/W	OFF
SM531	Executing an interrupt I503 when pulse output ends for axis 4 (Y0.6/Y0.7)	○	OFF	OFF	–	N	R/W	OFF
SM532	Y0.7 is outputting.	○	OFF	OFF	–	N	R	OFF
SM533	The outputting of Y0.7 is complete.	○	OFF	OFF	–	N	R/W	OFF
*SM534	Stopping the output of Y0.7.	○	OFF	OFF	–	N	R/W	OFF
SM535	Output completion auto-reset for Y0.7	○	OFF	OFF	–	N	R/W	OFF
*SM536	The output immediately stops when the instruction is disabled or stops for Y0.6/axis 4 (Y0.6/Y0.7)	○	OFF	OFF	–	N	R/W	OFF
*SM537	The output immediately stops when the instruction is disabled or stops for Y0.7	○	OFF	OFF	–	N	R/W	OFF
SM540	Y0.8/axis 5 (Y0.8/Y0.9) is outputting.	○	OFF	OFF	–	N	R	OFF
SM541	The outputting of Y0.8/axis 5 (Y0.8/Y0.9) is complete.	○	OFF	OFF	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM542	Reversing the output direction of axis 5 (Y0.9)	○	OFF	OFF	–	N	R/W	OFF
*SM543	Stopping the output of Y0.8/axis 5 (Y0.8/Y0.9)	○	OFF	OFF	–	N	R/W	OFF
SM544	Enabling the positive maximum value for axis 5 (Y0.8/Y0.9)	○	–	–	–	Y	R/W	OFF
SM545	The alarm of the positive limit for axis 5 (Y0.8/Y0.9)	○	OFF	OFF	–	N	R/W	OFF
SM546	Enabling the negative maximum value for axis 5 (Y0.8/Y0.9)	○	–	–	–	Y	R/W	OFF
SM547	The alarm of the negative limit for axis 5 (Y0.8/Y0.9)	○	OFF	OFF	–	N	R/W	OFF
*SM548	Enabling the S curve ramp-up/down for axis 5 (Y0.8/Y0.9)	○	OFF	OFF	–	N	R/W	OFF
SM549	Enabling fixed slope ramp-up/down for axis 5 (Y0.8/Y0.9)	○	OFF	OFF	–	N	R/W	OFF
SM550	Output completion auto-reset for Y0.8/axis 5 (Y0.8/Y0.9)	○	OFF	OFF	–	N	R/W	OFF
SM551	Executing an interrupt I504 when pulse output ends for axis 5 (Y0.8/Y0.9)	○	OFF	OFF	–	N	R/W	OFF
SM552	Y0.9 is outputting.	○	OFF	OFF	–	N	R	OFF
SM553	The outputting of Y0.9 is complete.	○	OFF	OFF	–	N	R/W	OFF
*SM554	Stopping the output of Y0.9.	○	OFF	OFF	–	N	R/W	OFF
SM555	Output completion auto-reset for Y0.9	○	OFF	OFF	–	N	R/W	OFF
*SM556	The output immediately stops when the instruction is disabled or stops for Y0.8/axis 5 (Y0.8/Y0.9)	○	OFF	OFF	–	N	R/W	OFF
*SM557	The output immediately stops when the instruction is disabled or stops for Y0.9	○	OFF	OFF	–	N	R/W	OFF
SM560	Y0.10/axis 6 (Y0.10/Y0.11) is outputting.	○	OFF	OFF	–	N	R	OFF
SM561	The outputting of Y0.10/axis 6 (Y0.10/Y0.11) is complete.	○	OFF	OFF	–	N	R/W	OFF
SM562	Reversing the output direction of axis 6 (Y0.11)	○	OFF	OFF	–	N	R/W	OFF
*SM563	Stopping the output of Y0.10/axis 6 (Y0.10/Y0.11)	○	OFF	OFF	–	N	R/W	OFF
SM564	Enabling the positive maximum value for axis 6 (Y0.10/Y0.11)	○	–	–	–	Y	R/W	OFF
SM565	The alarm of the positive limit for axis 6 (Y0.10/Y0.11)	○	OFF	OFF	–	N	R/W	OFF
SM566	Enabling the negative maximum value for axis 6 (Y0.10/Y0.11)	○	–	–	–	Y	R/W	OFF
SM567	The alarm of the negative limit for axis 6 (Y0.10/Y0.11)	○	OFF	OFF	–	N	R/W	OFF
*SM568	Enabling the S curve ramp-up/down for axis 6 (Y0.10/Y0.11)	○	OFF	OFF	–	N	R/W	OFF
SM569	Enabling fixed slope ramp-up/down for axis 6 (Y0.10/Y0.11)	○	OFF	OFF	–	N	R/W	OFF
SM570	Output completion auto-reset for Y0.10/axis 6 (Y0.10/Y0.11)	○	OFF	OFF	–	N	R/W	OFF
SM571	Executing an interrupt I505 when pulse output ends for axis 6 (Y0.10/Y0.11)	○	OFF	OFF	–	N	R/W	OFF
SM572	Y0.11 is outputting.	○	OFF	OFF	–	N	R	OFF
SM573	The outputting of Y0.11 is complete.	○	OFF	OFF	–	N	R/W	OFF
*SM574	Stopping outputting of the Y0.11.	○	OFF	OFF	–	N	R/W	OFF
SM575	Output completion auto-reset for Y0.11	○	OFF	OFF	–	N	R/W	OFF
*SM576	The output immediately stops when the instruction is disabled or stops for Y0.10/axis 6 (Y0.10/Y0.11)	○	OFF	OFF	–	N	R/W	OFF
*SM577	The output immediately stops when the instruction is disabled or stops for Y0.11	○	OFF	OFF	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM580	All outputs immediately stop when the instruction is disabled or stops.	○	OFF	OFF	–	N	R/W	OFF
SM600	Zero flag	○	OFF	–	–	N	R	OFF
SM601	Borrow flag	○	OFF	–	–	N	R	OFF
SM602	Carry flag	○	OFF	–	–	N	R	OFF
SM604	Setting the working mode of the instruction SORT. ON: The descending order OFF: The ascending order	○	OFF	–	–	N	R/W	OFF
SM605	Designating the working mode of the instruction SMOV	○	OFF	–	–	N	R/W	OFF
SM606	8-bit or 16-bit working mode	○	OFF	–	–	N	R/W	OFF
SM607	It is the matrix comparison flag. ON: Comparing the equivalent values OFF: Comparing the different values	○	OFF	–	–	N	R/W	OFF
SM608	The matrix comparison comes to an end. When the last bits are compared, SM608 is ON.	○	OFF	–	–	N	R	OFF
SM609	When SM609 is ON, the comparison starts from bit 0.	○	OFF	–	–	N	R	OFF
SM610	It is the matrix bit search flag. When the matching bits are compared, the comparison stops immediately, and SM610 is ON.	○	OFF	–	–	N	R	OFF
SM611	It is the matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.	○	OFF	–	–	N	R	OFF
SM612	It is the matrix pointer increasing flag. The current value of the pointer increases by one.	○	OFF	–	–	N	R/W	OFF
SM613	It is the matrix pointer clearing flag. The current value of the pointer is cleared to zero.	○	OFF	–	–	N	R/W	OFF
SM614	It is the carry flag for the matrix rotation/shift/output.	○	OFF	–	–	N	R	OFF
SM615	It is the borrow flag for the matrix shift/output.	○	OFF	–	–	N	R/W	OFF
SM616	It is the direction flag for the matrix rotation/shift. The bits are shifted leftward when SM616 is OFF, whereas the bits are shifted rightward when SM616 is ON.	○	OFF	–	–	N	R/W	OFF
SM617	The bits with the value 0 or 1 are counted.	○	OFF	–	–	N	R/W	OFF
SM618	It is ON when the matrix counting result is 0.	○	OFF	–	–	N	R/W	OFF
SM619	It is ON when the instruction EI is executed.	○	OFF	OFF	–	N	R	OFF
SM620	When the results gotten from the comparison by using the instruction CMPT# are that all devices are ON, SM620 is ON.	○	OFF	–	–	N	R	OFF
SM621	It sets the counting mode of HC0. (HC0 counts down when SM621 is ON.)	○	OFF	–	–	N	R/W	OFF
SM622	It sets the counting mode of HC. (HC1 counts down when SM622 is ON.)	○	OFF	–	–	N	R/W	OFF
SM623	It sets the counting mode of HC2. (HC2 counts down when SM623 is ON.)	○	OFF	–	–	N	R/W	OFF
SM624	It sets the counting mode of HC3. (HC3 counts down when SM624 is ON.)	○	OFF	–	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM625	It sets the counting mode of HC4. (HC4 counts down when SM625 is ON.)	○	OFF	–	–	N	R/W	OFF
SM626	It sets the counting mode of HC5. (HC5 counts down when SM626 is ON.)	○	OFF	–	–	N	R/W	OFF
SM627	It sets the counting mode of HC6. (HC6 counts down when SM627 is ON.)	○	OFF	–	–	N	R/W	OFF
SM628	It sets the counting mode of HC7. (HC7 counts down when SM628 is ON.)	○	OFF	–	–	N	R/W	OFF
SM629	It sets the counting mode of HC8. (HC8 counts down when SM629 is ON.)	○	OFF	–	–	N	R/W	OFF
SM630	It sets the counting mode of HC9. (HC9 counts down when SM630 is ON.)	○	OFF	–	–	N	R/W	OFF
SM631	It sets the counting mode of HC10. (HC10 counts down when SM631 is ON.)	○	OFF	–	–	N	R/W	OFF
SM632	It sets the counting mode of HC11. (HC11 counts down when SM632 is ON.)	○	OFF	–	–	N	R/W	OFF
SM633	It sets the counting mode of HC12. (HC12 counts down when SM633 is ON.)	○	OFF	–	–	N	R/W	OFF
SM634	It sets the counting mode of HC13. (HC13 counts down when SM634 is ON.)	○	OFF	–	–	N	R/W	OFF
SM635	It sets the counting mode of HC14. (HC14 counts down when SM635 is ON.)	○	OFF	–	–	N	R/W	OFF
SM636	It sets the counting mode of HC15. (HC15 counts down when SM636 is ON.)	○	OFF	–	–	N	R/W	OFF
SM637	It sets the counting mode of HC16. (HC16 counts down when SM637 is ON.)	○	OFF	–	–	N	R/W	OFF
SM638	It sets the counting mode of HC17. (HC17 counts down when SM638 is ON.)	○	OFF	–	–	N	R/W	OFF
SM639	It sets the counting mode of HC18. (HC18 counts down when SM639 is ON.)	○	OFF	–	–	N	R/W	OFF
SM640	It sets the counting mode of HC19. (HC19 counts down when SM640 is ON.)	○	OFF	–	–	N	R/W	OFF
SM641	It sets the counting mode of HC20. (HC20 counts down when SM641 is ON.)	○	OFF	–	–	N	R/W	OFF
SM642	It sets the counting mode of HC21. (HC21 counts down when SM642 is ON.)	○	OFF	–	–	N	R/W	OFF
SM643	It sets the counting mode of HC22. (HC22 counts down when SM643 is ON.)	○	OFF	–	–	N	R/W	OFF
SM644	It sets the counting mode of HC23. (HC23 counts down when SM644 is ON.)	○	OFF	–	–	N	R/W	OFF
SM645	It sets the counting mode of HC24. (HC24 counts down when SM645 is ON.)	○	OFF	–	–	N	R/W	OFF
SM646	It sets the counting mode of HC25. (HC25 counts down when SM646 is ON.)	○	OFF	–	–	N	R/W	OFF
SM647	It sets the counting mode of HC26. (HC26 counts down when SM647 is ON.)	○	OFF	–	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM648	It sets the counting mode of HC27. (HC27 counts down when SM648 is ON.)	○	OFF	–	–	N	R/W	OFF
SM649	It sets the counting mode of HC28. (HC28 counts down when SM649 is ON.)	○	OFF	–	–	N	R/W	OFF
SM650	It sets the counting mode of HC29. (HC29 counts down when SM650 is ON.)	○	OFF	–	–	N	R/W	OFF
SM651	It sets the counting mode of HC30. (HC30 counts down when SM651 is ON.)	○	OFF	–	–	N	R/W	OFF
SM652	It sets the counting mode of HC31. (HC31 counts down when SM652 is ON.)	○	OFF	–	–	N	R/W	OFF
SM653	It sets the counting mode of HC32. (HC32 counts down when SM653 is ON.)	○	OFF	–	–	N	R/W	OFF
SM654	It sets the counting mode of HC33. (HC33 counts down when SM653 is ON.)	○	OFF	–	–	N	R/W	OFF
SM655	It sets the counting mode of HC34. (HC34 counts down when SM655 is ON.)	○	OFF	–	–	N	R/W	OFF
SM656	It sets the counting mode of HC35. (HC35 counts down when SM656 is ON.)	○	OFF	–	–	N	R/W	OFF
SM657	It sets the counting mode of HC36. (HC36 counts down when SM657 is ON.)	○	OFF	–	–	N	R/W	OFF
SM658	It sets the counting mode of HC37. (HC37 counts down when SM658 is ON.)	○	OFF	–	–	N	R/W	OFF
SM659	It sets the counting mode of HC38. (HC38 counts down when SM659 is ON.)	○	OFF	–	–	N	R/W	OFF
SM660	It sets the counting mode of HC39. (HC39 counts down when SM660 is ON.)	○	OFF	–	–	N	R/W	OFF
SM661	It sets the counting mode of HC40. (HC40 counts down when SM661 is ON.)	○	OFF	–	–	N	R/W	OFF
SM662	It sets the counting mode of HC41. (HC41 counts down when SM662 is ON.)	○	OFF	–	–	N	R/W	OFF
SM663	It sets the counting mode of HC42. (HC42 counts down when SM663 is ON.)	○	OFF	–	–	N	R/W	OFF
SM664	It sets the counting mode of HC43. (HC43 counts down when SM664 is ON.)	○	OFF	–	–	N	R/W	OFF
SM665	It sets the counting mode of HC44. (HC44 counts down when SM665 is ON.)	○	OFF	–	–	N	R/W	OFF
SM666	It sets the counting mode of HC45. (HC45 counts down when SM666 is ON.)	○	OFF	–	–	N	R/W	OFF
SM667	It sets the counting mode of HC46. (HC46 counts down when SM667 is ON.)	○	OFF	–	–	N	R/W	OFF
SM668	It sets the counting mode of HC47. (HC47 counts down when SM668 is ON.)	○	OFF	–	–	N	R/W	OFF
SM669	It sets the counting mode of HC48. (HC48 counts down when SM669 is ON.)	○	OFF	–	–	N	R/W	OFF
SM670	It sets the counting mode of HC49. (HC49 counts down when SM670 is ON.)	○	OFF	–	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM671	It sets the counting mode of HC50. (HC50 counts down when SM671 is ON.)	○	OFF	–	–	N	R/W	OFF
SM672	It sets the counting mode of HC51. (HC51 counts down when SM672 is ON.)	○	OFF	–	–	N	R/W	OFF
SM673	It sets the counting mode of HC52. (HC52 counts down when SM673 is ON.)	○	OFF	–	–	N	R/W	OFF
SM674	It sets the counting mode of HC53. (HC53 counts down when SM674 is ON.)	○	OFF	–	–	N	R/W	OFF
SM675	It sets the counting mode of HC54. (HC54 counts down when SM675 is ON.)	○	OFF	–	–	N	R/W	OFF
SM676	It sets the counting mode of HC55. (HC55 counts down when SM676 is ON.)	○	OFF	–	–	N	R/W	OFF
SM677	It sets the counting mode of HC56. (HC56 counts down when SM677 is ON.)	○	OFF	–	–	N	R/W	OFF
SM678	It sets the counting mode of HC57. (HC57 counts down when SM678 is ON.)	○	OFF	–	–	N	R/W	OFF
SM679	It sets the counting mode of HC58. (HC58 counts down when SM679 is ON.)	○	OFF	–	–	N	R/W	OFF
SM680	It sets the counting mode of HC59. (HC59 counts down when SM680 is ON.)	○	OFF	–	–	N	R/W	OFF
SM681	It sets the counting mode of HC60. (HC60 counts down when SM681 is ON.)	○	OFF	–	–	N	R/W	OFF
SM682	It sets the counting mode of HC61. (HC61 counts down when SM682 is ON.)	○	OFF	–	–	N	R/W	OFF
SM683	It sets the counting mode of HC62. (HC62 counts down when SM683 is ON.)	○	OFF	–	–	N	R/W	OFF
SM684	It sets the counting mode of HC63. (HC63 counts down when SM684 is ON.)	○	OFF	–	–	N	R/W	OFF
SM685	The instruction DSCLP uses the floating-point operation.	○	OFF	–	–	N	R/W	OFF
SM686	Mode of the instruction RAMP	○	OFF	–	–	N	R/W	OFF
SM687	The execution of the instruction RAMP is complete.	○	OFF	–	–	N	R/W	OFF
SM688	The execution of the instruction INCD is complete.	○	OFF	–	–	N	R/W	OFF
SM690	String control mode	○	OFF	–	–	N	R/W	OFF
SM691	The input mode of the instruction HKY is the 16-bit mode. The input is the hexadecimal input if SM691 is ON, whereas A~F are function keys if it is OFF.	○	OFF	–	–	N	R/W	OFF
SM692	After the execution of the instruction HKY is complete, SM692 is ON for a scan cycle.	○	OFF	–	–	N	R/W	OFF
SM693	After the execution of the instruction SEGL is complete, SM693 is ON for a scan cycle.	○	OFF	–	–	N	R/W	OFF
SM694	After the execution of the instruction DSW is complete, SM694 is ON for a scan cycle.	○	OFF	–	–	N	R/W	OFF
SM695	It is the radian/degree flag. ON: The degree	○	OFF	–	–	N	R/W	OFF
SM749	An error occurs in the initialization of the data exchange via	○	OFF	–	–	N	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	COM1							
*SM750	Data exchange via COM1 has been enabled by ISPSoft.	○	OFF	–	–	H	R/W	OFF
*SM752	Connection 1 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM753	Connection 2 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM754	Connection 3 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM755	Connection 4 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM756	Connection 5 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM757	Connection 6 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM758	Connection 7 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM759	Connection 8 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM760	Connection 9 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM761	Connection 10 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM762	Connection 11 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM763	Connection 12 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM764	Connection 13 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM765	Connection 14 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM766	Connection 15 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM767	Connection 16 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM768	Connection 17 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM769	Connection 18 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM770	Connection 19 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM771	Connection 20 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM772	Connection 21 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM773	Connection 22 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM774	Connection 23 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM775	Connection 24 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM776	Connection 25 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM777	Connection 26 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM778	Connection 27 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM779	Connection 28 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM780	Connection 29 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM781	Connection 30 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM782	Connection 31 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM783	Connection 31 via COM1 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM784	The data has been received via COM 1 connection 1 for data exchange.	○	OFF	–	–	N	R	OFF
*SM785	The data has been received via COM 1 connection 2 for data exchange.	○	OFF	–	–	N	R	OFF
*SM786	The data has been received via COM 1 connection 3 for data exchange.	○	OFF	–	–	N	R	OFF
*SM787	The data has been received via COM 1 connection 4 for data exchange.	○	OFF	–	–	N	R	OFF
*SM788	The data has been received via COM 1 connection 5 for data exchange.	○	OFF	–	–	N	R	OFF
*SM789	The data has been received via COM 1 connection 6 for data exchange.	○	OFF	–	–	N	R	OFF
*SM790	The data has been received via COM 1 connection 7 for data exchange.	○	OFF	–	–	N	R	OFF
*SM791	The data has been received via COM 1 connection 8 for data exchange.	○	OFF	–	–	N	R	OFF
*SM792	The data has been received via COM 1 connection 9 for data exchange.	○	OFF	–	–	N	R	OFF
*SM793	The data has been received via COM 1 connection 10 for data exchange.	○	OFF	–	–	N	R	OFF
*SM794	The data has been received via COM 1 connection 11 for data exchange.	○	OFF	–	–	N	R	OFF
*SM795	The data has been received via COM 1 connection 12 for data exchange.	○	OFF	–	–	N	R	OFF
*SM796	The data has been received via COM 1 connection 13 for data exchange.	○	OFF	–	–	N	R	OFF
*SM797	The data has been received via COM 1 connection 14 for data exchange.	○	OFF	–	–	N	R	OFF
*SM798	The data has been received via COM 1 connection 15 for data exchange.	○	OFF	–	–	N	R	OFF
*SM799	The data has been received via COM 1 connection 16 for data exchange.	○	OFF	–	–	N	R	OFF
*SM800	The data has been received via COM 1 connection 17 for data exchange.	○	OFF	–	–	N	R	OFF
*SM801	The data has been received via COM 1 connection 18 for	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	data exchange.							
*SM802	The data has been received via COM 1 connection 19 for data exchange.	○	OFF	–	–	N	R	OFF
*SM803	The data has been received via COM 1 connection 20 for data exchange.	○	OFF	–	–	N	R	OFF
*SM804	The data has been received via COM 1 connection 21 for data exchange.	○	OFF	–	–	N	R	OFF
*SM805	The data has been received via COM 1 connection 22 for data exchange.	○	OFF	–	–	N	R	OFF
*SM806	The data has been received via COM 1 connection 23 for data exchange.	○	OFF	–	–	N	R	OFF
*SM807	The data has been received via COM 1 connection 24 for data exchange.	○	OFF	–	–	N	R	OFF
*SM808	The data has been received via COM 1 connection 25 for data exchange.	○	OFF	–	–	N	R	OFF
*SM809	The data has been received via COM 1 connection 26 for data exchange.	○	OFF	–	–	N	R	OFF
*SM810	The data has been received via COM 1 connection 27 for data exchange.	○	OFF	–	–	N	R	OFF
*SM811	The data has been received via COM 1 connection 28 for data exchange.	○	OFF	–	–	N	R	OFF
*SM812	The data has been received via COM 1 connection 29 for data exchange.	○	OFF	–	–	N	R	OFF
*SM813	The data has been received via COM 1 connection 30 for data exchange.	○	OFF	–	–	N	R	OFF
*SM814	The data has been received via COM 1 connection 31 for data exchange.	○	OFF	–	–	N	R	OFF
*SM815	The data has been received via COM 1 connection 32 for data exchange.	○	OFF	–	–	N	R	OFF
*SM816	An error occurs in the COM1 connection 1 for data exchange	○	OFF	–	–	N	R	OFF
*SM817	An error occurs in the COM1 connection 2 for data exchange	○	OFF	–	–	N	R	OFF
*SM818	An error occurs in the COM1 connection 3 for data exchange	○	OFF	–	–	N	R	OFF
*SM819	An error occurs in the COM1 connection 4 for data exchange	○	OFF	–	–	N	R	OFF
*SM820	An error occurs in the COM1 connection 5 for data exchange	○	OFF	–	–	N	R	OFF
*SM821	An error occurs in the COM1 connection 6 for data exchange	○	OFF	–	–	N	R	OFF
*SM822	An error occurs in the COM1 connection 7 for data exchange	○	OFF	–	–	N	R	OFF
*SM823	An error occurs in the COM1 connection 8 for data exchange	○	OFF	–	–	N	R	OFF
*SM824	An error occurs in the COM1 connection 9 for data exchange	○	OFF	–	–	N	R	OFF
*SM825	An error occurs in the COM1 connection 10 for data exchange	○	OFF	–	–	N	R	OFF
*SM826	An error occurs in the COM1 connection 11 for data exchange	○	OFF	–	–	N	R	OFF
*SM827	An error occurs in the COM1 connection 12 for data exchange	○	OFF	–	–	N	R	OFF
*SM828	An error occurs in the COM1 connection 13 for data exchange	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM829	An error occurs in the COM1 connection 14 for data exchange	○	OFF	–	–	N	R	OFF
*SM830	An error occurs in the COM1 connection 15 for data exchange	○	OFF	–	–	N	R	OFF
*SM831	An error occurs in the COM1 connection 16 for data exchange	○	OFF	–	–	N	R	OFF
*SM832	An error occurs in the COM1 connection 17 for data exchange	○	OFF	–	–	N	R	OFF
*SM833	An error occurs in the COM1 connection 18 for data exchange	○	OFF	–	–	N	R	OFF
*SM834	An error occurs in the COM1 connection 19 for data exchange	○	OFF	–	–	N	R	OFF
*SM835	An error occurs in the COM1 connection 20 for data exchange	○	OFF	–	–	N	R	OFF
*SM836	An error occurs in the COM1 connection 21 for data exchange	○	OFF	–	–	N	R	OFF
*SM837	An error occurs in the COM1 connection 22 for data exchange	○	OFF	–	–	N	R	OFF
*SM838	An error occurs in the COM1 connection 23 for data exchange	○	OFF	–	–	N	R	OFF
*SM839	An error occurs in the COM1 connection 24 for data exchange	○	OFF	–	–	N	R	OFF
*SM840	An error occurs in the COM1 connection 25 for data exchange	○	OFF	–	–	N	R	OFF
*SM841	An error occurs in the COM1 connection 26 for data exchange	○	OFF	–	–	N	R	OFF
*SM842	An error occurs in the COM1 connection 27 for data exchange	○	OFF	–	–	N	R	OFF
*SM843	An error occurs in the COM1 connection 28 for data exchange	○	OFF	–	–	N	R	OFF
*SM844	An error occurs in the COM1 connection 29 for data exchange	○	OFF	–	–	N	R	OFF
*SM845	An error occurs in the COM1 connection 30 for data exchange	○	OFF	–	–	N	R	OFF
*SM846	An error occurs in the COM1 connection 31 for data exchange	○	OFF	–	–	N	R	OFF
*SM847	An error occurs in the COM1 connection 32 for data exchange	○	OFF	–	–	N	R	OFF
SM861	An error occurs in the initialization of the data exchange via COM2.	○	OFF	–	–	N	R/W	OFF
*SM862	Data exchange via COM2 has been enabled by ISPSoft.	○	OFF	–	–	H	R/W	OFF
*SM864	Connection 1 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM865	Connection 2 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM866	Connection 3 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM867	Connection 4 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM868	Connection 5 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM869	Connection 6 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM870	Connection 7 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM871	Connection 8 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM872	Connection 9 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM873	Connection 10 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM874	Connection 11 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM875	Connection 12 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM876	Connection 13 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM877	Connection 14 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM878	Connection 15 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM879	Connection 16 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM880	Connection 17 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM881	Connection 18 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM882	Connection 19 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM883	Connection 20 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM884	Connection 21 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM885	Connection 22 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM886	Connection 23 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM887	Connection 24 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM888	Connection 25 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM889	Connection 26 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM890	Connection 27 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM891	Connection 28 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM892	Connection 29 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM893	Connection 30 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM894	Connection 31 via COM2 for data exchange has been	○	OFF	–	–	H	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	started.							
*SM895	Connection 32 via COM2 for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM896	The data has been received via COM 2 connection 1 for data exchange.	○	OFF	–	–	N	R	OFF
*SM897	The data has been received via COM 2 connection 2 for data exchange.	○	OFF	–	–	N	R	OFF
*SM898	The data has been received via COM 2 connection 3 for data exchange.	○	OFF	–	–	N	R	OFF
*SM899	The data has been received via COM 2 connection 4 for data exchange.	○	OFF	–	–	N	R	OFF
*SM900	The data has been received via COM 2 connection 5 for data exchange.	○	OFF	–	–	N	R	OFF
*SM901	The data has been received via COM 2 connection 6 for data exchange.	○	OFF	–	–	N	R	OFF
*SM902	The data has been received via COM 2 connection 7 for data exchange.	○	OFF	–	–	N	R	OFF
*SM903	The data has been received via COM 2 connection 8 for data exchange.	○	OFF	–	–	N	R	OFF
*SM904	The data has been received via COM 2 connection 9 for data exchange.	○	OFF	–	–	N	R	OFF
*SM905	The data has been received via COM 2 connection 10 for data exchange.	○	OFF	–	–	N	R	OFF
*SM906	The data has been received via COM 2 connection 11 for data exchange.	○	OFF	–	–	N	R	OFF
*SM907	The data has been received via COM 2 connection 12 for data exchange.	○	OFF	–	–	N	R	OFF
*SM908	The data has been received via COM 2 connection 13 for data exchange.	○	OFF	–	–	N	R	OFF
*SM909	The data has been received via COM 2 connection 14 for data exchange.	○	OFF	–	–	N	R	OFF
*SM910	The data has been received via COM 2 connection 15 for data exchange.	○	OFF	–	–	N	R	OFF
*SM911	The data has been received via COM 2 connection 16 for data exchange.	○	OFF	–	–	N	R	OFF
*SM912	The data has been received via COM 2 connection 17 for data exchange.	○	OFF	–	–	N	R	OFF
*SM913	The data has been received via COM 2 connection 18 for data exchange.	○	OFF	–	–	N	R	OFF
*SM914	The data has been received via COM 2 connection 19 for data exchange.	○	OFF	–	–	N	R	OFF
*SM915	The data has been received via COM 2 connection 20 for data exchange.	○	OFF	–	–	N	R	OFF
*SM916	The data has been received via COM 2 connection 21 for data exchange.	○	OFF	–	–	N	R	OFF
*SM917	The data has been received via COM 2 connection 22 for data exchange.	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM918	The data has been received via COM 2 connection 23 for data exchange.	○	OFF	–	–	N	R	OFF
*SM919	The data has been received via COM 2 connection 24 for data exchange.	○	OFF	–	–	N	R	OFF
*SM920	The data has been received via COM 2 connection 25 for data exchange.	○	OFF	–	–	N	R	OFF
*SM921	The data has been received via COM 2 connection 26 for data exchange.	○	OFF	–	–	N	R	OFF
*SM922	The data has been received via COM 2 connection 27 for data exchange.	○	OFF	–	–	N	R	OFF
*SM923	The data has been received via COM 2 connection 28 for data exchange.	○	OFF	–	–	N	R	OFF
*SM924	The data has been received via COM 2 connection 29 for data exchange.	○	OFF	–	–	N	R	OFF
*SM925	The data has been received via COM 2 connection 30 for data exchange.	○	OFF	–	–	N	R	OFF
*SM926	The data has been received via COM 2 connection 31 for data exchange.	○	OFF	–	–	N	R	OFF
*SM927	The data has been received via COM 2 connection 32 for data exchange.	○	OFF	–	–	N	R	OFF
*SM928	An error occurs in the COM2 connection 1 for data exchange	○	OFF	–	–	N	R	OFF
*SM929	An error occurs in the COM2 connection 2 for data exchange	○	OFF	–	–	N	R	OFF
*SM930	An error occurs in the COM2 connection 3 for data exchange	○	OFF	–	–	N	R	OFF
*SM931	An error occurs in the COM2 connection 4 for data exchange	○	OFF	–	–	N	R	OFF
*SM932	An error occurs in the COM2 connection 5 for data exchange	○	OFF	–	–	N	R	OFF
*SM933	An error occurs in the COM2 connection 6 for data exchange	○	OFF	–	–	N	R	OFF
*SM934	An error occurs in the COM2 connection 7 for data exchange	○	OFF	–	–	N	R	OFF
*SM935	An error occurs in the COM2 connection 8 for data exchange	○	OFF	–	–	N	R	OFF
*SM936	An error occurs in the COM2 connection 9 for data exchange	○	OFF	–	–	N	R	OFF
*SM937	An error occurs in the COM2 connection 10 for data exchange	○	OFF	–	–	N	R	OFF
*SM938	An error occurs in the COM2 connection 11 for data exchange	○	OFF	–	–	N	R	OFF
*SM939	An error occurs in the COM2 connection 12 for data exchange	○	OFF	–	–	N	R	OFF
*SM940	An error occurs in the COM2 connection 13 for data exchange	○	OFF	–	–	N	R	OFF
*SM941	An error occurs in the COM2 connection 14 for data exchange	○	OFF	–	–	N	R	OFF
*SM942	An error occurs in the COM2 connection 15 for data exchange	○	OFF	–	–	N	R	OFF
*SM943	An error occurs in the COM2 connection 16 for data exchange	○	OFF	–	–	N	R	OFF
*SM944	An error occurs in the COM2 connection 17 for data exchange	○	OFF	–	–	N	R	OFF
*SM945	An error occurs in the COM2 connection 18 for data	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	exchange							
*SM946	An error occurs in the COM2 connection 19 for data exchange	○	OFF	–	–	N	R	OFF
*SM947	An error occurs in the COM2 connection 20 for data exchange	○	OFF	–	–	N	R	OFF
*SM948	An error occurs in the COM2 connection 21 for data exchange	○	OFF	–	–	N	R	OFF
*SM949	An error occurs in the COM2 connection 22 for data exchange	○	OFF	–	–	N	R	OFF
*SM950	An error occurs in the COM2 connection 23 for data exchange	○	OFF	–	–	N	R	OFF
*SM951	An error occurs in the COM2 connection 24 for data exchange	○	OFF	–	–	N	R	OFF
*SM952	An error occurs in the COM2 connection 25 for data exchange	○	OFF	–	–	N	R	OFF
*SM953	An error occurs in the COM2 connection 26 for data exchange	○	OFF	–	–	N	R	OFF
*SM954	An error occurs in the COM2 connection 27 for data exchange	○	OFF	–	–	N	R	OFF
*SM955	An error occurs in the COM2 connection 28 for data exchange	○	OFF	–	–	N	R	OFF
*SM956	An error occurs in the COM2 connection 29 for data exchange	○	OFF	–	–	N	R	OFF
*SM957	An error occurs in the COM2 connection 30 for data exchange	○	OFF	–	–	N	R	OFF
*SM958	An error occurs in the COM2 connection 31 for data exchange	○	OFF	–	–	N	R	OFF
*SM959	An error occurs in the COM2 connection 32 for data exchange	○	OFF	–	–	N	R	OFF
SM1000	It is the Ethernet setting flag. When SM1000 is ON, the data in SR1000~SR1006 is written into the flash memory.	○	OFF	–	–	N	R/W	OFF
SM1001	The state of the Ethernet connectivity	○	OFF	–	–	N	R	OFF
*SM1090	The TCP connection is busy.	○	OFF	–	–	N	R	OFF
*SM1091	The UDP connection is busy.	○	OFF	–	–	N	R	OFF
SM1100	The network cable is not connected	○	OFF	–	–	N	R	OFF
*SM1106	Basic management–Ethernet connection error	○	OFF	–	–	N	R	OFF
*SM1107	Basic management of Ethernet–Basic setting error	○	OFF	–	–	N	R	OFF
*SM1109	Basic management of the TCP/UDP socket–The local port is already used.	○	OFF	–	–	N	R	OFF
SM1111	EtherNet/IP data exchange flag	○	OFF	–	–	N	R	OFF
*SM1113	Email service error	○	OFF	–	–	N	R	OFF
*SM1116	It is the switch of trigger 1 in the email.	○	OFF	–	–	N	R	OFF
*SM1117	Trigger 1 in the email	○	OFF	–	–	N	R	OFF
*SM1119	When trigger 1 is triggered and the email has been sent successfully; SM1119 is ON.	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM1120	When trigger 1 is triggered but the email cannot be sent due to email content error; SM1120 is ON.	○	OFF	–	–	N	R	OFF
*SM1122	When trigger 1 is triggered and there is an SMTP server response timeout, SM1122 is ON.	○	OFF	–	–	N	R	OFF
*SM1123	When trigger 1 is triggered and there is an SMTP server response error, SM1123 is ON.	○	OFF	–	–	N	R	OFF
*SM1124	When trigger 1 is triggered and the size of the attachment exceeds the limit, SM1124 is ON.	○	OFF	–	–	N	R	OFF
*SM1125	When trigger 1 is triggered and the attachment is not found, SM1125 is ON.	○	OFF	–	–	N	R	OFF
*SM1126	It is the switch of trigger 2 in the email.	○	OFF	–	–	N	R	OFF
*SM1127	Trigger 2 in the email	○	OFF	–	–	N	R	OFF
*SM1129	When trigger 2 is triggered and the email has been sent successfully; SM1129 is ON.	○	OFF	–	–	N	R	OFF
*SM1130	When trigger 2 is triggered but the email cannot be sent due to email content error; SM1130 is ON.	○	OFF	–	–	N	R	OFF
*SM1132	When trigger 2 is triggered and there is an SMTP server response timeout, SM1132 is ON.	○	OFF	–	–	N	R	OFF
*SM1133	When trigger 2 is triggered and there is an SMTP server response error, SM1133 is ON.	○	OFF	–	–	N	R	OFF
*SM1134	When trigger 2 is triggered and the size of the attachment exceeds the limit, SM1134 is ON.	○	OFF	–	–	N	R	OFF
*SM1135	When trigger 2 is triggered and the attachment is not found, SM1135 is ON.	○	OFF	–	–	N	R	OFF
*SM1136	It is the switch of trigger 3 in the email.	○	OFF	–	–	N	R	OFF
*SM1137	Trigger 3 in the email	○	OFF	–	–	N	R	OFF
*SM1139	When trigger 3 is triggered and the email has been sent successfully; SM1139 is ON.	○	OFF	–	–	N	R	OFF
*SM1140	When trigger 3 is triggered but the email cannot be sent due to email content error; SM1140 is ON.	○	OFF	–	–	N	R	OFF
*SM1142	When trigger 3 is triggered and there is an SMTP server response timeout, SM1142 is ON.	○	OFF	–	–	N	R	OFF
*SM1143	When trigger 3 is triggered and there is an SMTP server response error, SM1143 is ON.	○	OFF	–	–	N	R	OFF
*SM1144	When trigger 3 is triggered and the size of the attachment exceeds the limit, SM1144 is ON.	○	OFF	–	–	N	R	OFF
*SM1145	When trigger 3 is triggered and the attachment is not found, SM1145 is ON.	○	OFF	–	–	N	R	OFF
*SM1146	It is the switch of trigger 4 in the email.	○	OFF	–	–	N	R	OFF
*SM1147	Trigger 4 in the email	○	OFF	–	–	N	R	OFF
*SM1149	When trigger 4 is triggered and the email has been sent successfully; SM1149 is ON.	○	OFF	–	–	N	R	OFF
*SM1150	When trigger 4 is triggered but the email cannot be sent due to email content error; SM1150 is ON.	○	OFF	–	–	N	R	OFF
*SM1152	When trigger 4 is triggered and there is an SMTP server	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	response timeout, SM1152 is ON.							
*SM1153	When trigger 4 is triggered and there is an SMTP server response error, SM1153 is ON.	○	OFF	–	–	N	R	OFF
*SM1154	When trigger 4 is triggered and the size of the attachment exceeds the limit, SM1154 is ON.	○	OFF	–	–	N	R	OFF
*SM1155	When trigger 4 is triggered and the attachment is not found, SM1155 is ON.	○	OFF	–	–	N	R	OFF
*SM1166	An error occurs in the initialization of the data exchange via Ethernet port.	○	–	–	–	N	R	OFF
*SM1167	Data exchange via Ethernet port has been enabled by ISPSoft.	○	OFF	–	–	H	R/W	OFF
*SM1168	Connection 1 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1169	Connection 2 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1170	Connection 3 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1171	Connection 4 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1172	Connection 5 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1173	Connection 6 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1174	Connection 7 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1175	Connection 8 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1176	Connection 9 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1177	Connection 10 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1178	Connection 11 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1179	Connection 12 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1180	Connection 13 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1181	Connection 14 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1182	Connection 15 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1183	Connection 16 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1184	Connection 17 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1185	Connection 18 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM1186	Connection 19 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1187	Connection 20 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1188	Connection 21 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1189	Connection 22 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1190	Connection 23 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1191	Connection 24 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1192	Connection 25 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1193	Connection 26 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1194	Connection 27 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1195	Connection 28 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1196	Connection 29 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1197	Connection 30 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1198	Connection 31 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1199	Connection 32 via Ethernet port for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
*SM1200	The data has been received via Ethernet port connection 1 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1201	The data has been received via Ethernet port connection 2 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1202	The data has been received via Ethernet port connection 3 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1203	The data has been received via Ethernet port connection 4 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1204	The data has been received via Ethernet port connection 5 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1205	The data has been received via Ethernet port connection 6 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1206	The data has been received via Ethernet port connection 7 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1207	The data has been received via Ethernet port connection 8 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1208	The data has been received via Ethernet port connection 9 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1209	The data has been received via Ethernet port connection 10	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	for data exchange.							
*SM1210	The data has been received via Ethernet port connection 11 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1211	The data has been received via Ethernet port connection 12 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1212	The data has been received via Ethernet port connection 13 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1213	The data has been received via Ethernet port connection 14 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1214	The data has been received via Ethernet port connection 15 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1215	The data has been received via Ethernet port connection 16 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1216	The data has been received via Ethernet port connection 17 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1217	The data has been received via Ethernet port connection 18 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1218	The data has been received via Ethernet port connection 19 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1219	The data has been received via Ethernet port connection 20 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1220	The data has been received via Ethernet port connection 21 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1221	The data has been received via Ethernet port connection 22 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1222	The data has been received via Ethernet port connection 23 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1223	The data has been received via Ethernet port connection 24 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1224	The data has been received via Ethernet port connection 25 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1225	The data has been received via Ethernet port connection 26 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1226	The data has been received via Ethernet port connection 27 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1227	The data has been received via Ethernet port connection 28 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1228	The data has been received via Ethernet port connection 29 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1229	The data has been received via Ethernet port connection 30 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1230	The data has been received via Ethernet port connection 31 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1231	The data has been received via Ethernet port connection 32 for data exchange.	○	OFF	–	–	N	R	OFF
*SM1232	An error occurs in the Ethernet port connection 1 for data exchange	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM1233	An error occurs in the Ethernet port connection 2 for data exchange	○	OFF	–	–	N	R	OFF
*SM1234	An error occurs in the Ethernet port connection 3 for data exchange	○	OFF	–	–	N	R	OFF
*SM1235	An error occurs in the Ethernet port connection 4 for data exchange	○	OFF	–	–	N	R	OFF
*SM1236	An error occurs in the Ethernet port connection 5 for data exchange	○	OFF	–	–	N	R	OFF
*SM1237	An error occurs in the Ethernet port connection 6 for data exchange	○	OFF	–	–	N	R	OFF
*SM1238	An error occurs in the Ethernet port connection 7 for data exchange	○	OFF	–	–	N	R	OFF
*SM1239	An error occurs in the Ethernet port connection 8 for data exchange	○	OFF	–	–	N	R	OFF
*SM1240	An error occurs in the Ethernet port connection 9 for data exchange	○	OFF	–	–	N	R	OFF
*SM1241	An error occurs in the Ethernet port connection 10 for data exchange	○	OFF	–	–	N	R	OFF
*SM1242	An error occurs in the Ethernet port connection 11 for data exchange	○	OFF	–	–	N	R	OFF
*SM1243	An error occurs in the Ethernet port connection 12 for data exchange	○	OFF	–	–	N	R	OFF
*SM1244	An error occurs in the Ethernet port connection 13 for data exchange	○	OFF	–	–	N	R	OFF
*SM1245	An error occurs in the Ethernet port connection 14 for data exchange	○	OFF	–	–	N	R	OFF
*SM1246	An error occurs in the Ethernet port connection 15 for data exchange	○	OFF	–	–	N	R	OFF
*SM1247	An error occurs in the Ethernet port connection 16 for data exchange	○	OFF	–	–	N	R	OFF
*SM1248	An error occurs in the Ethernet port connection 17 for data exchange	○	OFF	–	–	N	R	OFF
*SM1249	An error occurs in the Ethernet port connection 18 for data exchange	○	OFF	–	–	N	R	OFF
*SM1250	An error occurs in the Ethernet port connection 19 for data exchange	○	OFF	–	–	N	R	OFF
*SM1251	An error occurs in the Ethernet port connection 20 for data exchange	○	OFF	–	–	N	R	OFF
*SM1252	An error occurs in the Ethernet port connection 21 for data exchange	○	OFF	–	–	N	R	OFF
*SM1253	An error occurs in the Ethernet port connection 22 for data exchange	○	OFF	–	–	N	R	OFF
*SM1254	An error occurs in the Ethernet port connection 23 for data exchange	○	OFF	–	–	N	R	OFF
*SM1255	An error occurs in the Ethernet port connection 24 for data exchange	○	OFF	–	–	N	R	OFF
*SM1256	An error occurs in the Ethernet port connection 25 for data	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	exchange							
*SM1257	An error occurs in the Ethernet port connection 26 for data exchange	○	OFF	–	–	N	R	OFF
*SM1258	An error occurs in the Ethernet port connection 27 for data exchange	○	OFF	–	–	N	R	OFF
*SM1259	An error occurs in the Ethernet port connection 28 for data exchange	○	OFF	–	–	N	R	OFF
*SM1260	An error occurs in the Ethernet port connection 29 for data exchange	○	OFF	–	–	N	R	OFF
*SM1261	An error occurs in the Ethernet port connection 30 for data exchange	○	OFF	–	–	N	R	OFF
*SM1262	An error occurs in the Ethernet port connection 31 for data exchange	○	OFF	–	–	N	R	OFF
*SM1263	An error occurs in the Ethernet port connection 32 for data exchange	○	OFF	–	–	N	R	OFF
SM1269	Socket configuration error	○	OFF	–	–	N	R/W	OFF
SM1270	TCP socket 1–The connection is successful.	○	OFF	–	–	N	R	OFF
SM1271	TCP socket 1–The data has been received.	○	OFF	–	–	N	R	OFF
SM1272	TCP socket 1–The data has been sent.	○	OFF	–	–	N	R	OFF
SM1273	TCP socket 1–The connection is being started.	○	OFF	–	–	N	R	OFF
SM1274	TCP socket 1–The connection is being closed.	○	ON	–	–	Y	R	ON
SM1275	TCP socket 1–The data is being sent.	○	OFF	–	–	N	R	OFF
SM1277	TCP socket 1–Error flag	○	OFF	–	–	N	R	OFF
SM1278	TCP socket 2–The connection is successful.	○	OFF	–	–	N	R	OFF
SM1279	TCP socket 2–The data has been received.	○	OFF	–	–	N	R	OFF
SM1280	TCP socket 2–The data has been sent.	○	OFF	–	–	N	R	OFF
SM1281	TCP socket 2–The connection is being started.	○	OFF	–	–	N	R	OFF
SM1282	TCP socket 2–The connection is being closed.	○	ON	–	–	Y	R	ON
SM1283	TCP socket 2–The data is being sent.	○	OFF	–	–	N	R	OFF
SM1285	TCP socket 2–Error flag	○	OFF	–	–	N	R	OFF
SM1286	TCP socket 3The connection is successful.	○	OFF	–	–	N	R	OFF
SM1287	TCP socket 3–The data has been received.	○	OFF	–	–	N	R	OFF
SM1288	TCP socket 3–The data has been sent.	○	OFF	–	–	N	R	OFF
SM1289	TCP socket 3–The connection is being started.	○	OFF	–	–	N	R	OFF
SM1290	TCP socket 3–The connection is being closed.	○	ON	–	–	Y	R	ON
SM1291	TCP socket 3–The data is being sent.	○	OFF	–	–	N	R	OFF
SM1293	TCP socket 3–Error flag	○	OFF	–	–	N	R	OFF
SM1294	TCP socket 4–The connection is successful.	○	OFF	–	–	N	R	OFF
SM1295	TCP socket 4–The data has been received.	○	OFF	–	–	N	R	OFF
SM1296	TCP socket 4–The data has been sent.	○	OFF	–	–	N	R	OFF
SM1297	TCP socket 4–The connection is being started.	○	OFF	–	–	N	R	OFF
SM1298	TCP socket 4–The connection is being closed.	○	ON	–	–	Y	R	ON
SM1299	TCP socket 4–The data is being sent.	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM1301	TCP socket 4—Error flag	○	OFF	–	–	N	R	OFF
SM1334	UDP socket 1—The connection has been started.	○	OFF	–	–	N	R	OFF
SM1335	UDP socket 1—The data has been received.	○	OFF	–	–	N	R	OFF
SM1336	UDP socket 1—The data has been sent.	○	OFF	–	–	N	R	OFF
SM1337	UDP socket 1—The connection has been closed.	○	ON	–	–	Y	R	ON
SM1338	UDP socket 1—Error flag	○	OFF	–	–	N	R	OFF
SM1339	UDP socket 2—The connection has been started.	○	OFF	–	–	N	R	OFF
SM1340	UDP socket 2—The data has been received.	○	OFF	–	–	N	R	OFF
SM1341	UDP socket 2—The data has been sent.	○	OFF	–	–	N	R	OFF
SM1342	UDP socket 2—The connection has been closed.	○	ON	–	–	Y	R	ON
SM1343	UDP socket 2—Error flag	○	OFF	–	–	N	R	OFF
SM1344	UDP socket 3—The connection has been started.	○	OFF	–	–	N	R	OFF
SM1345	UDP socket 3—The data has been received.	○	OFF	–	–	N	R	OFF
SM1346	UDP socket 3—The data has been sent.	○	OFF	–	–	N	R	OFF
SM1347	UDP socket 3—The connection has been closed.	○	ON	–	–	Y	R	ON
SM1348	UDP socket 3—Error flag	○	OFF	–	–	N	R	OFF
SM1349	UDP socket 4—The connection has been started.	○	OFF	–	–	N	R	OFF
SM1350	UDP socket 4—The data has been received.	○	OFF	–	–	N	R	OFF
SM1351	UDP socket 4—The data has been sent.	○	OFF	–	–	N	R	OFF
SM1352	UDP socket 4—The connection has been closed.	○	ON	–	–	Y	R	ON
SM1353	UDP socket 4—Error flag	○	OFF	–	–	N	R	OFF
SM1375	Data exchange via EtherNet/IP has been started.	○	OFF	–	–	H	R/W	OFF
SM1376	Connection 1 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1377	Connection 2 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1378	Connection 3 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1379	Connection 4 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1380	Connection 5 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1381	Connection 6 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1382	Connection 7 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1383	Connection 8 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1384	Connection 9 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1385	Connection 10 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1386	Connection 11 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM1387	Connection 12 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1388	Connection 13 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1389	Connection 14 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1390	Connection 15 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1391	Connection 16 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1392	Connection 17 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1393	Connection 18 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1394	Connection 19 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1395	Connection 20 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1396	Connection 21 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1397	Connection 22 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1398	Connection 23 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1399	Connection 24 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1400	Connection 25 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1401	Connection 26 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1402	Connection 27 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1403	Connection 28 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1404	Connection 29 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1405	Connection 30 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1406	Connection 31 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1407	Connection 32 via EtherNet/IP for data exchange has been started.	○	OFF	–	–	H	R/W	OFF
SM1408	An error occurs in the EtherNet/IP connection 1 for data exchange	○	OFF	–	–	N	R	OFF
SM1409	An error occurs in the EtherNet/IP connection 2 for data exchange	○	OFF	–	–	N	R	OFF
SM1410	An error occurs in the EtherNet/IP connection 3 for data	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	exchange							
SM1411	An error occurs in the EtherNet/IP connection 4 for data exchange	○	OFF	–	–	N	R	OFF
SM1412	An error occurs in the EtherNet/IP connection 5 for data exchange	○	OFF	–	–	N	R	OFF
SM1413	An error occurs in the EtherNet/IP connection 6 for data exchange	○	OFF	–	–	N	R	OFF
SM1414	An error occurs in the EtherNet/IP connection 7 for data exchange	○	OFF	–	–	N	R	OFF
SM1415	An error occurs in the EtherNet/IP connection 8 for data exchange	○	OFF	–	–	N	R	OFF
SM1416	An error occurs in the EtherNet/IP connection 9 for data exchange	○	OFF	–	–	N	R	OFF
SM1417	An error occurs in the EtherNet/IP connection 10 for data exchange	○	OFF	–	–	N	R	OFF
SM1418	An error occurs in the EtherNet/IP connection 11 for data exchange	○	OFF	–	–	N	R	OFF
SM1419	An error occurs in the EtherNet/IP connection 12 for data exchange	○	OFF	–	–	N	R	OFF
SM1420	An error occurs in the EtherNet/IP connection 13 for data exchange	○	OFF	–	–	N	R	OFF
SM1421	An error occurs in the EtherNet/IP connection 14 for data exchange	○	OFF	–	–	N	R	OFF
SM1422	An error occurs in the EtherNet/IP connection 15 for data exchange	○	OFF	–	–	N	R	OFF
SM1423	An error occurs in the EtherNet/IP connection 16 for data exchange	○	OFF	–	–	N	R	OFF
SM1424	An error occurs in the EtherNet/IP connection 17 for data exchange	○	OFF	–	–	N	R	OFF
SM1425	An error occurs in the EtherNet/IP connection 18 for data exchange	○	OFF	–	–	N	R	OFF
SM1426	An error occurs in the EtherNet/IP connection 19 for data exchange	○	OFF	–	–	N	R	OFF
SM1427	An error occurs in the EtherNet/IP connection 20 for data exchange	○	OFF	–	–	N	R	OFF
SM1428	An error occurs in the EtherNet/IP connection 21 for data exchange	○	OFF	–	–	N	R	OFF
SM1429	An error occurs in the EtherNet/IP connection 22 for data exchange	○	OFF	–	–	N	R	OFF
SM1430	An error occurs in the EtherNet/IP connection 23 for data exchange	○	OFF	–	–	N	R	OFF
SM1431	An error occurs in the EtherNet/IP connection 24 for data exchange	○	OFF	–	–	N	R	OFF
SM1432	An error occurs in the EtherNet/IP connection 25 for data exchange	○	OFF	–	–	N	R	OFF
SM1433	An error occurs in the EtherNet/IP connection 26 for data exchange	○	OFF	–	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM1434	An error occurs in the EtherNet/IP connection 27 for data exchange	○	OFF	–	–	N	R	OFF
SM1435	An error occurs in the EtherNet/IP connection 28 for data exchange	○	OFF	–	–	N	R	OFF
SM1436	An error occurs in the EtherNet/IP connection 29 for data exchange	○	OFF	–	–	N	R	OFF
SM1437	An error occurs in the EtherNet/IP connection 30 for data exchange	○	OFF	–	–	N	R	OFF
SM1438	An error occurs in the EtherNet/IP connection 31 for data exchange	○	OFF	–	–	N	R	OFF
SM1439	An error occurs in the EtherNet/IP connection 32 for data exchange	○	OFF	–	–	N	R	OFF
SM1440	An error occurs in the EtherNet/IP I/O connection 1	○	OFF	-	-	R	否	OFF
SM1441	An error occurs in the EtherNet/IP I/O connection 2	○	OFF	-	-	R	否	OFF
SM1442	An error occurs in the EtherNet/IP I/O connection 3	○	OFF	-	-	R	否	OFF
SM1443	An error occurs in the EtherNet/IP I/O connection 4	○	OFF	-	-	R	否	OFF
SM1444	An error occurs in the EtherNet/IP I/O connection 5	○	OFF	-	-	R	否	OFF
SM1445	An error occurs in the EtherNet/IP I/O connection 6	○	OFF	-	-	R	否	OFF
SM1446	An error occurs in the EtherNet/IP I/O connection 7	○	OFF	-	-	R	否	OFF
SM1447	An error occurs in the EtherNet/IP I/O connection 8	○	OFF	-	-	R	否	OFF
SM1631	Positioning is completed for the axis 1 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1632	Positioning is completed for the axis 2 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1633	Positioning is completed for the axis 3 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1634	Positioning is completed for the axis 4 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1635	Positioning is completed for the axis 5 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1636	Positioning is completed for the axis 6 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1637	Positioning is completed for the axis 7 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1638	Positioning is completed for the axis 8 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1641	Communication stops for the axis 1 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1642	Communication stops for the axis 2 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1643	Communication stops for the axis 3 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1644	Communication stops for the axis 4 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1645	Communication stops for the axis 5 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1646	Communication stops for the axis 6 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1647	Communication stops for the axis 7 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1648	Communication stops for the axis 8 of ASD-A2 CAN	○	OFF	OFF	–	N	R/W	OFF
SM1651	Servo is ON for the axis 1 of ASD-A2 CAN	○	OFF	OFF	–	N	R	OFF
SM1652	Servo is ON for the axis 2 of ASD-A2 CAN	○	OFF	OFF	–	N	R	OFF
SM1653	Servo is ON for the axis 3 of ASD-A2 CAN	○	OFF	OFF	–	N	R	OFF
SM1654	Servo is ON for the axis 4 of ASD-A2 CAN	○	OFF	OFF	–	N	R	OFF

SM	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM1655	Servo is ON for the axis 5 of ASD-A2 CAN	○	OFF	OFF	–	N	R	OFF
SM1656	Servo is ON for the axis 6 of ASD-A2 CAN	○	OFF	OFF	–	N	R	OFF
SM1657	Servo is ON for the axis 7 of ASD-A2 CAN	○	OFF	OFF	–	N	R	OFF
SM1658	Servo is ON for the axis 8 of ASD-A2 CAN	○	OFF	OFF	–	N	R	OFF
SM1661	The function of going back and forth is enabled for the axis 1 of ASD-A2 CAN.	○	OFF	OFF	–	N	R/W	OFF
SM1662	The function of going back and forth is enabled for the axis 2 of ASD-A2 CAN.	○	OFF	OFF	–	N	R/W	OFF
SM1663	The function of going back and forth is enabled for the axis 3 of ASD-A2 CAN.	○	OFF	OFF	–	N	R/W	OFF
SM1664	The function of going back and forth is enabled for the axis 4 of ASD-A2 CAN.	○	OFF	OFF	–	N	R/W	OFF
SM1665	The function of going back and forth is enabled for the axis 5 of ASD-A2 CAN.	○	OFF	OFF	–	N	R/W	OFF
SM1666	The function of going back and forth is enabled for the axis 6 of ASD-A2 CAN.	○	OFF	OFF	–	N	R/W	OFF
SM1667	The function of going back and forth is enabled for the axis 7 of ASD-A2 CAN.	○	OFF	OFF	–	N	R/W	OFF
SM1668	The function of going back and forth is enabled for the axis 8 of ASD-A2 CAN.	○	OFF	OFF	–	N	R/W	OFF
SM1671	The go-back/go-forth direction indication flag for the axis 1 of ASD-A2 CAN.	○	OFF	OFF	–	N	R	OFF
SM1672	The go-back/go-forth direction indication flag for the axis 2 of ASD-A2 CAN.	○	OFF	OFF	–	N	R	OFF
SM1673	The go-back/go-forth direction indication flag for the axis 3 of ASD-A2 CAN.	○	OFF	OFF	–	N	R	OFF
SM1674	The go-back/go-forth direction indication flag for the axis 4 of ASD-A2 CAN.	○	OFF	OFF	–	N	R	OFF
SM1675	The go-back/go-forth direction indication flag for the axis 5 of ASD-A2 CAN.	○	OFF	OFF	–	N	R	OFF
SM1676	The go-back/go-forth direction indication flag for the axis 6 of ASD-A2 CAN.	○	OFF	OFF	–	N	R	OFF
SM1677	The go-back/go-forth direction indication flag for the axis 7 of ASD-A2 CAN.	○	OFF	OFF	–	N	R	OFF
SM1678	The go-back/go-forth direction indication flag for the axis 8 of ASD-A2 CAN.	○	OFF	OFF	–	N	R	OFF
SM1681	Initialization and communication are completed for ASD-A2 CAN.	○	OFF	OFF	–	N	R	OFF
SM1682	The CANopen communication error flag for ASD-A2 CAN	○	OFF	OFF	–	N	R	OFF

*1: For items with a * mark, please refer to the additional remark for Additional Remarks on Special Auxiliary Relays and Special Data Registers for details.

*2: The system will execute according to the parameters set in HWCONFIG, when the power of SM is from OFF to ON, the state is -, and the latched area is marked as N.

*3: The communication card here means AS-F232, AS-F422 and AS-F485.

2.2.8 Refresh Time of Special Auxiliary Relays

Special auxiliary relay	Refresh time
SM0~SM1	The system automatically sets the flag to ON and resets it to OFF. The flag is automatically set to ON when there is an operation error.
SM5	The system automatically sets the flag to ON and resets it to OFF. The flag is automatically set to ON when an error occurred during the program is written in the PLC.
SM6	When power-on, the system checks whether the data in the latched area is lost, if the data is lost, the flag is ON. Users reset it to OFF.
SM7	When power supply (24V) is not sufficient, the flag is ON. Users reset it to OFF.
SM8	The system automatically sets SM8 to ON and resets it to OFF. SM8 is automatically set to ON when there is a watchdog timer error.
SM9	The system automatically sets SM9 to ON and resets it to OFF. SM9 is automatically set to ON when there is a system error.
SM10	The system automatically sets SM10 to ON and resets it to OFF. SM10 is automatically set to ON when there is an I/O bus error.
SM22~SM24	Users set the flag to ON, and the system automatically resets it to OFF. The log is cleared when the flag is ON.
SM25~SM26	When users edit via the ISPSoft, the flag is ON; when users log out of ISPSoft, the flag is automatically set to OFF.
SM28	The system will check if there is anything wrong. When there is something wrong, the flag is ON. Users reset it to OFF.
SM30	The flag is ON automatically when there is an error occurs in the remote module. The system resets it to OFF.
SM34	The flag is ON when wrong password is entered. The system resets it to OFF.
SM36	Users set the flag to ON and the system will save the data to the memory card. After the saving is complete, the system resets it to OFF automatically.
SM76~SM77	Users set the flag to ON and the system will execution the communication task. After the communication task is complete, the system resets it to OFF automatically.
SM78~SM79	The flag is ON while the communication is in process. After the communication is complete, the system resets it to OFF automatically.
SM80~SM81	The flag is ON, after the reception is complete. Users reset it to OFF.
SM82~SM83	The flag is ON, when an error occurs in response. Users reset it to OFF.
SM84~SM85	The flag is ON, when a timeout occurs. Users reset it to OFF.
SM86~SM87	Users set the flag to ON and reset it to OFF. ON: The 8-bit mode OFF: The 16-bit mode
SM90~SM91	Users set the flag to ON. After the communication protocol is changed, the system resets it to OFF.
SM94~SM95	After power-on, the flag is ON/OFF according to the settings in HWCONFIG; users can change this setting.
SM96~SM97	Users set the flag to ON. After the data is sent, the system automatically resets the flag to OFF.
SM98~SM99	The flag is ON while the communication is in process. After the communication is complete, the system resets it to OFF automatically.
SM100~SM101	The system automatically sets the flag to ON, and users reset it to OFF. The flag is set to ON when the command is received.
SM102~SM103	The system automatically sets the flag to ON, and users reset it to OFF.

Special auxiliary relay	Refresh time
	The flag is automatically set to ON when the command received is wrong.
SM104~SM105	The system automatically sets the flag to ON, and users reset it to OFF. The flag is set to ON when there is a receive timeout.
SM106~SM107	Users set the flag to ON and reset it to OFF. ON: The 8-bit mode OFF: The 16-bit mode
SM166~SM167	Users set the flag to ON and reset it to OFF.
SM168~SM171	The system automatically sets the flag to ON, and reset it to OFF.
SM204~SM205	Users set the flag to ON, and the system automatically resets it to OFF. ON: Clearing the non-latched/latched areas
SM206	Users set SM206 to ON and reset it to OFF. ON: Inhibiting all output
SM209	Users set SM209 to ON, and the system automatically resets it to OFF. ON: The communication protocol of COM1 changes.
SM210	Users set SM210 to ON and reset it to OFF for COM1. ON: The RTU mode OFF: The ASCII mode
SM211	Users set SM211 to ON, and the system automatically resets it to OFF. ON: The communication protocol of COM2 changes.
SM212	Users set SM210 to ON and reset it to OFF for COM2. ON: The RTU mode OFF: The ASCII mode
SM215	Users set SM215 to ON and reset it to OFF. ON: The PLC runs. OFF: The PLC stops.
SM218	The system checks the real-time clock when power-on. ON: when the real-time clock is malfunction Users reset it to OFF.
SM219	The system monitors the battery power of the real-time clock. ON: when the power is low The system resets it to OFF.
SM220	Users set SM220 to ON and reset it to OFF. ON: Calibrating the real-time clock within ±30 seconds
SM221	The flag is refreshed according to the settings in HWCONFIG or when the instruction API1607 DST is executed. ON: the instruction DST is executed.
SM270 ~ SM275	The flag is ON when the instruction CSFO is executed. ON: enabling the function of reversing the input direction OFF: disabling the function of reversing the input direction
SM281~SM288	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM291~SM296	Users set the flag to ON and reset it to OFF. ON: enabling the function of clearing the input points OFF: disabling the function of clearing the input points
SM300	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up

Special auxiliary relay	Refresh time
SM301~SM303	The system set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM304	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM305~SM307	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM308	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM309~SM311	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM312	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM313~SM315	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM316	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM317~SM319	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM320	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM321~SM323	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM332	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM333~SM335	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM336	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM337~SM339	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM340	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up

Special auxiliary relay	Refresh time
SM341	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM342	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM343	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM344	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM345	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM346	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM347	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM348	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM349	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM350	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM351	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM352~SM353	Users set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM400~SM403	The system automatically sets the flag to ON and resets it to OFF. The flag is refreshed every scan cycle.
SM404	The system automatically sets the flag to ON and resets it to OFF. SM404 is refreshed every 5 milliseconds.
SM405	The system automatically sets SM405 to ON and resets it to OFF. SM405 is refreshed every 50 milliseconds.
SM406	The system automatically sets SM406 to ON and resets it to OFF. SM406 is refreshed every 100 milliseconds.
SM407	The system automatically sets SM407 to ON and resets it to OFF. SM407 is refreshed every 0.5 seconds.
SM450	The system automatically sets SM450 to ON and resets it to OFF. ON: The memory card is inserted into the PLC.

Special auxiliary relay	Refresh time
	OFF: The memory card is removed out of the PLC.
SM452~SM453	The system sets the flag to ON or OFF.
SM454	Users set the flag to ON or OFF.
SM455	The system sets the flag to ON or OFF.
SM456	Users set the flag to ON to save. After the saving is complete, the system resets it to OFF.
SM457	The system sets the flag to ON or OFF.
SM460	The system sets the flag to ON or OFF.
SM461	The system sets the flag to ON and users reset it to OFF.
SM462~SM464	Users set the flag to ON or OFF.
SM465	The system sets the flag to ON and users reset it to OFF.
SM466	Users set the flag to ON or OFF.
SM467	The system sets the flag to ON and users reset it to OFF.
SM468~SM471	Users set the flag to ON or OFF.
SM472	The system sets the flag to ON or OFF.
SM473	The system sets the flag to ON and users reset it to OFF.
SM474~SM477	Users set the flag to ON or OFF.
SM480	The system sets the flag to ON or OFF.
SM481	The system sets the flag to ON and users reset it to OFF.
SM482~SM484	Users set the flag to ON or OFF.
SM485	The system sets the flag to ON and users reset it to OFF.
SM486	Users set the flag to ON or OFF.
SM487	The system sets the flag to ON and users reset it to OFF.
SM488~SM491	Users set the flag to ON or OFF.
SM492	The system sets the flag to ON or OFF.
SM493	The system sets the flag to ON and users reset it to OFF.
SM494~SM497	Users set the flag to ON or OFF.
SM500	The system sets the flag to ON or OFF.
SM501	The system sets the flag to ON and users reset it to OFF.
SM502~SM504	Users set the flag to ON or OFF.
SM505	The system sets the flag to ON and users reset it to OFF.
SM506	Users set the flag to ON or OFF.
SM507	The system sets the flag to ON and users reset it to OFF.
SM508~SM511	Users set the flag to ON or OFF.
SM512	The system sets the flag to ON or OFF.
SM513	The system sets the flag to ON and users reset it to OFF.
SM514~SM517	Users set the flag to ON or OFF.
SM520	The system sets the flag to ON or OFF.
SM521	The system sets the flag to ON and users reset it to OFF.
SM522~SM524	Users set the flag to ON or OFF.
SM525	The system sets the flag to ON and users reset it to OFF.
SM526	Users set the flag to ON or OFF.
SM527	The system sets the flag to ON and users reset it to OFF.
SM528~SM531	Users set the flag to ON or OFF.
SM532	The system sets the flag to ON or OFF.
SM533	The system sets the flag to ON and users reset it to OFF.
SM534~SM537	Users set the flag to ON or OFF.

Special auxiliary relay	Refresh time
SM540	The system sets the flag to ON or OFF.
SM541	The system sets the flag to ON and users reset it to OFF.
SM542~SM544	Users set the flag to ON or OFF.
SM545	The system sets the flag to ON and users reset it to OFF.
SM546	Users set the flag to ON or OFF.
SM547	The system sets the flag to ON and users reset it to OFF.
SM548~SM551	Users set the flag to ON or OFF.
SM552	The system sets the flag to ON or OFF.
SM553	The system sets the flag to ON and users reset it to OFF.
SM554~SM557	Users set the flag to ON or OFF.
SM560	The system sets the flag to ON or OFF.
SM561	The system sets the flag to ON and users reset it to OFF.
SM562~SM564	Users set the flag to ON or OFF.
SM565	The system sets the flag to ON and users reset it to OFF.
SM566	Users set the flag to ON or OFF.
SM567	The system sets the flag to ON and users reset it to OFF.
SM568~SM569	Users set the flag to ON or OFF.
SM572	The system sets the flag to ON or OFF.
SM573	The system sets the flag to ON and users reset it to OFF.
SM574	Users set the flag to ON or OFF.
SM580	Users set the flag to ON and the system resets it to OFF. ON: disabling the function of high-speed output
SM600~SM602	The system automatically sets the flag to ON and resets it to OFF. The flag is refreshed when the instruction is executed.
SM604	Users set SM604 to ON and reset it to OFF. ON: Sort by descending OFF: Sort by ascending
SM605	Users set SM605 to ON and reset it to OFF.
SM606	Users set SM606 to ON and reset it to OFF. ON: The 8-bit mode OFF: The 16-bit mode
SM607	Users set the flag to ON or OFF.
SM608	The flag is refreshed when the instruction is executed.
SM609	Users set the flag to ON or OFF.
SM610~SM611	The flag is refreshed when the instruction is executed.
SM612~SM613	Users set the flag to ON or OFF.
SM614	The flag is refreshed when the instruction is executed.
SM615~SM617	Users set the flag to ON or OFF.
SM618	The flag is refreshed when the instruction is executed.
SM619	The flag is refreshed when the instruction EI or DI is executed.
SM620	The flag is refreshed when the instruction CMPT is executed.
SM621~SM686	Users set the flag to ON or OFF.
SM687	The flag is refreshed when the instruction RAMP is executed.
SM688	The flag is refreshed when the instruction INCD is executed.
SM690~SM691	Users set the flag to ON or OFF.
SM692	The flag is refreshed when the instruction HKY is executed.

Special auxiliary relay	Refresh time
SM693	The flag is refreshed when the instruction SEGL is executed.
SM694	The flag is refreshed when the instruction DSW is executed.
SM695	Users set the flag to ON or OFF.
SM749	Every time after the parameters of data exchange are downloaded, the system is refreshed when power-on.
SM750~SM783	After the parameters of data exchange are downloaded, users set the flag to ON or OFF.
SM784~SM847	The flag is ON when the system is refreshed.
SM861	Every time after the parameters of data exchange are downloaded, the system is refreshed when power-on.
SM862~SM895	After the parameters of data exchange are downloaded, users set the flag to ON or OFF.
SM896~SM959	The flag is ON, when the system is refreshed automatically.
SM1000	Users set the flag to ON and after saving; the system sets the flag to OFF.
SM1001	The flag is ON, when the Ethernet connection is active. The flag is OFF, when the Ethernet connection is not active.
SM1090	SM1090 is ON when the TCP connection is busy.
SM1091	SM1091 is ON when the UDP connection is busy.
SM1100	The flag is refreshed when API2200-API2210 is executed or the network cable is reconnected.
SM1106	SM1106 is ON when the PHY initialization fails.
SM1107	SM1107 is ON when the IP address, the netmask address, and the gateway address are set incorrectly.
SM1109	SM1109 is ON when the function of the socket is enabled and the same port is used.
SM1111	Users set the flag to ON or OFF.
SM1113	The flag is ON, when there is a server error.
SM1116	SM1116 is ON when the trigger of the PLC parameter is enabled.
SM1117	SM1117 is ON when the trigger of the PLC parameter is triggered.
SM1119	SM1119 is ON when the trigger is enabled and the last mail has been sent successfully.
SM1120	SM1120 is ON when the trigger is enabled and the last mail has been sent in error.
SM1122~SM1123	The flag is ON when the trigger is enabled and there is an SMTP server response timeout.
SM1124	SM1124 is ON when the trigger is enabled and the size of the attachment exceeds the limit.
SM1125	SM1125 is ON when the trigger is enabled and the attachment is not found.
SM1126~SM1127	The flag is ON when the trigger of the PLC parameter is enabled.
SM1129	SM1129 is ON when the trigger is enabled and the last mail has been sent successfully.
SM1130	SM1130 is ON when the trigger is enabled and the last mail has been sent in error.
SM1132	SM1132 is ON when the trigger is enabled and there is an SMTP server response timeout.
SM1133	SM1133 is ON when the trigger is enabled and there is an SMTP server response error.
SM1134	SM1134 is ON when the trigger is enabled and the size of the attachment exceeds the limit.
SM1135	SM1135 is ON when the trigger is enabled and the attachment is not found.
SM1136	SM1136 is ON when the trigger of the PLC parameter is enabled.
SM1137	SM1137 is ON when the trigger of the PLC parameter is triggered.
SM1139	SM1139 is ON when the trigger is enabled and the last mail has been sent successfully.
SM1140	SM1140 is ON when the trigger is enabled and the last mail has been sent in error.
SM1142	SM1142 is ON when the trigger is enabled and there is an SMTP server response timeout.
SM1143	SM1143 is ON when the trigger is enabled and there is an SMTP server response error.
SM1144	SM1144 is ON when the trigger is enabled and the size of the attachment exceeds the limit.
SM1145	SM1145 is ON when the trigger is enabled and the attachment is not found.

Special auxiliary relay	Refresh time
SM1146	SM1146 is ON when the trigger of the PLC parameter is enabled.
SM1147	SM1147 is ON when the trigger of the PLC parameter is triggered.
SM1149	SM1149 is ON when the trigger is enabled and the last mail has been sent successfully.
SM1150	SM1150 is ON when the trigger is enabled and the last mail has been sent in error.
SM1152	SM1152 is ON when the trigger is enabled and there is an SMTP server response timeout.
SM1153	SM1153 is ON when the trigger is enabled and there is an SMTP server response error.
SM1154	SM1154 is ON when the trigger is enabled and the size of the attachment exceeds the limit.
SM1155	SM1155 is ON when the trigger is enabled and the attachment is not found.
SM1166	After the parameters of data exchange are downloaded, the system is refreshed.
SM1167~SM1199	After the parameters of data exchange are downloaded, users set the flag to ON or OFF.
SM1200~SM1263	The flag is ON, when the system is refreshed.
SM1269	The flag is ON, when there is a socket configuration error.
SM1270~SM1353	The flag is refreshed when the socket function is executed.
SM1375~SM1407	After the parameters of data exchange via EtherNet/IP are downloaded, users set the flag to ON or OFF.1.
SM1408~SM1439	The flag is ON when an error occurs in data exchange via EtherNet/IP.
SM1440~SM1447	The flag is ON when a timeout occurs in the slave of the I/O connection via EtherNet/IP
SM1631~SM1638	The system sets the flag to ON and users set it to OFF.
SM1641~SM1648	Users set the flag to ON or OFF.
SM1651~SM1658	The system sets the flag to ON or OFF.
SM1661~SM1668	Users set the flag to ON or OFF.
SM1671~SM1682	The system sets the flag to ON or OFF.

2.2.9 Stepping Relays (S)

Function of the stepping relay:

The stepping relay can be easily used in the industrial automation to set the procedure. It is the most basic device in the sequential function chart (SFC). Please refer to ISPSoft User Manual for more information related to sequential function charts.

There are 2048 stepping relays, i.e. S0~S2047. Every stepping relay is like an output relay in that it has an output coil, contact A, and contact B. It can be used several times in the program, but it cannot directly drive the external load. Besides, the stepping relay can be used as a general auxiliary relay when it is not used in the sequential function chart.

2.2.10 Timers (T)

1. 100 millisecond timer: The timer specified by the instruction TMR takes 100 milliseconds as the timing unit.
2. 1 millisecond timer: The timer specified by the instruction TMRH takes 1 millisecond as the timing unit.
3. The accumulative timers are ST0~ST511. If users want to use the device-monitoring function, they can monitor T0~T511.
4. If the same timer is used repeatedly in the program, including in different instructions TMR and TMRH, the setting value is the one that the value of the timer matches first.
5. If the same timer is used repeatedly in the program, it is OFF when one of the conditional contacts is OFF.
6. If the same timer is used repeatedly in the program as the timer for the subroutine's exclusive use and the accumulative timer in the program, it is OFF when one of the conditional contacts is OFF.

7. When the timer is switched from ON to OFF and the conditional contact is ON, the timer is reset and counts again.

8. When the instruction TMR is executed, the specified timer coil is ON and the timer begins to count. As the value of the timer matches the setting value (value of the timer ≧ setting value), the state of the contact is ON.

A. General-purpose timer

When the instruction TMR is executed, the general-purpose timer begins to count. As the value of the timer matches the setting value, the output coil is ON.

1. When X0.0=ON and the timer takes 100ms as the timing unit, the output coil T0 will be ON, when the value of the timer = setting value100.

2. When X0.0=OFF or the power is off, the value of the timer is 0 and the output coil T0 will be OFF.

B. Accumulative timer

When the instruction TMR is executed, the accumulative timer begins to count. As the value of the timer matches the setting value, the output coil is ON. As long as users add the letter S in front of the letter T, the timer becomes the accumulative timer. When the conditional contact is OFF, the value of the accumulative timer is not reset. When the conditional contact is ON, the timer will count from the current value.

3. When X0.0=ON and the timer T250 takes 100ms as the timing unit, the output coil T250 will be ON, when the value of the timer = setting value100.

4. When X0.0=OFF or the power is off, the timer T250 stops counting, and the value of the timer stays the same. When X0.0=ON, the value of the timer will be accumulated and when the accumulated value = setting value 100, the output coil T250 will be ON.

C. Timer used in the function block

T412~T511 are the timers which users can use in the functional block or the interrupt.

When the instruction TMR or END is executed, the timer used in the functional block begins to count. As the value of the timer matches the setting value, the output coil is ON.

If the general-purpose timer is used in the functional block or the interrupt, and the functional is not executed, the timer cannot count correctly.

2.2.11 Counters

- Characteristics of the 16-bit counter

Item	16-bit counter
Type	General type
Number	C0~C511
Direction	Counting up
Setting value	0~32,767
Specification of the setting value	The setting value can be either the constant or the value in the data register.
Change of the current value	The counter stops counting when the value of the counter matches the setting value.
Output contact	The contact is ON when the value of the counter matches the setting value.
Reset	When the instruction RST is executed, the current value is cleared to zero, and the contact is reset of OFF.
Action of the contact	After the scan is complete, the contact acts.

- Function of the counter

Each time the input switches from OFF to ON, the value of the counter is the same as the output coil. Users can use either the decimal constant or the value in the data register as the setting value.

16-bit counter:

1. Setting range: 0~32,767 (The setting values 0 and 1 mean the same thing in that the output contact is ON when the counter counts for the first time.)
2. For the general-purpose counter, the current value of the counter is cleared when there is a power cut. If the counter is the latched one, the current value of the counter and the state of the contact before the power cut will be retained. The latched counter counts from the current value when the power supply is restored.
3. If users use the instruction MOV or ISPSoft to transmit a value bigger than the setting value to the current value register C0, the contact of the counter C0 will be ON and the current value will become the same as the setting value next time X0.1 is switched from OFF to ON.
4. Users can use either the constant or the value in the data register as the setting value of the counter.
5. The setting value of the counter can be a positive or a negative. If the counter counts up from 32,767, the next current value becomes 0.

1. When X0.0=ON, the instruction RST will be executed and the current value of C0 will be reset to zero and the output contact of the counter C0 will be FF.
2. When X0.1 is from OFF to ON, the value of the counter increases by one increment.
3. When the value of the counter C0 reached the setting value 5, the contact of the counter C0 will be ON (the current value of C0 = the setting value = 5). After that the trigger from X0.1 will not be accepted by C0 and the current value of C0 will stay at the value 5.

2.2.12 32-bit Counters (HC)

- Characteristics of the 32-bit counter

Item	32-bit counter		
Type	Up/down counter	Up counter	High-speed counter
Number	HC0 ~ HC63	HC64 ~ HC199	HC200 ~ HC255
Direction	Counting up/down	Counting up	Counting up/down
Setting value	-2,147,483,648~+2,147,483,647		
Specification of the setting value	The setting value can be either the constant or the value occupying two data registers (32-bit).		
Change of the current value	The counter keeps counting even after the value of the counter matches the setting value.		
Output contact	The contact is ON when the value of the addition counter matches the setting value. The contact is reset to OFF when the value of the subtraction counter matches the setting value.		
Reset	When the instruction RST is executed, the current value is cleared to zero, and the contact is reset to OFF.		
Action of the contact	After the instruction DCNT scan is complete, the contact acts.		

- 32-bit general-purpose addition/subtraction counter

  1. Setting range: -2,147,483,648~2,147,483,647
  2. The switch between the 32-bit general-purpose addition counters and the 32-bit general-purpose subtraction counters depends on the states of the special auxiliary relays SM621~SM684. For example, the counter HC0 is the addition counter when SM621 is OFF, whereas HC0 is the subtraction counter when SM621 is ON.
  3. Users can use either the constant or the value in the data registers as the setting value of the counter, and the setting value can be a positive or a negative. If users use the value in the data registers as the setting value of the counter, the setting value occupies two consecutive registers.
  4. For the general-purpose counter, the current value of the counter is cleared when there is a power cut. If the counter is the latched one, the current value of the counter and the state of the contact before the power cut will be retained. The latched counter counts from the current value when the power supply is restored.
  5. If the counter counts up from 2,147,483,647, the next current value becomes -2,147,483,648. If the counter counts down from -2,147,483,648, the next current value becomes 2,147,483,647.

- 32-bit high speed addition/subtraction counter

Please refer to the instruction description of API1004 DCNT in AS Series Programming Manual for more details.

Example:

1. X10.0 drives SM621 to determine counting direction (up/down) of HC0.
2. When X11.0 goes from OFF to ON, RST instruction will be executed and the PV in HC0 will be cleared to 0 and its contact is OFF.
3. When X12.0 goes from OFF to ON, PV of HC0 will count up (plus 1) or count down (minus 1).
4. When PV in HC0 changes from -6 to -5, the contact HC0 will go from OFF to ON. When PV in HC0 changes from -5 to -6, the contact HC0 will go from ON to OFF.
5. If MOV instruction is applied through ISPSoft to designate a value bigger than SV to the PV register of HC0, next time when X12.0 goes from OFF to ON, the contact HC0 will be ON and PV of HC0 will equal SV.

2.2.13 Data Registers (D)

The data register stores the 16-bit data. The highest bit represents either a positive sign or a negative sign, and the values which can be stored in the data registers range from -32,768 to +32,767. Two 16-bit registers can be combined into a 32-bit register, i.e. (D+1, D) in which the register whose number is smaller represents the low 16 bits. The highest bit represents either a positive sign or a negative sign, and the values which can be stored in the data registers range from -2,147,483,648 to +2,147,483,647. Besides, four 16-bit registers can be combined into a 64-bit register, i.e. (D+3, D+2, D+1, D) in which the register whose number is smaller represents the lower 16 bits. The highest bit represents either a positive sign or a negative sign, and the values which can be stored in the data registers range from -9,223,372,036,854,776 to +9,223,372,036,854,775,807. The data registers can also be used to refresh the values in the control registers in the modules other than digital I/O modules. Please refer to ISPSoft User Manual for more information regarding refreshing the values in the control registers.

The registers can be classified into two types according to their properties.

1. General-purpose register: When the PLC begins to run, or is disconnected, the value in the register will be cleared to zero. If users want to retain the data when the PLC begins to RUN, they can refer to ISPSoft User Manual for more information. Please notice that the value will still be cleared to zero when the PLC is disconnected.
2. Latched register: If the PLC is disconnected, the data in the latched register will not be cleared. In other words, the value before the disconnection is still retained. If users want to clear the data in the latched area, they can use RST or ZRST.

2.2.14 Special Data Registers (SR)

Every special data register has its definition and specific function. The system statuses and the error messages are stored in the special data registers. Besides, the special data registers can be used to monitor the system statuses. The special data registers and their functions are listed as follows. As to the SR numbers marked “*”, users can refer to the additional remarks on special auxiliary relays/special data registers. The “R” in the attribute column indicates that the special data register can read the data, whereas the “R/W” in the attribute column indicates that it can read and write the data. In addition, the mark “–” indicates that the status of the special data register does not make any change. The mark “#” indicates that the system will be set according to the status of the PLC, and users can read the setting value and refer to the related manual for more information.

The “Y” in the column latched means it is latched, the “N” means it is non-latched; as for “H”, it means it follows the settings in HWCONFIG. While execution, the programs in the PLC are editable; but the settings in the HWCONFIG are the same.

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR0	Error-detecting code of the PLC operation/operand error	○	0	0	-	N	R	0
SR1	The address of the operation error (32-bit)	○	0	0	-	N	R	0
SR2								
SR4	Error-detecting code of the grammar check error	○	0	0	-	N	R	0
SR5	The address of the grammar check error (32-bit)	○	0	0	-	N	R	0
SR6								
*SR8	Step address at which the watchdog timer is ON (32-bit)	○	0	-	-	N	R	0
SR9								
SR23	The number of times the MAC address is made.	○	-	-	-	N	R	-
SR28	The last output number of which the instruction high	○	-1	-1	-1	N	R	-1

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	speed output is used repeatedly							
SR32 SR33	The last instruction address that exceeded the allowed range.	○	-1	-1	-	N	R	-1
*SR36	The system saves data to the memory card. This function need to work with SM36.	○	0	-	-	N	R/W	0
*SR40	Number of error logs	○	-	-	-	Y	R	0
*SR41	Error log pointer	○	-	-	-	Y	R	0
*SR43	Error log 1: The module ID	○	-	-	-	Y	R	0
*SR44	Error log 1: The error code	○	-	-	-	Y	R	0
*SR45	Error log 1: The year and the month	○	-	-	-	Y	R	0
*SR46	Error log 1: The day and the hour	○	-	-	-	Y	R	0
*SR47	Error log 1: The minute and the second	○	-	-	-	Y	R	0
*SR49	Error log 2: The module ID	○	-	-	-	Y	R	0
*SR50	Error log 2: The error code	○	-	-	-	Y	R	0
*SR51	Error log 2: The year and the month	○	-	-	-	Y	R	0
*SR52	Error log 2: The day and the hour	○	-	-	-	Y	R	0
*SR53	Error log 2: The minute and the second	○	-	-	-	Y	R	0
*SR55	Error log 3: The module ID	○	-	-	-	Y	R	0
*SR56	Error log 3: The error code	○	-	-	-	Y	R	0
*SR57	Error log 3: The year and the month	○	-	-	-	Y	R	0
*SR58	Error log 3: The day and the hour	○	-	-	-	Y	R	0
*SR59	Error log 3: The minute and the second	○	-	-	-	Y	R	0
*SR61	Error log 4: The module ID	○	-	-	-	Y	R	0
*SR62	Error log 4: The error code	○	-	-	-	Y	R	0
*SR63	Error log 4: The year and the month	○	-	-	-	Y	R	0
*SR64	Error log 4: The day and the hour	○	-	-	-	Y	R	0
*SR65	Error log 4: The minute and the second	○	-	-	-	Y	R	0
*SR67	Error log 5: The module ID	○	-	-	-	Y	R	0
*SR68	Error log 5: The error code	○	-	-	-	Y	R	0
*SR69	Error log 5: The year and the month	○	-	-	-	Y	R	0
*SR70	Error log 5: The day and the hour	○	-	-	-	Y	R	0
*SR71	Error log 5: The minute and the second	○	-	-	-	Y	R	0
*SR73	Error log 6: The module ID	○	-	-	-	Y	R	0
*SR74	Error log 6: The error code	○	-	-	-	Y	R	0
*SR75	Error log 6: The year and the month	○	-	-	-	Y	R	0
*SR76	Error log 6: The day and the hour	○	-	-	-	Y	R	0
*SR77	Error log 6: The minute and the second	○	-	-	-	Y	R	0
*SR79	Error log 7: The module ID	○	-	-	-	Y	R	0
*SR80	Error log 7: The error code	○	-	-	-	Y	R	0
*SR81	Error log 7: The year and the month	○	-	-	-	Y	R	0
*SR82	Error log 7: The day and the hour	○	-	-	-	Y	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR83	Error log 7: The minute and the second	○	-	-	-	Y	R	0
*SR85	Error log 8: The module ID	○	-	-	-	Y	R	0
*SR86	Error log 8: The error code	○	-	-	-	Y	R	0
*SR87	Error log 8: The year and the month	○	-	-	-	Y	R	0
*SR88	Error log 8: The day and the hour	○	-	-	-	Y	R	0
*SR89	Error log 8: The minute and the second	○	-	-	-	Y	R	0
*SR91	Error log 9: The module ID	○	-	-	-	Y	R	0
*SR92	Error log 9: The error code	○	-	-	-	Y	R	0
*SR93	Error log 9: The year and the month	○	-	-	-	Y	R	0
*SR94	Error log 9: The day and the hour	○	-	-	-	Y	R	0
*SR95	Error log 9: The minute and the second	○	-	-	-	Y	R	0
*SR97	Error log 10: The module ID	○	-	-	-	Y	R	0
*SR98	Error log 10: The error code	○	-	-	-	Y	R	0
*SR99	Error log 10: The year and the month	○	-	-	-	Y	R	0
*SR100	Error log 10: The day and the hour	○	-	-	-	Y	R	0
*SR101	Error log 10: The minute and the second	○	-	-	-	Y	R	0
*SR103	Error log 11: The module ID	○	-	-	-	Y	R	0
*SR104	Error log 11: The error code	○	-	-	-	Y	R	0
*SR105	Error log 11: The year and the month	○	-	-	-	Y	R	0
*SR106	Error log 11: The day and the hour	○	-	-	-	Y	R	0
*SR107	Error log 11: The minute and the second	○	-	-	-	Y	R	0
*SR109	Error log 12: The module ID	○	-	-	-	Y	R	0
*SR110	Error log 12: The error code	○	-	-	-	Y	R	0
*SR111	Error log 12: The year and the month	○	-	-	-	Y	R	0
*SR112	Error log 12: The day and the hour	○	-	-	-	Y	R	0
*SR113	Error log 12: The minute and the second	○	-	-	-	Y	R	0
*SR115	Error log 13: The module ID	○	-	-	-	Y	R	0
*SR116	Error log 13: The error code	○	-	-	-	Y	R	0
*SR117	Error log 13: The year and the month	○	-	-	-	Y	R	0
*SR118	Error log 13: The day and the hour	○	-	-	-	Y	R	0
*SR119	Error log 13: The minute and the second	○	-	-	-	Y	R	0
*SR121	Error log 14: The module ID	○	-	-	-	Y	R	0
*SR122	Error log 14: The error code	○	-	-	-	Y	R	0
*SR123	Error log 14: The year and the month	○	-	-	-	Y	R	0
*SR124	Error log 14: The day and the hour	○	-	-	-	Y	R	0
*SR125	Error log 14: The minute and the second	○	-	-	-	Y	R	0
*SR127	Error log 15: The module ID	○	-	-	-	Y	R	0
*SR128	Error log 15: The error code	○	-	-	-	Y	R	0
*SR129	Error log 15: The year and the month	○	-	-	-	Y	R	0
*SR130	Error log 15: The day and the hour	○	-	-	-	Y	R	0
*SR131	Error log 15: The minute and the second	○	-	-	-	Y	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR133	Error log 16: The module ID	○	-	-	-	Y	R	0
*SR134	Error log 16: The error code	○	-	-	-	Y	R	0
*SR135	Error log 16: The year and the month	○	-	-	-	Y	R	0
*SR136	Error log 16: The day and the hour	○	-	-	-	Y	R	0
*SR137	Error log 16: The minute and the second	○	-	-	-	Y	R	0
*SR139	Error log 17: The module ID	○	-	-	-	Y	R	0
*SR140	Error log 17: The error code	○	-	-	-	Y	R	0
*SR141	Error log 17: The year and the month	○	-	-	-	Y	R	0
SR142	Error log 17: The day and the hour	○	-	-	-	Y	R	0
*SR143	Error log 17: The minute and the second	○	-	-	-	Y	R	0
*SR145	Error log 18: The module ID	○	-	-	-	Y	R	0
*SR146	Error log 18: The error code	○	-	-	-	Y	R	0
*SR147	Error log 18: The year and the month	○	-	-	-	Y	R	0
*SR148	Error log 18: The day and the hour	○	-	-	-	Y	R	0
*SR149	Error log 18: The minute and the second	○	-	-	-	Y	R	0
*SR151	Error log 19: The module ID	○	-	-	-	Y	R	0
*SR152	Error log 19: The error code	○	-	-	-	Y	R	0
*SR153	Error log 19: The year and the month	○	-	-	-	Y	R	0
*SR154	Error log 19: The day and the hour	○	-	-	-	Y	R	0
*SR155	Error log 19: The minute and the second	○	-	-	-	Y	R	0
*SR157	Error log 20: The module ID	○	-	-	-	Y	R	0
*SR158	Error log 20: The error code	○	-	-	-	Y	R	0
*SR159	Error log 20: The year and the month	○	-	-	-	Y	R	0
*SR160	Error log 20: The day and the hour	○	-	-	-	Y	R	0
*SR161	Error log 20: The minute and the second	○	-	-	-	Y	R	0
SR162 SR163	Duration of how long the PLC is being powered-on (unit: minutes) (32-bit)	○	-	-	-	Y	R	-
SR166	VR0 value (need to work with SM166)	○	0	-	-	N	R	0
SR167	VR1 value (need to work with SM167)	○	0	-	-	N	R	0
SR168	The value in the channel 1 of the Function Card 1 F2AD	○	0	-	-	N	R	0
SR169	The value in the channel 2 of the Function Card 1 F2AD	○	0	-	-	N	R	0
SR170	The value in the channel 1 of the Function Card 2 F2AD	○	0	-	-	N	R	0
SR171	The value in the channel 2 of the Function Card 2 F2AD	○	0	-	-	N	R	0
SR172	The value in the channel 1 of the Function Card 1 F2DA	○	0	-	0	N	R/W	0
SR173	The value in the channel 2 of the Function Card 1 F2DA	○	0	-	0	N	R/W	0
SR174	The value in the channel 1 of the Function Card 2 F2DA	○	0	-	0	N	R/W	0
SR175	The value in the channel 2 of the Function Card 2 F2DA	○	0	-	0	N	R/W	0
SR176	The communication ID of the Function Card 1 (COM11)	○	-	-	-	N	R	1
SR177	The protocol code of the Function Card 1 (COM11)	○	-	-	-	N	R	0x24
SR178	The communication ID of the Function Card 2 (COM12)	○	-	-	-	N	R	1
SR179	The protocol code of the Function Card 2 (COM12)	○	-	-	-	N	R	0x24

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR180	The last error code of warning	○	0	-	-	N	R	0
SR182	The duration of a timeout for the Communication Card 1 (unit: minutes), when the value is 0, there is no timeout.	○	-	-	-	N	R/W	200
SR183	The duration of a timeout for the Communication Card 2 (unit: minutes), when the value is 0, there is no timeout.	○	-	-	-	N	R/W	200
SR185	The communication cycle time of all the remote modules (unit: minutes)	○	0	-	-	N	R	0
SR190	Frequency multiplication of the high speed counter group 1 (default: 1-time frequency)	○	1	-	-	N	R/W	1
SR191	Frequency multiplication of the high speed counter group 2 (default: 1-time frequency)	○	1	-	-	N	R/W	1
SR192	Frequency multiplication of the high speed counter group 3 (default: 1-time frequency)	○	1	-	-	N	R/W	1
SR193	Frequency multiplication of the high speed counter group 4 (default: 1-time frequency)	○	1	-	-	N	R/W	1
SR194	Frequency multiplication of the high speed counter group 5 (default: 1-time frequency)	○	1	-	-	N	R/W	1
SR195	Frequency multiplication of the high speed counter group 6 (default: 1-time frequency)	○	1	-	-	N	R/W	1
SR196	Frequency multiplication of the high speed counter group 7 (default: 1-time frequency)	○	1	-	-	N	R/W	1
SR197	Frequency multiplication of the high speed counter group 8 (default: 1-time frequency)	○	1	-	-	N	R/W	1
SR198	Pi (π), floating-point number (32-bit)	○	16#0FDB	16#0FDB	16#0FDB	N	R	16#0FDB
SR199			16#4049	16#4049	16#4049	N		16#4049
*SR201	Communication address of COM1	○	-	-	-	H	R/W	1
*SR202	Communication address of COM2	○	-	-	-	H	R/W	1
*SR209	Communication protocol of COM1	○	-	-	-	H	R/W	16#0024
*SR210	COM1 communication timeout (unit: minute) 0 means no timeout	○	-	-	-	H	R/W	0
*SR212	Communication protocol of COM2	○	-	-	-	H	R/W	16#0024
*SR213	COM2 communication timeout (unit: minute) 0 means no timeout	○	-	-	-	H	R/W	0
*SR215	Name of the function car 1	○	-	-	-	N	R	0
*SR216	Name of the function car 2	○	-	-	-	N	R	0
SR217	Baudrate value for COM1 (unit:100)	○	96	-	-	H	R/W	96
SR218	Baudrate value for COM2 (unit:100)	○	96	-	-	H	R/W	96
*SR220	Value of the year in the real-time clock (RTC): 00~99 (A.D.)	○	-	-	-	Y	R	0
*SR221	Value of the month in the real-time clock (RTC): 01~12	○	-	-	-	Y	R	1

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR222	Value of the day in the real-time clock (RTC): 1~31	○	-	-	-	Y	R	1
*SR223	Value of the hour in the real-time clock (RTC): 00~23	○	-	-	-	Y	R	0
*SR224	Value of the minute in the real-time clock (RTC): 00~59	○	-	-	-	Y	R	0
*SR225	Value of the second in the real-time clock (RTC): 00~59	○	-	-	-	Y	R	0
*SR226	Value of the week in the real-time clock (RTC): 1~7	○	-	-	-	Y	R	1
*SR227	Number of download logs (The maximum number is 20.)	○	-	-	-	Y	R	0
*SR228	Download log pointer	○	-	-	-	Y	R	0
*SR229	Download log 1: The action number	○	-	-	-	Y	R	0
*SR230	Download log 1: The year and the month	○	-	-	-	Y	R	0
*SR231	Download log 1: The day and the hour	○	-	-	-	Y	R	0
*SR232	Download log 1: The minute and the second	○	-	-	-	Y	R	0
*SR233	Download log 2: The action number	○	-	-	-	Y	R	0
*SR234	Download log 2: The year and the month	○	-	-	-	Y	R	0
*SR235	Download log 2: The day and the hour	○	-	-	-	Y	R	0
*SR236	Download log 2: The minute and the second	○	-	-	-	Y	R	0
*SR237	Download log 3: The action number	○	-	-	-	Y	R	0
*SR238	Download log 3: The year and the month	○	-	-	-	Y	R	0
*SR239	Download log 3: The day and the hour	○	-	-	-	Y	R	0
*SR240	Download log 3: The minute and the second	○	-	-	-	Y	R	0
*SR241	Download log 4: The action number	○	-	-	-	Y	R	0
*SR242	Download log 4: The year and the month	○	-	-	-	Y	R	0
*SR243	Download log 4: The day and the hour	○	-	-	-	Y	R	0
*SR244	Download log 4: The minute and the second	○	-	-	-	Y	R	0
*SR245	Download log 5: The action number	○	-	-	-	Y	R	0
*SR246	Download log 5: The year and the month	○	-	-	-	Y	R	0
*SR247	Download log 5: The day and the hour	○	-	-	-	Y	R	0
*SR248	Download log 5: The minute and the second	○	-	-	-	Y	R	0
*SR249	Download log 6: The action number	○	-	-	-	Y	R	0
*SR250	Download log 6: The year and the month	○	-	-	-	Y	R	0
*SR251	Download log 6: The day and the hour	○	-	-	-	Y	R	0
*SR252	Download log 6: The minute and the second	○	-	-	-	Y	R	0
*SR253	Download log 7: The action number	○	-	-	-	Y	R	0
*SR254	Download log 7: The year and the month	○	-	-	-	Y	R	0
*SR255	Download log 7: The day and the hour	○	-	-	-	Y	R	0
*SR256	Download log 7: The minute and the second	○	-	-	-	Y	R	0
*SR257	Download log 8: The action number	○	-	-	-	Y	R	0
*SR258	Download log 8: The year and the month	○	-	-	-	Y	R	0
*SR259	Download log 8: The day and the hour	○	-	-	-	Y	R	0
*SR260	Download log 8: The minute and the second	○	-	-	-	Y	R	0
*SR261	Download log 9: The action number	○	-	-	-	Y	R	0
*SR262	Download log 9: The year and the month	○	-	-	-	Y	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR263	Download log 9: The day and the hour	○	-	-	-	Y	R	0
*SR264	Download log 9: The minute and the second	○	-	-	-	Y	R	0
*SR265	Download log 10: The action number	○	-	-	-	Y	R	0
*SR266	Download log 10: The year and the month	○	-	-	-	Y	R	0
*SR267	Download log 10: The day and the hour	○	-	-	-	Y	R	0
*SR268	Download log 10: The minute and the second	○	-	-	-	Y	R	0
*SR269	Download log 11: The action number	○	-	-	-	Y	R	0
*SR270	Download log 11: The year and the month	○	-	-	-	Y	R	0
*SR271	Download log 11: The day and the hour	○	-	-	-	Y	R	0
*SR272	Download log 11: The minute and the second	○	-	-	-	Y	R	0
*SR273	Download log 12: The action number	○	-	-	-	Y	R	0
*SR274	Download log 12: The year and the month	○	-	-	-	Y	R	0
*SR275	Download log 12: The day and the hour	○	-	-	-	Y	R	0
*SR276	Download log 12: The minute and the second	○	-	-	-	Y	R	0
*SR277	Download log 13: The action number	○	-	-	-	Y	R	0
*SR278	Download log 13: The year and the month	○	-	-	-	Y	R	0
*SR279	Download log 13: The day and the hour	○	-	-	-	Y	R	0
*SR280	Download log 13: The minute and the second	○	-	-	-	Y	R	0
*SR281	Download log 14: The action number	○	-	-	-	Y	R	0
*SR282	Download log 14: The year and the month	○	-	-	-	Y	R	0
*SR283	Download log 14: The day and the hour	○	-	-	-	Y	R	0
*SR284	Download log 14: The minute and the second	○	-	-	-	Y	R	0
*SR285	Download log 15: The action number	○	-	-	-	Y	R	0
*SR286	Download log 15: The year and the month	○	-	-	-	Y	R	0
*SR287	Download log 15: The day and the hour	○	-	-	-	Y	R	0
*SR288	Download log 15: The minute and the second	○	-	-	-	Y	R	0
*SR289	Download log 16: The action number	○	-	-	-	Y	R	0
*SR290	Download log 16: The year and the month	○	-	-	-	Y	R	0
*SR291	Download log 16: The day and the hour	○	-	-	-	Y	R	0
*SR292	Download log 16: The minute and the second	○	-	-	-	Y	R	0
*SR293	Download log 17: The action number	○	-	-	-	Y	R	0
*SR294	Download log 17: The year and the month	○	-	-	-	Y	R	0
*SR295	Download log 17: The day and the hour	○	-	-	-	Y	R	0
*SR296	Download log 17: The minute and the second	○	-	-	-	Y	R	0
*SR297	Download log 18: The action number	○	-	-	-	Y	R	0
*SR298	Download log 18: The year and the month	○	-	-	-	Y	R	0
*SR299	Download log 18: The day and the hour	○	-	-	-	Y	R	0
*SR300	Download log 18: The minute and the second	○	-	-	-	Y	R	0
*SR301	Download log 19: The action number	○	-	-	-	Y	R	0
*SR302	Download log 19: The year and the month	○	-	-	-	Y	R	0
*SR303	Download log 19: The day and the hour	○	-	-	-	Y	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR304	Download log 19: The minute and the second	○	-	-	-	Y	R	0
*SR305	Download log 20: The action number	○	-	-	-	Y	R	0
*SR306	Download log 20: The year and the month	○	-	-	-	Y	R	0
*SR307	Download log 20: The day and the hour	○	-	-	-	Y	R	0
*SR308	Download log 20: The minute and the second	○	-	-	-	Y	R	0
*SR309	Number of PLC status change logs (The maximum number is 20.)	○	-	-	-	Y	R	0
*SR310	PLC status change log pointer	○	-	-	-	Y	R	0
*SR311	PLC status change log 1: The action number	○	-	-	-	Y	R	0
*SR312	PLC status change log 1: The year and the month	○	-	-	-	Y	R	0
*SR313	PLC status change log 1: The day and the hour	○	-	-	-	Y	R	0
*SR314	PLC status change log 1: The minute and the second	○	-	-	-	Y	R	0
*SR315	PLC status change log 2: The action number	○	-	-	-	Y	R	0
*SR316	PLC status change log 2: The year and the month	○	-	-	-	Y	R	0
*SR317	PLC status change log 2: The day and the hour	○	-	-	-	Y	R	0
*SR318	PLC status change log 2: The minute and the second	○	-	-	-	Y	R	0
*SR319	PLC status change log 3: The action number	○	-	-	-	Y	R	0
*SR320	PLC status change log 3: The year and the month	○	-	-	-	Y	R	0
*SR321	PLC status change log 3: The day and the hour	○	-	-	-	Y	R	0
*SR322	PLC status change log 3: The minute and the second	○	-	-	-	Y	R	0
*SR323	PLC status change log 4: The action number	○	-	-	-	Y	R	0
*SR324	PLC status change log 4: The year and the month	○	-	-	-	Y	R	0
*SR325	PLC status change log 4: The day and the hour	○	-	-	-	Y	R	0
*SR326	PLC status change log 4: The minute and the second	○	-	-	-	Y	R	0
*SR327	PLC status change log 5: The action number	○	-	-	-	Y	R	0
*SR328	PLC status change log 5: The year and the month	○	-	-	-	Y	R	0
*SR329	PLC status change log 5: The day and the hour	○	-	-	-	Y	R	0
*SR330	PLC status change log 5: The minute and the second	○	-	-	-	Y	R	0
*SR331	PLC status change log 6: The action number	○	-	-	-	Y	R	0
*SR332	PLC status change log 6: The year and the month	○	-	-	-	Y	R	0
*SR333	PLC status change log 6: The day and the hour	○	-	-	-	Y	R	0
*SR334	PLC status change log 6: The minute and the second	○	-	-	-	Y	R	0
*SR335	PLC status change log 7: The action number	○	-	-	-	Y	R	0
*SR336	PLC status change log 7: The year and the month	○	-	-	-	Y	R	0
*SR337	PLC status change log 7: The day and the hour	○	-	-	-	Y	R	0
*SR338	PLC status change log 7: The minute and the second	○	-	-	-	Y	R	0
*SR339	PLC status change log 8: The action number	○	-	-	-	Y	R	0
*SR340	PLC status change log 8: The year and the month	○	-	-	-	Y	R	0
*SR341	PLC status change log 8: The day and the hour	○	-	-	-	Y	R	0
*SR342	PLC status change log 8: The minute and the second	○	-	-	-	Y	R	0
*SR343	PLC status change log 9: The action number	○	-	-	-	Y	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR344	PLC status change log 9: The year and the month	○	-	-	-	Y	R	0
*SR345	PLC status change log 9: The day and the hour	○	-	-	-	Y	R	0
*SR346	PLC status change log 9: The minute and the second	○	-	-	-	Y	R	0
*SR347	PLC status change log 10: The action number	○	-	-	-	Y	R	0
*SR348	PLC status change log 10: The year and the month	○	-	-	-	Y	R	0
*SR349	PLC status change log 10: The day and the hour	○	-	-	-	Y	R	0
*SR350	PLC status change log 10: The minute and the second	○	-	-	-	Y	R	0
*SR351	PLC status change log 11: The action number	○	-	-	-	Y	R	0
*SR352	PLC status change log 11: The year and the month	○	-	-	-	Y	R	0
*SR353	PLC status change log 11: The day and the hour	○	-	-	-	Y	R	0
*SR354	PLC status change log 11: The minute and the second	○	-	-	-	Y	R	0
*SR355	PLC status change log 12: The action number	○	-	-	-	Y	R	0
*SR356	PLC status change log 12: The year and the month	○	-	-	-	Y	R	0
*SR357	PLC status change log 12: The day and the hour	○	-	-	-	Y	R	0
*SR358	PLC status change log 12: The minute and the second	○	-	-	-	Y	R	0
*SR359	PLC status change log 13: The action number	○	-	-	-	Y	R	0
*SR360	PLC status change log 13: The year and the month	○	-	-	-	Y	R	0
*SR361	PLC status change log 13: The day and the hour	○	-	-	-	Y	R	0
*SR362	PLC status change log 13: The minute and the second	○	-	-	-	Y	R	0
*SR363	PLC status change log 14: The action number	○	-	-	-	Y	R	0
*SR364	PLC status change log 14: The year and the month	○	-	-	-	Y	R	0
*SR365	PLC status change log 14: The day and the hour	○	-	-	-	Y	R	0
*SR366	PLC status change log 14: The minute and the second	○	-	-	-	Y	R	0
*SR367	PLC status change log 15: The action number	○	-	-	-	Y	R	0
*SR368	PLC status change log 15: The year and the month	○	-	-	-	Y	R	0
*SR369	PLC status change log 15: The day and the hour	○	-	-	-	Y	R	0
*SR370	PLC status change log 15: The minute and the second	○	-	-	-	Y	R	0
*SR371	PLC status change log 16: The action number	○	-	-	-	Y	R	0
*SR372	PLC status change log 16: The year and the month	○	-	-	-	Y	R	0
*SR373	PLC status change log 16: The day and the hour	○	-	-	-	Y	R	0
*SR374	PLC status change log 16: The minute and the second	○	-	-	-	Y	R	0
*SR375	PLC status change log 17: The action number	○	-	-	-	Y	R	0
*SR376	PLC status change log 17: The year and the month	○	-	-	-	Y	R	0
*SR377	PLC status change log 17: The day and the hour	○	-	-	-	Y	R	0
*SR378	PLC status change log 17: The minute and the second	○	-	-	-	Y	R	0
*SR379	PLC status change log 18: The action number	○	-	-	-	Y	R	0
*SR380	PLC status change log 18: The year and the month	○	-	-	-	Y	R	0
*SR381	PLC status change log 18: The day and the hour	○	-	-	-	Y	R	0
*SR382	PLC status change log 18: The minute and the second	○	-	-	-	Y	R	0
*SR383	PLC status change log 19: The action number	○	-	-	-	Y	R	0
*SR384	PLC status change log 19: The year and the month	○	-	-	-	Y	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR385	PLC status change log 19: The day and the hour	○	-	-	-	Y	R	0
*SR386	PLC status change log 19: The minute and the second	○	-	-	-	Y	R	0
*SR387	PLC status change log 20: The action number	○	-	-	-	Y	R	0
*SR388	PLC status change log 20: The year and the month	○	-	-	-	Y	R	0
*SR389	PLC status change log 20: The day and the hour	○	-	-	-	Y	R	0
*SR390	PLC status change log 20: The minute and the second	○	-	-	-	Y	R	0
*SR391	Value of the year in the real-time clock (RTC): 00~99 (A.D.)	○	-	-	-	Y	R	0
*SR392	Value of the month in the real-time clock (RTC): 01~12	○	-	-	-	Y	R	1
*SR393	Value of the day in the real-time clock (RTC): 1~31	○	-	-	-	Y	R	1
*SR394	Value of the hour in the real-time clock (RTC): 00~23	○	-	-	-	Y	R	0
*SR395	Value of the minute in the real-time clock (RTC): 00~59	○	-	-	-	Y	R	0
*SR396	Value of the second in the real-time clock (RTC): 00~59	○	-	-	-	Y	R	0
*SR397	Value of the week in the real-time clock (RTC): 1~7	○	-	-	-	Y	R	1
SR407	When the PLC runs, the value in SR407 increases by one every second. SR407 counts from 0 to 32767, and then from -32768 to 0.	○	0	0	-	N	R/W	0
SR408	When the PLC runs, the value in SR408 increases by one every scan cycle. SR408 counts from 0 to 32767, and then from -32768 to 0.	○	0	0	-	N	R/W	0
SR411	The current scan time is stored in SR411 and SR412, and the unit of measurement is 100 microseconds. The value of the millisecond is stored in SR411. (The range is 0~65535.) The value of the microsecond is stored in SR421. (The range is 0~900.). For example, 12 is stored in SR411 and 300 is stored in SR412 when the current scan time is 12.3 milliseconds.	○	0	-	-	N	R	0
SR412		○	0	-	-	N	R	0
SR413	The maximum scan time is stored in SR413 and SR414, and the unit of measurement is 100 microseconds. The value of the millisecond is stored in SR413.	○	0	-	-	N	R	0
SR414								
SR415	The maximum scan time is stored in SR415 and SR416, and the unit of measurement is 100 microseconds. The value of the millisecond is stored in SR415.	○	0	-	-	N	R	0
SR416								
SR421	Duration of the timer interrupt I601 (unit: minutes); the default is 0, meaning the system will follow the settings in HWCONFIG.	○	0	0	-	N	R/W	0
SR422	Duration of the timer interrupt I602 (unit: minutes); the default is 0, meaning the system will follow the settings in HWCONFIG.	○	0	0	-	N	R/W	0
SR423	Duration of the timer interrupt I603 (unit: minutes); the default is 0, meaning the system will follow the settings in HWCONFIG.	○	0	0	-	N	R/W	0
SR424	Duration of the timer interrupt I604 (unit: minutes); the default is 0, meaning the system will follow the settings in HWCONFIG.	○	0	0	-	N	R/W	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR440	MAC address (Ex: 12:34:56:78:9A:BC => SR440=16#1234, SR441=16#5678, SR442=16#9ABC)	○	-	-	-	Y	R	-
SR441		○	-	-	-	Y	R	-
SR442		○	-	-	-	Y	R	-
SR443	The PLC series EX：AS324MTAW15500012 AS➔ SR443 = 16#5341 32 ➔ SR444 = 16#3233 4M ➔ SR445 = 16#4D34 TA ➔ SR446 = 16#4154 W1 ➔ SR447 = 16#3157 55 ➔ SR448 = 16#3535 00 ➔ SR449 = 16#3030 01 ➔ SR450 = 16#3130 2 ➔ SR451 = 16#0032	○	-	-	-	Y	R	-
SR444		○	-	-	-	Y	R	-
SR445		○	-	-	-	Y	R	-
SR446		○	-	-	-	Y	R	-
SR447		○	-	-	-	Y	R	-
SR448		○	-	-	-	Y	R	-
SR449		○	-	-	-	Y	R	-
SR450		○	-	-	-	Y	R	-
SR451		○	-	-	-	Y	R	-
*SR453	If an error occurs during the operation of the memory card, the error code will be recorded.	○	-	-	-	Y	R	0
SR460	The position where Y0.0/axis 1 (Y0.0/Y0.1) is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR461								
SR462	The output mode for the axis 1 (Y0.0/Y0.1)	○	-	-	-	Y	R/W	0
SR463	The starting/ending frequency of the axis 1(Y0.0/Y0.1)	○	-	-	-	Y	R/W	200
SR464	The accelerating time of the axis 1(Y0.0/Y0.1)	○	-	-	-	Y	R/W	200
SR465	The decelerating time of the axis 1(Y0.0/Y0.1)	○	-	-	-	Y	R/W	200
SR466	The JOG frequency of the axis 1(Y0.0/Y0.1)	○	-	-	-	Y	R/W	200
SR467	The number of the axis 1 (Y0.0/Y0.1) in the position planning table that is currently outputting	○	0	0	-	N	R	0
SR468	The numerator value transferred from the machine unit in the axis 1	○	-	-	-	H	R/W	0
SR469	The denominator value transferred from the machine unit in the axis 1	○	-	-	-	H	R/W	0
SR470	The position of the Machine unit in axis 1 (single-precision floating-point values)	○	-	-	-	Y	R	0
SR471								
SR472	The target frequency of the fixed slope in axis 1	○	-	-	-	Y	R	0
SR473								
SR474	The position where Y0.1 is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR475								
SR476	The starting/ending frequency of the Y0.1	○	-	-	-	Y	R/W	200
SR477	The accelerating/decelerating time of the Y0.1.	○	-	-	-	Y	R/W	200
SR480	The position where Y0.2/axis 2 (Y0.2/Y0.3) is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR481								
SR482	The output mode for the axis 2 (Y0.2/Y0.3)	○	-	-	-	Y	R/W	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR483	The starting/ending frequency of the Y0.2/axis 2 (Y0.2/Y0.3)	○	-	-	-	Y	R/W	200
SR484	The accelerating time of the Y0.2/axis 2 (Y0.2/Y0.3)	○	-	-	-	Y	R/W	200
SR485	The decelerating time of the Y0.2/axis 2 (Y0.2/Y0.3)	○	-	-	-	Y	R/W	200
SR486	The JOG frequency of the axis 2 (Y0.2/Y0.3)	○	-	-	-	Y	R/W	200
SR487	The number of the Y0.2/axis 2 (Y0.2/Y0.3) in the position planning table that is currently outputting	○	0	0	-	N	R	0
SR488	The numerator value transferred from the machine unit in the axis 2	○	-	-	-	H	R/W	0
SR489	The denominator value transferred from the machine unit in the axis 2	○	-	-	-	H	R/W	0
SR490 SR491	The position of the Machine unit in axis 2 (single-precision floating-point values)	○	-	-	-	Y	R	0
SR492 SR493	The target frequency of the fixed slope in axis 2	○	-	-	-	Y	R	0
SR494 SR495	The position where Y0.3 is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR496	The starting/ending frequency of the Y0.3	○	-	-	-	Y	R/W	200
SR497	The accelerating/decelerating time of the Y0.3.	○	-	-	-	Y	R/W	200
SR500 SR501	The position where Y0.4/axis 3 (Y0.4/Y0.5) is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR502	The output mode for the axis 3 (Y0.4/Y0.5)	○	-	-	-	Y	R/W	0
SR503	The starting/ending frequency of the Y0.4/axis 3 (Y0.4/Y0.5)	○	-	-	-	Y	R/W	200
SR504	The accelerating time of the Y0.4/axis 3 (Y0.4/Y0.5)	○	-	-	-	Y	R/W	200
SR505	The decelerating time of the Y0.4/axis 3 (Y0.4/Y0.5)	○	-	-	-	Y	R/W	200
SR506	The JOG frequency of the axis 3 (Y0.4/Y0.5)	○	-	-	-	Y	R/W	200
SR507	The number of the axis 3 (Y0.4/Y0.5) in the position planning table that is currently outputting	○	0	0	-	N	R	0
SR508	The numerator value transferred from the machine unit in the axis 3	○	-	-	-	H	R/W	0
SR509	The denominator value transferred from the machine unit in the axis 3	○	-	-	-	H	R/W	0
SR510 SR511	The position of the Machine unit in axis 3 (single-precision floating-point values)	○	-	-	-	Y	R	0
SR512 SR513	The target frequency of the fixed slope in axis 3	○	-	-	-	Y	R	0
SR514 SR515	The position where Y0.5 is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR516	The starting/ending frequency of the Y0.5	○	-	-	-	Y	R/W	200
SR517	The accelerating/decelerating time of the Y0.5	○	-	-	-	Y	R/W	200
SR520	The position where Y0.6/axis 4 (Y0.6/Y0.7) is outputting.	○	-	-	-	Y	R/W	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR521	(unit: number of pulse)							
SR522	The output mode for the axis 4 (Y0.6/Y0.7)	○	-	-	-	Y	R/W	0
SR523	The starting/ending frequency of the Y0.6/axis 4 (Y0.6/Y0.7)	○	-	-	-	Y	R/W	200
SR524	The accelerating time of the Y0.6/axis 4 (Y0.6/Y0.7)	○	-	-	-	Y	R/W	200
SR525	The decelerating time of the Y0.6/axis 4 (Y0.6/Y0.7)	○	-	-	-	Y	R/W	200
SR526	The JOG frequency of the axis 4 (Y0.6/Y0.7)	○	-	-	-	Y	R/W	200
SR527	The number of the axis 4 (Y0.6/Y0.7) in the position planning table that is currently outputting	○	0	0	-	N	R	0
SR528	The numerator value transferred from the machine unit in the axis 4	○	-	-	-	H	R/W	0
SR529	The denominator value transferred from the machine unit in the axis 4	○	-	-	-	H	R/W	0
SR530 SR531	The position of the Machine unit in axis 4 (single-precision floating-point values)	○	-	-	-	Y	R	0
SR532 SR533	The target frequency of the fixed slope in axis 4	○	-	-	-	Y	R	0
SR534 SR535	The position where Y0.7 is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR536	The starting/ending frequency of the Y0.7	○	-	-	-	Y	R/W	200
SR537	The accelerating/decelerating time of the Y0.7	○	-	-	-	Y	R/W	200
SR540 SR541	The position where Y0.8/axis 5 (Y0.8/Y0.9) is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR542	The output mode for the axis 5 (Y0.8/Y0.9)	○	-	-	-	Y	R/W	0
SR543	The starting/ending frequency of the Y0.8/axis 5 (Y0.8/Y0.9)	○	-	-	-	Y	R/W	200
SR544	The accelerating time of the Y0.8/axis 5 (Y0.8/Y0.9)	○	-	-	-	Y	R/W	200
SR545	The decelerating time of the Y0.8/axis 5 (Y0.8/Y0.9)	○	-	-	-	Y	R/W	200
SR546	The JOG frequency of the axis 5 (Y0.8/Y0.9)	○	-	-	-	Y	R/W	200
SR547	The number of the axis 5 (Y0.8/Y0.9) in the position planning table that is currently outputting	○	0	0	-	N	R	0
SR548	The numerator value transferred from the machine unit in the axis 5	○	-	-	-	H	R/W	0
SR549	The denominator value transferred from the machine unit in the axis 5	○	-	-	-	H	R/W	0
SR550 SR551	The position of the Machine unit in axis 5 (single-precision floating-point values)	○	-	-	-	Y	R	0
SR552 SR553	The target frequency of the fixed slope in axis 5	○	-	-	-	Y	R	0
SR554 SR555	The position where Y0.9 is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR556	The starting/ending frequency of the Y0.9	○	-	-	-	Y	R/W	200

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR557	The accelerating/decelerating time of the Y0.9	○	-	-	-	Y	R/W	200
SR560 SR561	The position where Y0.10/axis 6 (Y0.10/Y0.11) is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR562	The output mode for the axis 6 (Y0.10/Y0.11)	○	-	-	-	Y	R/W	0
SR563	The starting/ending frequency of the Y0.10/axis 6 (Y0.10/Y0.11)	○	-	-	-	Y	R/W	200
SR564	The accelerating time of the Y0.10/axis 6 (Y0.10/Y0.11)	○	-	-	-	Y	R/W	200
SR565	The decelerating time of the Y0.10/axis 6 (Y0.10/Y0.11)	○	-	-	-	Y	R/W	200
SR566	The JOG frequency of the axis 6 (Y0.10/Y0.11)	○	-	-	-	Y	R/W	200
SR567	The number of the axis 6 (Y0.10/Y0.11) in the position planning table that is currently outputting	○	0	0	-	N	R	0
SR568	The numerator value transferred from the machine unit in the axis 6	○	-	-	-	H	R/W	0
SR569	The denominator value transferred from the machine unit in the axis 6	○	-	-	-	H	R/W	0
SR570 SR571	The position of the Machine unit in axis 6 (single-precision floating-point values)	○	-	-	-	Y	R	0
SR572 SR573	The target frequency of the fixed slope in axis 6	○	-	-	-	Y	R	0
SR574 SR575	The position where Y0.11 is outputting. (unit: number of pulse)	○	-	-	-	Y	R/W	0
SR576	The starting/ending frequency of the Y0.11	○	-	-	-	Y	R/W	200
SR577	The accelerating/decelerating time of the Y0.11	○	-	-	-	Y	R/W	200
*SR580 *SR581	The position of the positive limit for axis 1 (Y0.0/Y0.1) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR582 *SR583	The position of the negative limit for axis 1 (Y0.0/Y0.1) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR584 *SR585	The position of the positive limit for axis 2 (Y0.2/Y0.3) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR586 *SR587	The position of the negative limit for axis 2 (Y0.2/Y0.3) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR588 *SR589	The position of the positive limit for axis 3(Y0.4/Y0.5) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR590 *SR591	The position of the negative limit for axis 3(Y0.4/Y0.5) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR592 *SR593	The position of the positive limit for axis 4(Y0.6/Y0.7) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR594 *SR595	The position of the negative limit for axis 4(Y0.6/Y0.7) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR596 *SR597	The position of the positive limit for axis 5(Y0.8/Y0.9) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR598 *SR599	The position of the negative limit for axis 5(Y0.8/Y0.9) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR600 *SR601	The position of the positive limit for axis 6(Y0.10/Y0.11) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR602 *SR603	The position of the negative limit for axis 6(Y0.10/Y0.11) in ISPSoft (unit: number of pulse)	○	-	-	-	H	R/W	0
*SR604	The S curve mode for axis 1 (Y0.0/Y0.1)	○	0	-	-	N	R/W	0
*SR605	The S curve mode for axis 2 (Y0.2/Y0.3)	○	0	-	-	N	R/W	0
*SR606	The S curve mode for axis 3 (Y0.4/Y0.5)	○	0	-	-	N	R/W	0
*SR607	The S curve mode for axis 4 (Y0.6/Y0.7)	○	0	-	-	N	R/W	0
*SR608	The S curve mode for axis 5 (Y0.8/Y0.9)	○	0	-	-	N	R/W	0
*SR609	The S curve mode for axis 6 (Y0.10/Y0.11)	○	0	-	-	N	R/W	0
SR610 SR611	The current output speed of axis 1 (Y0.0/Y0.1) (unit: Hz)	○	0	0	0	N	R	0
SR612 SR613	The current output speed of axis 2 (Y0.2/Y0.3) (unit: Hz)	○	0	0	0	N	R	0
SR614 SR615	The current output speed of axis 3 (Y0.4/Y0.5) (unit: Hz)	○	0	0	0	N	R	0
SR616 SR617	The current output speed of axis 4 (Y0.6/Y0.7) (unit: Hz)	○	0	0	0	N	R	0
SR618 SR619	The current output speed of axis 5 (Y0.8/Y0.9) (unit: Hz)	○	0	0	0	N	R	0
SR620 SR621	The current output speed of axis 6 (Y0.10/Y0.11) (unit: Hz)	○	0	0	0	N	R	0
SR623	The condition of the external interrupt : the input points of X0.0~X0.15 are falling-edge triggered	○	FFFF	FFFF	-	N	R	FFFF
SR624	The condition of the external interrupt : the input points of X0.0~X0.15 are rising-edge triggered	○	FFFF	FFFF	-	N	R	FFFF
SR625	The condition of the high-speed comparator interrupt I200~I233	○	FFFF	FFFF	-	N	R	FFFF
SR626	The condition of the high-speed comparator interrupt I240~I253	○	FFFF	FFFF	-	N	R	FFFF
SR627	The condition of the high-speed comparator interrupt I260~I267	○	FFFF	FFFF	-	N	R	FFFF
SR628	The condition of the communication interrupts I300~I307	○	FFFF	FFFF	-	N	R	FFFF
SR629	The condition of the output interrupts I500~I505	○	FFFF	FFFF	-	N	R	FFFF
SR630	The condition of the output interrupts I510~I519	○	FFFF	FFFF	-	N	R	FFFF
SR632	The condition of the timer interrupts I601~I604	○	FFFF	FFFF	-	N	R	FFFF
SR633	The condition of the extension module interrupts I400~I415	○	FFFF	FFFF	-	N	R	FFFF
SR634	The condition of the extension module interrupts	○	FFFF	FFFF	-	N	R	FFFF

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	I416~I431							
SR658	The number of the Delta CANopen communication axis from Delta servo which has a communication error	○	0	-	-	N	R	0
SR659	Delta CANopen communication error	○	0	-	-	N	R	0
SR661	The PR command of Delta CANopen communication axis 1 from Delta servo	○	0	-	-	N	R	0
SR662	The PR command of Delta CANopen communication axis 2 from Delta servo	○	0	-	-	N	R	0
SR663	The PR command of Delta CANopen communication axis 3 from Delta servo	○	0	-	-	N	R	0
SR664	The PR command of Delta CANopen communication axis 4 from Delta servo	○	0	-	-	N	R	0
SR665	The PR command of Delta CANopen communication axis 5 from Delta servo	○	0	-	-	N	R	0
SR666	The PR command of Delta CANopen communication axis 6 from Delta servo	○	0	-	-	N	R	0
SR667	The PR command of Delta CANopen communication axis 7 from Delta servo	○	0	-	-	N	R	0
SR668	The PR command of Delta CANopen communication axis 8 from Delta servo	○	0	-	-	N	R	0
SR671	The alarm code of Delta CANopen communication axis 1 from Delta servo	○	0	-	-	N	R	0
SR672	The alarm code of Delta CANopen communication axis 2 from Delta servo	○	0	-	-	N	R	0
SR673	The alarm code of Delta CANopen communication axis 3 from Delta servo	○	0	-	-	N	R	0
SR674	The alarm code of Delta CANopen communication axis 4 from Delta servo	○	0	-	-	N	R	0
SR675	The alarm code of Delta CANopen communication axis 5 from Delta servo	○	0	-	-	N	R	0
SR676	The alarm code of Delta CANopen communication axis 6 from Delta servo	○	0	-	-	N	R	0
SR677	The alarm code of Delta CANopen communication axis 7 from Delta servo	○	0	-	-	N	R	0
SR678	The alarm code of Delta CANopen communication axis 8 from Delta servo	○	0	-	-	N	R	0
SR681	The DO state of Delta CANopen communication axis 1 from Delta servo	○	0	-	-	N	R	0
SR682	The DO state of Delta CANopen communication axis 2 from Delta servo	○	0	-	-	N	R	0
SR683	The DO state of Delta CANopen communication axis 3 from Delta servo	○	0	-	-	N	R	0
SR684	The DO state of Delta CANopen communication axis 4 from Delta servo	○	0	-	-	N	R	0
SR685	The DO state of Delta CANopen communication axis 5 from Delta servo	○	0	-	-	N	R	0
SR686	The DO state of Delta CANopen communication axis 6 from Delta servo	○	0	-	-	N	R	0
SR687	The DO state of Delta CANopen communication axis 7 from Delta servo	○	0	-	-	N	R	0
SR688	The DO state of Delta CANopen communication axis 8 from Delta servo	○	0	-	-	N	R	0
SR691 SR692	The current position of Delta CANopen communication axis 1 from Delta servo (32-bit)	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR693 SR694	The current position of Delta CANopen communication axis 2 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR695 SR696	The current position of Delta CANopen communication axis 3 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR697 SR698	The current position of Delta CANopen communication axis 4 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR699 SR700	The current position of Delta CANopen communication axis 5 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR701 SR702	The current position of Delta CANopen communication axis 6 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR703 SR704	The current position of Delta CANopen communication axis 7 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR705 SR706	The current position of Delta CANopen communication axis 8 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR711 SR712	The target position of Delta CANopen communication axis 1 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR713 SR714	The target position of Delta CANopen communication axis 2 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR715 SR716	The target position of Delta CANopen communication axis 3 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR717 SR718	The target position of Delta CANopen communication axis 4 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR719 SR720	The target position of Delta CANopen communication axis 5 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR721 SR722	The target position of Delta CANopen communication axis 6 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR723 SR724	The target position of Delta CANopen communication axis 7 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR725 SR726	The target position of Delta CANopen communication axis 8 from Delta servo (32-bit)	○	0	-	-	N	R	0
SR820	The code of the state of the master/slave in CANopen DS301 communicaiton	○	-	-	-	Y	R	0
SR821	The code of the CANopen DS301 version	○	-	-	-	Y	R	-
SR822	CANopen communication baudrate (unit: 1kbps)	○	-	-	-	H	R	125
SR825	The code of the master state in CANopen DS301 communication	○	0	-	-	N	R	0
SR826	The state of slave ID 1~16 in CANopen DS301 communication	○	0	-	-	N	R	0
SR827	The state of slave ID 17~32 in CANopen DS301 communication	○	0	-	-	N	R	0
SR828	The state of slave ID 33~48 in CANopen DS301 communication	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR829	The state of slave ID 49~64 in CANopen DS301 communication	○	0	-	-	N	R	0
SR830	The state of slave ID 1 in CANopen DS301 communication	○	0	-	-	N	R	0
SR831	The state of slave ID 2 in CANopen DS301 communication	○	0	-	-	N	R	0
SR832	The state of slave ID 3 in CANopen DS301 communication	○	0	-	-	N	R	0
SR833	The state of slave ID 4 in CANopen DS301 communication	○	0	-	-	N	R	0
SR834	The state of slave ID 5 in CANopen DS301 communication	○	0	-	-	N	R	0
SR835	The state of slave ID 6 in CANopen DS301 communication	○	0	-	-	N	R	0
SR836	The state of slave ID 7 in CANopen DS301 communication	○	0	-	-	N	R	0
SR837	The state of slave ID 8 in CANopen DS301 communication	○	0	-	-	N	R	0
SR838	The state of slave ID 9 in CANopen DS301 communication	○	0	-	-	N	R	0
SR839	The state of slave ID 10 in CANopen DS301 communication	○	0	-	-	N	R	0
SR840	The state of slave ID 11 in CANopen DS301 communication	○	0	-	-	N	R	0
SR841	The state of slave ID 12 in CANopen DS301 communication	○	0	-	-	N	R	0
SR842	The state of slave ID 13 in CANopen DS301 communication	○	0	-	-	N	R	0
SR843	The state of slave ID 14 in CANopen DS301 communication	○	0	-	-	N	R	0
SR844	The state of slave ID 15 in CANopen DS301 communication	○	0	-	-	N	R	0
SR845	The state of slave ID 16 in CANopen DS301 communication	○	0	-	-	N	R	0
SR846	The state of slave ID 17 in CANopen DS301 communication	○	0	-	-	N	R	0
SR847	The state of slave ID 18 in CANopen DS301 communication	○	0	-	-	N	R	0
SR848	The state of slave ID 19 in CANopen DS301 communication	○	0	-	-	N	R	0
SR849	The state of slave ID 20 in CANopen DS301 communication	○	0	-	-	N	R	0
SR850	The state of slave ID 21 in CANopen DS301 communication	○	0	-	-	N	R	0
SR851	The state of slave ID 22 in CANopen DS301 communication	○	0	-	-	N	R	0
SR852	The state of slave ID 23 in CANopen DS301 communication	○	0	-	-	N	R	0
SR853	The state of slave ID 24 in CANopen DS301 communication	○	0	-	-	N	R	0
SR854	The state of slave ID 25 in CANopen DS301 communication	○	0	-	-	N	R	0
SR855	The state of slave ID 26 in CANopen DS301 communication	○	0	-	-	N	R	0
SR856	The state of slave ID 27 in CANopen DS301 communication	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR857	The state of slave ID 28 in CANopen DS301 communication	○	0	-	-	N	R	0
SR858	The state of slave ID 29 in CANopen DS301 communication	○	0	-	-	N	R	0
SR859	The state of slave ID 30 in CANopen DS301 communication	○	0	-	-	N	R	0
SR860	The state of slave ID 31 in CANopen DS301 communication	○	0	-	-	N	R	0
SR861	The state of slave ID 32 in CANopen DS301 communication	○	0	-	-	N	R	0
SR862	The state of slave ID 33 in CANopen DS301 communication	○	0	-	-	N	R	0
SR863	The state of slave ID 34 in CANopen DS301 communication	○	0	-	-	N	R	0
SR864	The state of slave ID 35 in CANopen DS301 communication	○	0	-	-	N	R	0
SR865	The state of slave ID 36in CANopen DS301 communication	○	0	-	-	N	R	0
SR866	The state of slave ID 37 in CANopen DS301 communication	○	0	-	-	N	R	0
SR867	The state of slave ID 38 in CANopen DS301 communication	○	0	-	-	N	R	0
SR868	The state of slave ID 39 in CANopen DS301 communication	○	0	-	-	N	R	0
SR869	The state of slave ID 40 in CANopen DS301 communication	○	0	-	-	N	R	0
SR870	The state of slave ID 41 in CANopen DS301 communication	○	0	-	-	N	R	0
SR871	The state of slave ID 42 in CANopen DS301 communication	○	0	-	-	N	R	0
SR872	The state of slave ID 43 in CANopen DS301 communication	○	0	-	-	N	R	0
SR873	The state of slave ID 44 in CANopen DS301 communication	○	0	-	-	N	R	0
SR874	The state of slave ID 45 in CANopen DS301 communication	○	0	-	-	N	R	0
SR875	The state of slave ID 46 in CANopen DS301 communication	○	0	-	-	N	R	0
SR876	The state of slave ID 47 in CANopen DS301 communication	○	0	-	-	N	R	0
SR877	The state of slave ID 48 in CANopen DS301 communication	○	0	-	-	N	R	0
SR878	The state of slave ID 49 in CANopen DS301 communication	○	0	-	-	N	R	0
SR879	The state of slave ID 50 in CANopen DS301 communication	○	0	-	-	N	R	0
SR880	The state of slave ID 51 in CANopen DS301 communication	○	0	-	-	N	R	0
SR881	The state of slave ID 52 in CANopen DS301 communication	○	0	-	-	N	R	0
SR882	The state of slave ID 53 in CANopen DS301 communication	○	0	-	-	N	R	0
SR883	The state of slave ID 54 in CANopen DS301 communication	○	0	-	-	N	R	0
SR884	The state of slave ID 55 in CANopen DS301 communication	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR885	The state of slave ID 56 in CANopen DS301 communication	○	0	-	-	N	R	0
SR886	The state of slave ID 57 in CANopen DS301 communication	○	0	-	-	N	R	0
SR887	The state of slave ID 58 in CANopen DS301 communication	○	0	-	-	N	R	0
SR888	The state of slave ID 59 in CANopen DS301 communication	○	0	-	-	N	R	0
SR889	The state of slave ID 60 in CANopen DS301 communication	○	0	-	-	N	R	0
SR890	The state of slave ID 61 in CANopen DS301 communication	○	0	-	-	N	R	0
SR891	The state of slave ID 62 in CANopen DS301 communication	○	0	-	-	N	R	0
SR892	The state of slave ID 63 in CANopen DS301 communication	○	0	-	-	N	R	0
SR893	The state of slave ID 64 in CANopen DS301 communication	○	0	-	-	N	R	0
SR900 SR901	The number of the samplings in the data logger (32-bit)	○	0	-	-	N	R	0
SR902	The code for the executions of data logger and the memory card (need to work with SM456), e.g. H5AA5: write the sampling data from the data logger into the memory card.	○	0	-	-	N	R/W	0
SR1000 SR1001	Ethernet IP address (32-bit)	○	-	-	-	H	R/W	0
SR1002 SR1003	Ethernet netmask address (32-bit)	○	-	-	-	H	R/W	0
SR1004 SR1005	Ethernet gateway address (32-bit)	○	-	-	-	H	R/W	0
SR1006	Time for which the TCP connection has been persistent	○	-	-	-	H	R/W	30
SR1007	Ethernet transmission speed	○	0	-	-	N	R	0
SR1009	Number of TCP connections	○	0	-	-	N	R	0
SR1010	A specific time when to resend via the TCP connection	○	-	-	-	N	R/W	20
SR1011	The connection number of MODBUS/TCP Server	○	0	-	-	N	R	0
SR1012	The connection number of MODBUS/TCP Client	○	0	-	-	N	R	0
SR1013	The connection number of EtherNet/IP Adapter	○	0	-	-	N	R	0
SR1014	The connection number of EtherNet/IP Scanner	○	0	-	-	N	R	0
SR1020	The state of the EtherNet/IP connection 1	○	0	-	-	N	R	0
SR1021	The state of the EtherNet/IP connection 2	○	0	-	-	N	R	0
SR1022	The state of the EtherNet/IP connection 3	○	0	-	-	N	R	0
SR1023	The state of the EtherNet/IP connection 4	○	0	-	-	N	R	0
SR1024	The state of the EtherNet/IP connection 5	○	0	-	-	N	R	0
SR1025	The state of the EtherNet/IP connection 6	○	0	-	-	N	R	0
SR1026	The state of the EtherNet/IP connection 7	○	0	-	-	N	R	0
SR1027	The state of the EtherNet/IP connection 8	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR1028	The state of the EtherNet/IP connection 9	○	0	-	-	N	R	0
SR1029	The state of the EtherNet/IP connection 10	○	0	-	-	N	R	0
SR1030	The state of the EtherNet/IP connection 11	○	0	-	-	N	R	0
SR1031	The state of the EtherNet/IP connection 12	○	0	-	-	N	R	0
SR1032	The state of the EtherNet/IP connection 13	○	0	-	-	N	R	0
SR1033	The state of the EtherNet/IP connection 14	○	0	-	-	N	R	0
SR1034	The state of the EtherNet/IP connection 15	○	0	-	-	N	R	0
SR1035	The state of the EtherNet/IP connection 16	○	0	-	-	N	R	0
SR1036	The state of the EtherNet/IP connection 17	○	0	-	-	N	R	0
SR1037	The state of the EtherNet/IP connection 18	○	0	-	-	N	R	0
SR1038	The state of the EtherNet/IP connection 19	○	0	-	-	N	R	0
SR1039	The state of the EtherNet/IP connection 20	○	0	-	-	N	R	0
SR1040	The state of the EtherNet/IP connection 21	○	0	-	-	N	R	0
SR1041	The state of the EtherNet/IP connection 22	○	0	-	-	N	R	0
SR1042	The state of the EtherNet/IP connection 23	○	0	-	-	N	R	0
SR1043	The state of the EtherNet/IP connection 24	○	0	-	-	N	R	0
SR1044	The state of the EtherNet/IP connection 25	○	0	-	-	N	R	0
SR1045	The state of the EtherNet/IP connection 26	○	0	-	-	N	R	0
SR1046	The state of the EtherNet/IP connection 27	○	0	-	-	N	R	0
SR1047	The state of the EtherNet/IP connection 28	○	0	-	-	N	R	0
SR1048	The state of the EtherNet/IP connection 29	○	0	-	-	N	R	0
SR1049	The state of the EtherNet/IP connection 30	○	0	-	-	N	R	0
SR1050	The state of the EtherNet/IP connection 31	○	0	-	-	N	R	0
SR1051	The state of the EtherNet/IP connection 32	○	0	-	-	N	R	0
SR1052	Refreshing time for the EtherNet/IP connection 1	○	0	-	-	N	R	0
SR1053	Refreshing time for the EtherNet/IP connection 2	○	0	-	-	N	R	0
SR1054	Refreshing time for the EtherNet/IP connection 3	○	0	-	-	N	R	0
SR1055	Refreshing time for the EtherNet/IP connection 4	○	0	-	-	N	R	0
SR1056	Refreshing time for the EtherNet/IP connection 5	○	0	-	-	N	R	0
SR1057	Refreshing time for the EtherNet/IP connection 6	○	0	-	-	N	R	0
SR1058	Refreshing time for the EtherNet/IP connection 7	○	0	-	-	N	R	0
SR1059	Refreshing time for the EtherNet/IP connection 8	○	0	-	-	N	R	0
SR1060	Refreshing time for the EtherNet/IP connection 9	○	0	-	-	N	R	0
SR1061	Refreshing time for the EtherNet/IP connection 10	○	0	-	-	N	R	0
SR1062	Refreshing time for the EtherNet/IP connection 11	○	0	-	-	N	R	0
SR1063	Refreshing time for the EtherNet/IP connection 12	○	0	-	-	N	R	0
SR1064	Refreshing time for the EtherNet/IP connection 13	○	0	-	-	N	R	0
SR1065	Refreshing time for the EtherNet/IP connection 14	○	0	-	-	N	R	0
SR1066	Refreshing time for the EtherNet/IP connection 15	○	0	-	-	N	R	0
SR1067	Refreshing time for the EtherNet/IP connection 16	○	0	-	-	N	R	0
SR1068	Refreshing time for the EtherNet/IP connection 17	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR1069	Refreshing time for the EtherNet/IP connection 18	○	0	-	-	N	R	0
SR1070	Refreshing time for the EtherNet/IP connection 19	○	0	-	-	N	R	0
SR1071	Refreshing time for the EtherNet/IP connection 20	○	0	-	-	N	R	0
SR1072	Refreshing time for the EtherNet/IP connection 21	○	0	-	-	N	R	0
SR1073	Refreshing time for the EtherNet/IP connection 22	○	0	-	-	N	R	0
SR1074	Refreshing time for the EtherNet/IP connection 23	○	0	-	-	N	R	0
SR1075	Refreshing time for the EtherNet/IP connection 24	○	0	-	-	N	R	0
SR1076	Refreshing time for the EtherNet/IP connection 25	○	0	-	-	N	R	0
SR1077	Refreshing time for the EtherNet/IP connection 26	○	0	-	-	N	R	0
SR1078	Refreshing time for the EtherNet/IP connection 27	○	0	-	-	N	R	0
SR1079	Refreshing time for the EtherNet/IP connection 28	○	0	-	-	N	R	0
SR1080	Refreshing time for the EtherNet/IP connection 29	○	0	-	-	N	R	0
SR1081	Refreshing time for the EtherNet/IP connection 30	○	0	-	-	N	R	0
SR1082	Refreshing time for the EtherNet/IP connection 31	○	0	-	-	N	R	0
SR1083	Refreshing time for the EtherNet/IP connection 32	○	0	-	-	N	R	0
SR1100 SR1101	The value of the input packet counter (32-bit)	○	0	-	-	N	R	0
SR1102 SR1103	The value of the input octet counter (32-bit)	○	0	-	-	N	R	0
SR1104 SR1105	The value of the output packet counter (32-bit)	○	0	-	-	N	R	0
SR1106 SR1107	The value of the output octet counter (32-bit)	○	0	-	-	N	R	0
SR1116	Email counter	○	0	-	-	N	R	0
SR1117	Email error counter	○	0	-	-	N	R	0
*SR1120	The actual connection time for data exchange via the Ethernet connection 1	○	0	-	-	N	R	0
*SR1121	The actual connection time for data exchange via the Ethernet connection 2	○	0	-	-	N	R	0
*SR1122	The actual connection time for data exchange via the Ethernet connection 3	○	0	-	-	N	R	0
*SR1123	The actual connection time for data exchange via the Ethernet connection 4	○	0	-	-	N	R	0
*SR1124	The actual connection time for data exchange via the Ethernet connection 5	○	0	-	-	N	R	0
*SR1125	The actual connection time for data exchange via the Ethernet connection 6	○	0	-	-	N	R	0
*SR1126	The actual connection time for data exchange via the Ethernet connection 7	○	0	-	-	N	R	0
*SR1127	The actual connection time for data exchange via the Ethernet connection 8	○	0	-	-	N	R	0
*SR1128	The actual connection time for data exchange via the Ethernet connection 9	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR1129	The actual connection time for data exchange via the Ethernet connection 10	○	0	-	-	N	R	0
*SR1130	The actual connection time for data exchange via the Ethernet connection 11	○	0	-	-	N	R	0
*SR1131	The actual connection time for data exchange via the Ethernet connection 12	○	0	-	-	N	R	0
*SR1132	The actual connection time for data exchange via the Ethernet connection 13	○	0	-	-	N	R	0
*SR1133	The actual connection time for data exchange via the Ethernet connection 14	○	0	-	-	N	R	0
*SR1134	The actual connection time for data exchange via the Ethernet connection 15	○	0	-	-	N	R	0
*SR1135	The actual connection time for data exchange via the Ethernet connection 16	○	0	-	-	N	R	0
*SR1136	The actual connection time for data exchange via the Ethernet connection 17	○	0	-	-	N	R	0
*SR1137	The actual connection time for data exchange via the Ethernet connection 18	○	0	-	-	N	R	0
*SR1138	The actual connection time for data exchange via the Ethernet connection 19	○	0	-	-	N	R	0
*SR1139	The actual connection time for data exchange via the Ethernet connection 20	○	0	-	-	N	R	0
*SR1140	The actual connection time for data exchange via the Ethernet connection 21	○	0	-	-	N	R	0
*SR1141	The actual connection time for data exchange via the Ethernet connection 22	○	0	-	-	N	R	0
*SR1142	The actual connection time for data exchange via the Ethernet connection 23	○	0	-	-	N	R	0
*SR1143	The actual connection time for data exchange via the Ethernet connection 24	○	0	-	-	N	R	0
*SR1144	The actual connection time for data exchange via the Ethernet connection 25	○	0	-	-	N	R	0
*SR1145	The actual connection time for data exchange via the Ethernet connection 26	○	0	-	-	N	R	0
*SR1146	The actual connection time for data exchange via the Ethernet connection 27	○	0	-	-	N	R	0
*SR1147	The actual connection time for data exchange via the Ethernet connection 28	○	0	-	-	N	R	0
*SR1148	The actual connection time for data exchange via the Ethernet connection 29	○	0	-	-	N	R	0
*SR1149	The actual connection time for data exchange via the Ethernet connection 30	○	0	-	-	N	R	0
*SR1150	The actual connection time for data exchange via the Ethernet connection 31	○	0	-	-	N	R	0
*SR1151	The actual connection time for data exchange via the Ethernet connection 32	○	0	-	-	N	R	0
*SR1152	The error code for data exchange via the Ethernet	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	connection 1							
*SR1153	The error code for data exchange via the Ethernet connection 2	○	0	-	-	N	R	0
*SR1154	The error code for data exchange via the Ethernet connection 3	○	0	-	-	N	R	0
*SR1155	The error code for data exchange via the Ethernet connection 4	○	0	-	-	N	R	0
*SR1156	The error code for data exchange via the Ethernet connection 5	○	0	-	-	N	R	0
*SR1157	The error code for data exchange via the Ethernet connection 6	○	0	-	-	N	R	0
*SR1158	The error code for data exchange via the Ethernet connection 7	○	0	-	-	N	R	0
*SR1159	The error code for data exchange via the Ethernet connection 8	○	0	-	-	N	R	0
*SR1160	The error code for data exchange via the Ethernet connection 9	○	0	-	-	N	R	0
*SR1161	The error code for data exchange via the Ethernet connection 10	○	0	-	-	N	R	0
*SR1162	The error code for data exchange via the Ethernet connection 11	○	0	-	-	N	R	0
*SR1163	The error code for data exchange via the Ethernet connection 12	○	0	-	-	N	R	0
*SR1164	The error code for data exchange via the Ethernet connection 13	○	0	-	-	N	R	0
*SR1165	The error code for data exchange via the Ethernet connection 14	○	0	-	-	N	R	0
*SR1166	The error code for data exchange via the Ethernet connection 15	○	0	-	-	N	R	0
*SR1167	The error code for data exchange via the Ethernet connection 16	○	0	-	-	N	R	0
*SR1168	The error code for data exchange via the Ethernet connection 17	○	0	-	-	N	R	0
*SR1169	The error code for data exchange via the Ethernet connection 18	○	0	-	-	N	R	0
*SR1170	The error code for data exchange via the Ethernet connection 19	○	0	-	-	N	R	0
*SR1171	The error code for data exchange via the Ethernet connection 20	○	0	-	-	N	R	0
*SR1172	The error code for data exchange via the Ethernet connection 21	○	0	-	-	N	R	0
*SR1173	The error code for data exchange via the Ethernet connection 22	○	0	-	-	N	R	0
*SR1174	The error code for data exchange via the Ethernet connection 23	○	0	-	-	N	R	0
*SR1175	The error code for data exchange via the Ethernet connection 24	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR1176	The error code for data exchange via the Ethernet connection 25	○	0	-	-	N	R	0
*SR1177	The error code for data exchange via the Ethernet connection 26	○	0	-	-	N	R	0
*SR1178	The error code for data exchange via the Ethernet connection 27	○	0	-	-	N	R	0
*SR1179	The error code for data exchange via the Ethernet connection 28	○	0	-	-	N	R	0
*SR1180	The error code for data exchange via the Ethernet connection 29	○	0	-	-	N	R	0
*SR1181	The error code for data exchange via the Ethernet connection 30	○	0	-	-	N	R	0
*SR1182	The error code for data exchange via the Ethernet connection 31	○	0	-	-	N	R	0
*SR1183	The error code for data exchange via the Ethernet connection 32	○	0	-	-	N	R	0
SR1318	Socket input counter	○	0	-	-	N	R	0
SR1319	Socket output counter	○	0	-	-	N	R	0
SR1320	Socket error counter	○	0	-	-	N	R	0
*SR1335	The actual cycle time of connection 1~32 for data exchange via COM1	○	0	-	-	N	R	0
*SR1336	The number of connection that is currently performing a cyclical data exchange via COM1	○	0	-	-	N	R	0
*SR1340	The error code for data exchange via the COM1 connection 1	○	0	-	-	N	R	0
*SR1341	The error code for data exchange via the COM1 connection 2	○	0	-	-	N	R	0
*SR1342	The error code for data exchange via the COM1 connection 3	○	0	-	-	N	R	0
*SR1343	The error code for data exchange via the COM1 connection 4	○	0	-	-	N	R	0
*SR1344	The error code for data exchange via the COM1 connection 5	○	0	-	-	N	R	0
*SR1345	The error code for data exchange via the COM1 connection 6	○	0	-	-	N	R	0
*SR1346	The error code for data exchange via the COM1 connection 7	○	0	-	-	N	R	0
*SR1347	The error code for data exchange via the COM1 connection 8	○	0	-	-	N	R	0
*SR1348	The error code for data exchange via the COM1 connection 9	○	0	-	-	N	R	0
*SR1349	The error code for data exchange via the COM1 connection 10	○	0	-	-	N	R	0
*SR1350	The error code for data exchange via the COM1 connection 11	○	0	-	-	N	R	0
*SR1351	The error code for data exchange via the COM1 connection 12	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR1352	The error code for data exchange via the COM1 connection 13	○	0	-	-	N	R	0
*SR1353	The error code for data exchange via the COM1 connection 14	○	0	-	-	N	R	0
*SR1354	The error code for data exchange via the COM1 connection 15	○	0	-	-	N	R	0
*SR1355	The error code for data exchange via the COM1 connection 16	○	0	-	-	N	R	0
*SR1356	The error code for data exchange via the COM1 connection 17	○	0	-	-	N	R	0
*SR1357	The error code for data exchange via the COM1 connection 18	○	0	-	-	N	R	0
*SR1358	The error code for data exchange via the COM1 connection 19	○	0	-	-	N	R	0
*SR1359	The error code for data exchange via the COM1 connection 20	○	0	-	-	N	R	0
*SR1360	The error code for data exchange via the COM1 connection 21	○	0	-	-	N	R	0
*SR1361	The error code for data exchange via the COM1 connection 22	○	0	-	-	N	R	0
*SR1362	The error code for data exchange via the COM1 connection 23	○	0	-	-	N	R	0
*SR1363	The error code for data exchange via the COM1 connection 24	○	0	-	-	N	R	0
*SR1364	The error code for data exchange via the COM1 connection 25	○	0	-	-	N	R	0
*SR1365	The error code for data exchange via the COM1 connection 26	○	0	-	-	N	R	0
*SR1366	The error code for data exchange via the COM1 connection 27	○	0	-	-	N	R	0
*SR1367	The error code for data exchange via the COM1 connection 28	○	0	-	-	N	R	0
*SR1368	The error code for data exchange via the COM1 connection 29	○	0	-	-	N	R	0
*SR1369	The error code for data exchange via the COM1 connection 30	○	0	-	-	N	R	0
*SR1370	The error code for data exchange via the COM1 connection 31	○	0	-	-	N	R	0
*SR1371	The error code for data exchange via the COM1 connection 32	○	0	-	-	N	R	0
*SR1375	The actual cycle time of connection 1~32 for data exchange via COM2	○	0	-	-	N	R	0
*SR1376	The number of connection that is currently performing a cyclical data exchange via COM2	○	0	-	-	N	R	0
*SR1380	The error code for data exchange via the COM2 connection 1	○	0	-	-	N	R	0
SR1381	The error code for data exchange via the COM2 connection 2	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR1382	The error code for data exchange via the COM2 connection 3	○	0	-	-	N	R	0
SR1383	The error code for data exchange via the COM2 connection 4	○	0	-	-	N	R	0
SR1384	The error code for data exchange via the COM2 connection 5	○	0	-	-	N	R	0
SR1385	The error code for data exchange via the COM2 connection 6	○	0	-	-	N	R	0
SR1386	The error code for data exchange via the COM2 connection 7	○	0	-	-	N	R	0
SR1387	The error code for data exchange via the COM2 connection 8	○	0	-	-	N	R	0
SR1388	The error code for data exchange via the COM2 connection 9	○	0	-	-	N	R	0
SR1389	The error code for data exchange via the COM2 connection 10	○	0	-	-	N	R	0
SR1390	The error code for data exchange via the COM2 connection 11	○	0	-	-	N	R	0
SR1391	The error code for data exchange via the COM2 connection 12	○	0	-	-	N	R	0
SR1392	The error code for data exchange via the COM2 connection 13	○	0	-	-	N	R	0
SR1393	The error code for data exchange via the COM2 connection 14	○	0	-	-	N	R	0
SR1394	The error code for data exchange via the COM2 connection 15	○	0	-	-	N	R	0
SR1395	The error code for data exchange via the COM2 connection 16	○	0	-	-	N	R	0
SR1396	The error code for data exchange via the COM2 connection 17	○	0	-	-	N	R	0
SR1397	The error code for data exchange via the COM2 connection 18	○	0	-	-	N	R	0
SR1398	The error code for data exchange via the COM2 connection 19	○	0	-	-	N	R	0
SR1399	The error code for data exchange via the COM2 connection 20	○	0	-	-	N	R	0
SR1400	The error code for data exchange via the COM2 connection 21	○	0	-	-	N	R	0
SR1401	The error code for data exchange via the COM2 connection 22	○	0	-	-	N	R	0
SR1402	The error code for data exchange via the COM2 connection 23	○	0	-	-	N	R	0
SR1403	The error code for data exchange via the COM2 connection 24	○	0	-	-	N	R	0
SR1404	The error code for data exchange via the COM2 connection 25	○	0	-	-	N	R	0
SR1405	The error code for data exchange via the COM2 connection 26	○	0	-	-	N	R	0

SR	Function	AS300 Series	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR1406	The error code for data exchange via the COM2 connection 27	○	0	-	-	N	R	0
SR1407	The error code for data exchange via the COM2 connection 28	○	0	-	-	N	R	0
SR1408	The error code for data exchange via the COM2 connection 29	○	0	-	-	N	R	0
SR1409	The error code for data exchange via the COM2 connection 30	○	0	-	-	N	R	0
SR1410	The error code for data exchange via the COM2 connection 31	○	0	-	-	N	R	0
SR1411	The error code for data exchange via the COM2 connection 32	○	0	-	-	N	R	0

註 1：*之 SR 請參考 SM/SR 補充說明

註 2：SR 中所述之「通訊卡」係指 AS-F232、AS-F422 及 AS-F485。

2.2.15 Refresh Time of Special Data Registers

Special data register	Refresh time
SR0~SR2	The register is refreshed when the program is executed in error.
SR4~SR6	The register is refreshed when there is a grammar check error
SR8~SR9	The register is refreshed when there is a watchdog timer error.
SR23	The register is refreshed when there is a watchdog timer error.
SR28	The register is refreshed when the last output number of which the instruction high speed output is used repeatedly.
SR32	The register is refreshed once when there is an error. -1 means no error occurred.
SR36	The register is refreshed by users. Users set the flag SM36 to ON and the system will save the data to the memory card. After the saving is complete, the system resets it to OFF automatically.
SR40~SR161	The register is refreshed when there is an error.
SR162~SR163	After the production of the PLC is complete, every time the PLC is supplied with power for more than 1 minute, the number increases by 1.
SR166~SR171	The register is refreshed by the system.
SR172~SR175	The register is refreshed by users.
SR176~SR179	The register is refreshed according to the settings in HWCONFIG.
SR180	The register is refreshed when the PLC is powered-on and powered-off.
SR182~SR183	The register is refreshed according to the settings in HWCONFIG when the PLC is powered-on. Users can edit the settings afterwards
SR185	Whenever a remote module is activated, after a cycle scan, the system refreshes the cycle time.
SR190~SR197	The register is refreshed by users.
SR198~SR199	The register is refreshed when the PLC is powered-on.
SR201~SR213	The register is refreshed according to the settings in HWCONFIG when the PLC is powered-on. Users can edit the settings afterwards
SR215~SR216	The register is refreshed by the system.
SR217~SR218	The register is refreshed according to the settings in HWCONFIG when powered-on. After that users can edit.

Special data register	Refresh time
SR220~SR226	The register is refreshed every scan cycle.
SR227~SR308	The register is refreshed when the program is downloaded to the PLC.
SR309~SR390	The register is refreshed when the status of the PLC changes.
SR391~SR397	The register is refreshed every scan cycle.
SR407	The register is refreshed every second.
SR408~SR416	The register is refreshed whenever the instruction END is executed.
SR421~SR424	The register is refreshed by users.
SR440~SR443	The register is refreshed when the PLC is powered-on.
SR444~SR451	The register is refreshed when the PLC is powered-on.
SR453	The register is refreshed when there is an error.
SR460	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR462~SR466	The register is refreshed by users.
SR467	The register is refreshed by the system.
SR468~SR469	When the PLC is supplied with power, the register is refreshed according to the position planning table. Users can edit the settings afterwards
SR470	The register is refreshed by the system.
SR472	The register is refreshed by users.
SR474	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR476~SR477	The register is refreshed by users.
SR480	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR482~SR486	The register is refreshed by users.
SR487	The register is refreshed by the system.
SR488~SR489	When the PLC is supplied with power, the register is refreshed according to the position planning table. Users can edit the settings afterwards
SR490	The register is refreshed by the system.
SR492	The register is refreshed by users.
SR494	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR496~SR497	The register is refreshed by users.
SR500	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR502~SR506	The register is refreshed by users.
SR507	The register is refreshed by the system.
SR508~SR509	When the PLC is supplied with power, the register is refreshed according to the position planning table. Users can edit the settings afterwards
SR510	The register is refreshed by the system.
SR512	The register is refreshed by users.
SR514	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR516~SR517	The register is refreshed by users.
SR520	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR522~SR526	The register is refreshed by users.
SR527	The register is refreshed by the system.

Special data register	Refresh time
SR528~SR529	When the PLC is supplied with power, the register is refreshed according to the position planning table. Users can edit the settings afterwards
SR530	The register is refreshed by the system.
SR532	The register is refreshed by users.
SR534	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the setting.
SR536~SR537	The register is refreshed by users.
SR540	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR542~SR546	The register is refreshed by users.
SR547	The register is refreshed by the system.
SR548~SR549	When the PLC is supplied with power, the register is refreshed according to the position planning table. Users can edit the settings afterwards
SR550	The register is refreshed by the system.
SR552	The register is refreshed by users.
SR554	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR556~SR557	The register is refreshed by users.
SR560	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR562~SR566	The register is refreshed by users.
SR567	The register is refreshed by the system.
SR568~SR569	When the PLC is supplied with power, the register is refreshed according to the position planning table. Users can edit the settings afterwards
SR570	The register is refreshed by the system.
SR572	The register is refreshed by users.
SR574	The register is refreshed by the system whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, users can edit the settings.
SR576~SR577	The register is refreshed by users.
SR580~SR603	The register is refreshed according to the settings in HWCONFIG when the PLC is powered-on. Users can edit the settings afterwards
SR604~SR609	The register is refreshed by users.
SR610~SR621	The register is refreshed whenever the output instruction is executed.
SR623~SR634	The register is refreshed whenever the instruction EIX or DIX is executed. ON: the interrupt is enabled OFF: the interrupt is disabled
SR658~SR901	The register is refreshed by the system.
SR820	The register is refreshed according to the settings in CANopen Builder.
SR821	The register is refreshed when the firmware is updated.
SR822	The register is refreshed according to the settings in HWCONFIG.
SR825~SR893	The register is refreshed by the system.
SR902	The register is refreshed by users.
SR1000~SR1006	The register is refreshed by users.
SR1007	The register is refreshed by the system.
SR1009	The register is refreshed by the system.
SR1010	The register is refreshed by users.
SR1011~SR1014	The register is refreshed by the system.

Special data register	Refresh time
SR1020~SR1107	1. The register is refreshed when a connection is established. 2. The register is refreshed every scan cycle.
SR1116~SR1117	The register is refreshed when the program is downloaded to the PLC.
SR1120~SR1183	The register is refreshed when the communication is complete.
SR1318~SR1320	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1335~SR1336	After the function of data exchange is enabled, the register is refreshed every scan cycle.
SR1340~SR1371	The register is refreshed when there is an error.
SR1375~SR1376	After the function of data exchange is enabled, the register is refreshed every scan cycle.
SR1380~SR1411	The register is refreshed when there is an error.

2.2.16 Additional Remarks on Special Auxiliary Relays and Special Data Registers

1. The scan timeout timer

- SM8/SR8

When a scan timeout occurs during the execution of the program, the error LED indicator on the PLC is ON all the time, and SM8 is ON.

The content of SR8 is the step address at which the watchdog timer is ON.

2. Clearing the warning light

- SM22

If SM22 is ON, the error log and the warning light will be cleared.

3. The real-time clock

- SM220, SR220~SR226, and SR391~SR397

SM220: Calibrating the real-time clock within ±30 seconds

When SM220 is switched from OFF to ON, the real-time clock is calibrated.

If the value of the second in the real-time clock is within the range between 0 and 29, the value of the minute is fixed, and the value of the second is cleared to zero.

If the value of the second in the real-time clock is within the range between 30 and 59, the value of the minute increases by one, and the value of the second is cleared to zero.

The corresponding functions and values of SR220~SR226 and SR391~SR397 are as follows.

Device		Function	Value
Binary-coded decimal system	Decimal system		
SR220	SR391	Year	00~99 (A.D.)
SR221	SR392	Month	1~12
SR222	SR393	Day	1~31
SR223	SR394	Hour	0~23
SR224	SR395	Minute	0~59
SR225	SR396	Second	0~59

Device		Function	Value
Binary-coded decimal system	Decimal system		
SR226	SR397	Week	1~7

SR391~SR397 correspond to SR220~ SR226. The difference between SR220~ SR226 and SR391~SR397 lies in the fact that the former adopts the binary-coded decimal while the latter adopts the decimal system. For example, December is represented as 12 in SR392 while it is represented as 12 in the binary-coded decimal. Please refer to chapter 6 for more information related to the instruction of real-time clock.

4. The functions related to communication

- SM96~SM107, SM209~SM212, SR201~SR202, and SR209~SR216

SR215 and SR216 are used to record the interface code of the communication port on the PLC. The functions represented by the interface codes are as follows.

Code	0	1	2	3	4	5	6
Function	No card	RS232	RS422	RS485	F2AD	F2DA	FCOPM

FCOPM function card only support Function Card 2.

When the interface of the communication port on the PLC is RS485, RS232, or RS422, SR209 records the communication format of COM1 on the PLC, and SR212 records the communication format of COM2 on the PLC. The setting values of the communication protocols are shown in the following table. Please refer to chapter 6 for more information related to the communication instructions.

b0	Data length		7 (value=0)		8 (value=1)
b1 **b2**	Parity bits		00	:	None
			01	:	Odd parity bits
			10	:	Even parity bits
b3	Stop bits		1 bit (value=0)		2 bits (value=1)
b4 **b5** **b6** **b7**	0001	(H 1)	:	4800	
	0010	(H 2)	:	9600	
	0011	(H 3)	:	19200	
	0100	(H 4)	:	38400	
	0101	(H 5)	:	57600	
	0110	(H 6)	:	115200	
	0111	(H 7)	:	230400	
	1000	(H 8)	:	500000	
	1001	(H 9)	:	921000	
	1010	(16#A)	:	Undefined	
	1011	(16#B)	:	Undefined	
	1100	(16#C)	:	Undefined	
	1101	(16#D)	:	Undefined	
	1110	(16#E)	:	Undefined	

	1111	(16#F)	:	User-defined*1	
b8~b15	Undefined (reserved)				

*1: Please refer to the HWCONFIG settings in ISPSoft for the user-defined baudrate.
*2: Please refer to section 6.19.3 for the use of communication flags and registers.

5. Clearing the contents of the device

- SM204/SM205

Device number	Device which is cleared
SM204 All non-latched areas are cleared.	The non-latched areas in the input relays, the output relays, the stepping relays, and the auxiliary relays are cleared. The non-latched areas in the timers, the counters, and the 32-bit counters are cleared. The non-latched areas in the data registers and the index registers are cleared. The watchdog timer does not act during this period of time.
SM205 All latched areas are cleared.	The latched areas in the timers, counters, and 32-bit counters are cleared. The latched auxiliary relays are cleared. The latched data registers are cleared. The watchdog timer does not act during this period of time.

Please refer to section 2.1.4 for more information related to the latched areas in the device range.

6. The error log in the PLC

- SR40~SR161

SR40: The maximum number of error logs which are stored in SR40 is 20. Every error log occupies 6 registers.

SR41: The error log pointer points to the latest error log. When an error occurs, the value of the error log pointer increases by one. The range of pointer values is 0~19. For example, the error log pointer points to the fourth error log when the value in SR41 is 3.

The time when the errors occur and the positions where the errors occur are recorded in SR42~SR161. The corresponding functions of these data registers are as follows.

Number	Slot	Module ID	Error code	Time when the error occurs					
				Year	Month	Day	Hour	Minute	Second
1	SR42 Low byte	SR43	SR44	SR45 High byte	SR45 Low byte	SR46 High byte	SR46 Low byte	SR47 High byte	SR47 Low byte
2	SR48 Low byte	SR49	SR50	SR51 High byte	SR51 Low byte	SR52 High byte	SR52 Low byte	SR53 High byte	SR53 Low byte
3	SR54 Low byte	SR55	SR56	SR57 High byte	SR57 Low byte	SR58 High byte	SR58 Low byte	SR59 High byte	SR59 Low byte
4	SR60 Low byte	SR61	SR62	SR63 High byte	SR63 Low byte	SR64 High byte	SR64 Low byte	SR65 High byte	SR65 Low byte
5	SR66 Low byte	SR67	SR68	SR69 High byte	SR69 Low byte	SR70 High byte	SR70 Low byte	SR71 High byte	SR71 Low byte
6	SR72 Low byte	SR73	SR74	SR75 High byte	SR75 Low byte	SR76 High byte	SR76 Low byte	SR77 High byte	SR77 Low byte
7	SR78 Low byte	SR79	SR80	SR81 High byte	SR81 Low byte	SR82 High byte	SR82 Low byte	SR83 High byte	SR83 Low byte
8	SR84 Low byte	SR85	SR86	SR87 High byte	SR87 Low byte	SR88 High byte	SR88 Low byte	SR89 High byte	SR89 Low byte

Number	Slot	Module ID	Error code	Time when the error occurs					
				Year	Month	Day	Hour	Minute	Second
9	SR90 Low byte	SR91	SR92	SR93 High byte	SR93 Low byte	SR94 High byte	SR94 Low byte	SR95 High byte	SR95 Low byte
10	SR96 Low byte	SR97	SR98	SR99 High byte	SR99 Low byte	SR100 High byte	SR100 Low byte	SR101 High byte	SR101 Low byte
11	SR102 Low byte	SR103	SR104	SR105 High byte	SR105 Low byte	SR106 High byte	SR106 Low byte	SR107 High byte	SR107 Low byte
12	SR108 Low byte	SR109	SR110	SR111 High byte	SR111 Low byte	SR112 High byte	SR112 Low byte	SR113 High byte	SR113 Low byte
13	SR114 Low byte	SR115	SR116	SR117 High byte	SR117 Low byte	SR118 High byte	SR118 Low byte	SR119 High byte	SR119 Low byte
14	SR120 Low byte	SR121	SR122	SR123 High byte	SR123 Low byte	SR124 High byte	SR124 Low byte	SR125 High byte	SR125 Low byte
15	SR126 Low byte	SR127	SR128	SR129 High byte	SR129 Low byte	SR130 High byte	SR130 Low byte	SR131 High byte	SR131 Low byte
16	SR132 Low byte	SR133	SR134	SR135 High byte	SR135 Low byte	SR136 High byte	SR136 Low byte	SR137 High byte	SR137 Low byte
17	SR138 Low byte	SR139	SR140	SR141 High byte	SR141 Low byte	SR142 High byte	SR142 Low byte	SR143 High byte	SR143 Low byte
18	SR144 Low byte	SR145	SR146	SR147 High byte	SR147 Low byte	SR148 High byte	SR148 Low byte	SR149 High byte	SR149 Low byte
19	SR150 Low byte	SR151	SR152	SR153 High byte	SR153 Low byte	SR154 High byte	SR154 Low byte	SR155 High byte	SR155 Low byte
20	SR156 Low byte	SR157	SR158	SR159 High byte	SR159 Low byte	SR160 High byte	SR160 Low byte	SR161 High byte	SR161 Low byte

7. The download log in the PLC

- SR227~SR308

SR227: The maximum number of download logs which are stored in SR227 is 20. Every download log occupies 4 registers. The download actions which are recorded are numbered, as shown in the following table.

Download action	Number
Downloading the program	1
Downloading the setting of the PLC	2
Downloading the module table	3

SR228: The download log pointer points to the latest download log. When a download action is executed, the value of the download log pointer increases by one. The range of pointer values is 0~19. For example, the download log pointer points to the fourth download log when the value in SR228 is 3.

The time when the downloading actions occur and the action numbers are recorded in SR229~SR30. The corresponding functions of these data registers are as follows.

Number	Action number	*Time when the download action occurs					
		Year	Month	Day	Hour	Minute	Second
1	SR229	SR230 High byte	SR230 Low byte	SR231 High byte	SR231 Low byte	SR232 High byte	SR232 Low byte

Number	Action number	*Time when the download action occurs					
		Year	Month	Day	Hour	Minute	Second
2	SR233	SR234 High byte	SR234 Low byte	SR235 High byte	SR235 Low byte	SR236 High byte	SR236 Low byte
3	SR237	SR238 High byte	SR238 Low byte	SR239 High byte	SR239 Low byte	SR240 High byte	SR240 Low byte
4	SR241	SR242 High byte	SR242 Low byte	SR243 High byte	SR243 Low byte	SR244 High byte	SR244 Low byte
5	SR245	SR246 High byte	SR246 Low byte	SR247 High byte	SR247 Low byte	SR248 High byte	SR248 Low byte
6	SR249	SR250 High byte	SR250 Low byte	SR251 High byte	SR251 Low byte	SR252 High byte	SR252 Low byte
7	SR253	SR254 High byte	SR254 Low byte	SR255 High byte	SR255 Low byte	SR256 High byte	SR256 Low byte
8	SR257	SR258 High byte	SR258 Low byte	SR259 High byte	SR259 Low byte	SR260 High byte	SR260 Low byte
9	SR261	SR262 High byte	SR262 Low byte	SR263 High byte	SR263 Low byte	SR264 High byte	SR264 Low byte
10	SR265	SR266 High byte	SR266 Low byte	SR267 High byte	SR267 Low byte	SR268 High byte	SR268 Low byte
11	SR269	SR270 High byte	SR270 Low byte	SR271 High byte	SR271 Low byte	SR272 High byte	SR272 Low byte
12	SR273	SR274 High byte	SR274 Low byte	SR275 High byte	SR275 Low byte	SR276 High byte	SR276 Low byte
13	SR277	SR278 High byte	SR278 Low byte	SR279 High byte	SR279 Low byte	SR280 High byte	SR280 Low byte
14	SR281	SR282 High byte	SR282 Low byte	SR283 High byte	SR283 Low byte	SR284 High byte	SR284 Low byte
15	SR285	SR286 High byte	SR286 Low byte	SR287 High byte	SR287 Low byte	SR288 High byte	SR288 Low byte
16	SR289	SR290 High byte	SR290 Low byte	SR291 High byte	SR291 Low byte	SR292 High byte	SR292 Low byte
17	SR293	SR294 High byte	SR294 Low byte	SR295 High byte	SR295 Low byte	SR296 High byte	SR296 Low byte
18	SR297	SR298 High byte	SR298 Low byte	SR299 High byte	SR299 Low byte	SR300 High byte	SR300 Low byte
19	SR301	SR302 High byte	SR302 Low byte	SR303 High byte	SR303 Low byte	SR304 High byte	SR304 Low byte
20	SR305	SR306 High byte	SR306 Low byte	SR307 High byte	SR307 Low byte	SR308 High byte	SR308 Low byte

*Time when the download action occurs: The data is stored as the values in the binary-coded decimal. The range of values is as follows.

Function	Value
Year	00~99 (A.D.)
Month	01~12
Day	01~31
Hour	00~23

Function	Value
Minute	00~59
Second	00~59

8. The PLC status change log

- SR309~SR390

SR309: The maximum number of PLC status change logs which are stored in SR309 is 20. Every PLC status change log occupies 4 registers. The PLC status change actions which are recorded are numbered, as shown in the following table.

PLC status change	Number
The PLC is supplied with power.	1
The PLC is disconnected.	2
The PLC starts to run.	3
The PLC stops running.	4
Default setting of the PLC	5

SR310: The PLC status change log pointer points to the latest PLC status change log. When the PLC status is changed once, the value of the PLC status change log pointer increases by one. The range of pointer values is 0~19. For example, the PLC status change log pointer points to the fourth PLC status change log when the value in SR310 is 3.

The time when the PLC status change actions occur is recorded in SR311~SR390. The corresponding functions of these data registers are as follows.

Number	Action number	*Time when the PLC status change action occurs					
		Year	Month	Day	Hour	Minute	Second
1	SR311	SR312 High byte	SR312 Low byte	SR313 High byte	SR313 Low byte	SR314 High byte	SR314 Low byte
2	SR315	SR316 High byte	SR316 Low byte	SR317 High byte	SR317 Low byte	SR318 High byte	SR318 Low byte
3	SR319	SR320 High byte	SR320 Low byte	SR321 High byte	SR321 Low byte	SR322 High byte	SR322 Low byte
4	SR323	SR324 High byte	SR324 Low byte	SR325 High byte	SR325 Low byte	SR326 High byte	SR326 Low byte
5	SR327	SR328 High byte	SR328 Low byte	SR329 High byte	SR329 Low byte	SR330 High byte	SR330 Low byte
6	SR331	SR332 High byte	SR332 Low byte	SR333 High byte	SR333 Low byte	SR334 High byte	SR334 Low byte
7	SR335	SR336 High byte	SR336 Low byte	SR337 High byte	SR337 Low byte	SR338 High byte	SR338 Low byte
8	SR339	SR340 High byte	SR340 Low byte	SR341 High byte	SR341 Low byte	SR342 High byte	SR342 Low byte
9	SR343	SR344 High byte	SR344 Low byte	SR345 High byte	SR345 Low byte	SR346 High byte	SR346 Low byte
10	SR347	SR348 High byte	SR348 Low byte	SR349 High byte	SR349 Low byte	SR350 High byte	SR350 Low byte

Number	Action number	*Time when the PLC status change action occurs					
		Year	Month	Day	Hour	Minute	Second
11	SR351	SR352 High byte	SR352 Low byte	SR353 High byte	SR353 Low byte	SR354 High byte	SR354 Low byte
12	SR355	SR356 High byte	SR356 Low byte	SR357 High byte	SR357 Low byte	SR358 High byte	SR358 Low byte
13	SR359	SR360 High byte	SR360 Low byte	SR361 High byte	SR361 Low byte	SR362 High byte	SR362 Low byte
14	SR363	SR364 High byte	SR364 Low byte	SR365 High byte	SR365 Low byte	SR366 High byte	SR366 Low byte
15	SR367	SR368 High byte	SR368 Low byte	SR369 High byte	SR369 Low byte	SR370 High byte	SR370 Low byte
16	SR371	SR372 High byte	SR372 Low byte	SR373 High byte	SR373 Low byte	SR374 High byte	SR374 Low byte
17	SR375	SR376 High byte	SR376 Low byte	SR377 High byte	SR377 Low byte	SR378 High byte	SR378 Low byte
18	SR379	SR380 High byte	SR380 Low byte	SR381 High byte	SR381 Low byte	SR382 High byte	SR382 Low byte
19	SR383	SR384 High byte	SR384 Low byte	SR385 High byte	SR385 Low byte	SR386 High byte	SR386 Low byte
20	SR387	SR388 High byte	SR388 Low byte	SR389 High byte	SR389 Low byte	SR390 High byte	SR390 Low byte

*Time when the PLC status change action occurs: The data is stored as the values in the binary-coded decimal. The range of values is as follows.

Function	Value
Year	00~99 (A.D.)
Month	01~12
Day	01~31
Hour	00~23
Minute	00~59
Second	00~59

9. The PLC operation flag

- SM400~SM403

SM400: The normally-open contact

SM401: The normally-closed contact

SM402: SM402 is ON during the first scan time, and then is switched OFF. The pulse width equals one scan time.

Users can use this contact to do the initial setting.

SM403: SM403 is OFF during the first scan time, and then is switched ON. That is, the negative pulse is generated the moment the PLC runs.

10. The initial clock pulse

- SM404 · SM405 · SM406 · SM407

The PLC provides seven types of clock pulses. When the PLC is supplied with power, the seven types of clock pulses act automatically.

Device	Function
SM404	10 millisecond clock pulse during which the pulse is ON for 5 milliseconds and is OFF for 5 milliseconds
SM405	100 millisecond clock pulse during which the pulse is ON for 50 milliseconds and is OFF for 50 milliseconds
SM406	200 millisecond clock pulse during which the pulse is ON for 100 milliseconds and is OFF for 100 milliseconds
SM407	One second clock pulse during which the pulse is ON for 500 milliseconds and is OFF for 500 milliseconds

The clock pulses are illustrated as follows.

11. The flags related to the memory card

- SM36、SM450~SM453、SM456、SR36、SR453、SR902

The memory card is used to backup the data in the PLC. Please refer to chapter 6 for instruction concerning the memory card.

Device	Function
SM36	Enable saving data to the memory card. When it is ON, the PLC will run according to the value in the SR36.
SM450	Whether the memory card exists ON: The memory card exists. OFF: The memory card does not exist.
SM452	The data in the memory card is being accessed. ON: The data in the memory card is being accessed. OFF: The data in the memory card is not accessed.
SM453	An error occurs during the operation of the memory card. ON: An error occurs.
SM456	Execution of data logger and the memory card. (ON: execution by the values in SR902)
SR36	The system saves data to the memory card. This function need to work with SM36.
SR453	If an error occurs during the operation of the memory card, the error code will be recorded.
SR902	The code for the executions of data logger and the memory card (need to work with SM456), e.g. H5AA5: write the sampling data from the data logger into the memory card.

- SR36 only store 2 logs：

A. The number 1234 means the error log (SR40~SR161) of the PLC is stored in the memory card.
B. The number 3456 means the error log (SR40~SR161) of the PLC and the PLC state changing log (SR309~SR390) are stored in the memory card.

12. The high-speed output instruction is being executed. The output immediately stops when the instruction is disabled or stops.

- SM476, SM477, SM496, SM497, SM516, SM517, SM536, SM537, SM556, SM557, SM576, SM577:

OFF (default): deceleration stops

ON: immediate stops

- SM463, SM474, SM483, SM494, SM503, SM514, SM523, SM534, SM543, SM554, SM563, SM574：the flags are used to pause outputting.
  When the flag is OFF to ON, it means the output will be stopped. This should work with the abovementioned flags; refer to the section above for more actions on stop).

When the flag is ON to OFF, it means the rest of the outputs will be executed.

13. Position control output limit in ISPSoft

- SR580~SR603

Positive output limit: set up the limit in ISPSoft; when the output position is greater than the positive limit, the output stops immediately.

Negative output limit: set up the limit in ISPSoft; when the output position is smaller than the negative limit, the output stops immediately.

When the limits of positive and negative output are both 0, it means the function is disabled. This function should work with the output instruction. The system will only check the limit set in the ISPSoft, when the instruction is executed. Thus it will not come to an immediate stop even when it is out of the output limit. If an immediate stop is needed, it is suggested to use the external input as the way to check the limit.

14. S curve mode

- SR604~SR609, SM468, SM488, SM508, SM528, SM548, SM568

There are 3 S curves, small, medium and large. The range is between 0 to 2. When the value exceeds the range, the system will treat the value as the minimum 0 or the maximum 2.

The S curve mode should work with the flags such as SM468, SM488, SM568 and so on. If the flag is ON, the parameters of the S curve will be executed by the output instruction.

15. Ethernet IP related flags

- SM1000、SR1000~SR1006

SM/SR	Function	Action
SM1000	Ethernet setting flag.	ON: the values in SR1000~SR1006 is written into the flash memory. After it is written, the PLC will reset it to OFF. NOTE 1: do not set the flag to ON continuously to avoid damage on the flash memory. NOTE 2: It is required the PLC to be in the stop state to write the values into the flash memory.
SR1000	Ethernet IP address (32-bit)	For example: 192.168.1.5, SR1000 will be 16#C0A8 and SR1001 will be 0105.
SR1001		
SR1002	Ethernet netmask address (32-bit)	For example: 255.255.255.0, SR1002 will be 16#FFFF, and SR1003 will be FF00
SR1003		
SR1004	Ethernet gateway address (32-bit)	For example: 192.168.1.1, SR1004 will be 16#C0A8, and SR1005 will be 0101
SR1005		
SR1006	Time for which the TCP connection has been persistent	Unit: second

- SM1090、SM1091、SM1106~SM1109

SM	Function	Action
SM1090	The TCP connection is busy.	ON: TCP connection timeout
SM1091	The UDP connection is busy.	ON: UDP connection timeout
SM1106	Ethernet connection error	OFF: when the PHY initialization succeeds. ON: when the PHY initialization fails.
SM1107	Ethernet basic setting error	OFF: correct basic setting ON: basic setting error
SM1109	TCP/UDP socket–the local port is already used.	ON: the same port has been used

For the error codes, the corresponding LED indicators, and other troubleshooting, please refer to chapter 12 of the AS Series operation manual.

16. Email settings

- SM1113，SM1116~SM1155

If the sending of email fails, the flag of the email service error SM1113 will be ON.

The triggers of email sending and the corresponding flags (SM1116~SM1155) are listed below:

Function \ Item	Trigger 1	Trigger 2	Trigger 3	Trigger 4
Email service	SM1116	SM1126	SM1136	SM1146
	ON: enabled , OFF: disabled			
Email sending	SM1117	SM1127	SM1137	SM1147
	ON: email is sending now, OFF: email has been sent			
Emails sent successfully	SM1119	SM1129	SM1139	SM1149
	ON: email has been sent successfully			
Error 1 of email sending	SM1120	SM1130	SM1140	SM1150
	The email cannot be sent due to email content error			
SMTP response timeout	SM1122	SM1132	SM1142	SM1152
	After sending the email out, the SMTP server responds timeout.			
SMTP server response error	SM1123	SM1133	SM1143	SM1153
	After sending the email out, the SMTP server responds error.			
Error 2 of email sending	SM1124	SM1134	SM1144	SM1154
	After sending the email out, the size of the attachment exceeds the limit.			
Error 3 of email sending	SM1125	SM1135	SM1145	SM1155
	After sending the email out, the attachment is not found, SM1125 is ON.			

17. Flags and registers concerning data exchange

- Data exchange flags for COM1 connections

SM	Type	Function
SM750	R/W	Data exchange via COM1 has been enabled by ISPSoft.
SM752 ~ SM783	R/W	Connection 1~32 via COM1 for data exchange has been started.
SM784 ~ SM815	R	The data has been received via COM 1 connection 1~32 for data exchange.
SM816 ~ SM847	R	An error occurs in the COM1 connection 1~32 for data exchange

- Data exchange flags for COM2 connections

SM	Type	Function
SM862	R/W	Data exchange via COM2 has been enabled by ISPSoft.
SM864 ~ SM895	R/W	Connection 1~32 via COM2 for data exchange has been started.
SM896 ~ SM927	R	The data has been received via COM2 connection 1~32 for data exchange.
SM928 ~ SM959	R	An error occurs in the COM2 connection 1~32 for data exchange

- Data registers for COM1 connections

SR	Function
SR1335	The actual cycle time of connection 1~32 for data exchange via COM1
SR1336	The number of connection that is currently performing a cyclical data exchange via COM1
SR1340 ~ SR1371	The error code for data exchange via the COM1 connection 1~32

- Data registers for COM2 connections

SR1375	The actual cycle time of connection 1~32 for data exchange via COM2
SR1376	The number of connection that is currently performing a cyclical data exchange via COM2
SR1380 ~ SR1411	The error code for data exchange via the COM2 connection 1~32

The error codes 1~7 are the standard response error codes of MODBUS protocol. The error code 9 means timeout.

- Data exchange flags for Ethernet connections

SM	Type	Function
SM1167	R/W	Data exchange via Ethernet port has been enabled by ISPSoft.
SM1168 ~ SM1199	R/W	Connection 1~32 via Ethernet port for data exchange has been started.
SM1200 ~ SM1231	R	The data has been received via Ethernet port connection 1~32 for data exchange.
SM1232 ~ SM1263	R	An error occurs in the Ethernet port connection 1~32 for data exchange

- Data registers for Ethernet connections

SR	Function
SR1120 ~ SR1151	The actual connection time for data exchange via the Ethernet connection 1~32
SR1152 ~ SR1183	The error code for data exchange via the Ethernet connection 1~32

- Error codes for Ethernet connections

Error code	Description
16#00XX	Remote module response error
16#F000	Ethernet connection is not established
16#F001	Remote module response timeout
16#F003	TCP connection timeout
16#F007	Response error
16#F009	Connection lost in the remote module

2.2.17 Index Register (E)

The index register is the 16-bit data register. It is like the general register in that the data can be read from it and written into it. However, it is mainly used as the index register. The range of index registers is E0~E9. Please refer to section 4.4 in AS Series Programming Manual for more information about the usage of index registers.

2.2.18 File Registers (FR)

- AS series PLC provides users with File Registers for storing larger amount of parameters.
- Users can edit, upload, download the parameters in the file registers via ISPSoft.
- The values in FR can be read while operating the PLC. Please refer to API2303 MEMW in AS Series programming manual for more information about how to write in FR.

MEMO

3

Chapter 3 Instruction Tables

Table of Contents

3.1 Instructions 3-2
3.1.1 Basic Instructions 3-2
3.1.2 Applied Instructions 3-2

3.2 Instruction Tables 3-3
3.2.1 Basic Instructions 3-3
3.2.2 Applied Instructions (Sorted numerically) 3-4
3.2.3 Applied Instructions (Sorted Alphabetically) 3-5
3.2.4 Device Tables 3-6

3.3 Lists of Basic Instructions 3-7

3.4 Lists of Applied Instructions 3-9
3.4.1 Applied Instructions (Sorted numerically) 3-9
3.4.2 Applied Instructions (Sorted Alphabetically) 3-37

3.1 Instructions

Instructions used in the AS300 series PLC include basic instructions and applied instructions.

3.1.1 Basic Instructions

Classification	Description
Contact instructions	Loading the contact, connecting the contact in series, connecting the contact in parallel, and etc.
Output instructions	Bit device output; pulse output
Master control Instructions	Setting and resetting the master control
Rising-edge/Falling-edge detection contact instructions	Triggering the instructions that load the contact, connect the contacts in series, and connect the contacts in parallel
Rising-edge/Falling-edge differential output instructions	Bit device differential output
Other instructions	Other instructions

3.1.2 Applied Instructions

API	Classification	Description
0000~0083	Comparison instructions	Comparisons such as =, <>, >, >=, <, <=, and etc.
0100~0118	Arithmetic instructions	Using binary numbers or binary-coded decimal numbers to add, subtract, multiply, or divide.
0200~0217	Data conversion instructions	Converting the binary-coded decimal number into the binary number, and converting the binary number into the binary-coded decimal number
0300~0310	Data transfer instructions	Transfer the specified data
0400~0402	Jump instructions	The program jumps.
0500~0504	Program execution instructions	Enabling or disabling the interrupt
0600	I/O refreshing instructions	Refreshing the I/O.
0700~0710	Convenience instructions	Instructions which are applied to the counters, the teaching timers, the special timers, and etc.
0800~0817	Logic instructions	Logical operations such as logical addition, logical multiplication, and etc.
0900~0904	Rotation instructions	Rotating/Shifting the specified data
1000~1011	Basic instructions	Timer instructions and counter instructions
1100~1115	Shift instructions	Shifting the specified data
1200~1225	Data processing instructions	16-bit data processing such as decoding and encoding.
1300~1302	Structure creation instructions	Nested loops
1400~1401	Module instructions	Reading the data from the special module and writing the data into the special module
1500~1517	Floating-point number instructions	Floating-point number operations
1600~1608	Real-time clock instructions	Reading/Writing, adding/subtracting and comparing the time
1700~1704	Peripheral instructions	I/O points connected to the peripheral
1806~1817	Communication instructions	Controlling the peripheral though communication

API	Classification	Description
1900~1905	Other instructions	Instructions which are different from those mentioned above
2100~2119	String processing instructions	Conversion between binary/binary-coded decimal numbers and ASCII codes; conversion between binary numbers and strings; conversion between floating-point numbers and strings; string processing
2200~2210	Ethernet instructions	Controlling the Ethernet data exchange
2300~2303	Memory card instructions	Reading the data from the memory card and writing the data into the memory card
2400~2401	Task control instructions	Controlling the task in the program
2500~2502	Sequential function charts (SFC) instructions	Controlling the SFC instructions
2700~2723	High-speed output instructions	High-speed output and position control instructions
2800~2807	Delta special CANopen communication instructions	CANopen communication instructions especially for Delta devices

3.2 Instruction Tables

3.2.1 Basic Instructions

Instruction code	Symbol	Function	Operand
LD	???	Loading the contact A/Connecting the contact A in series/Connecting the contact A in parallel	DX X Y M S T C HC D L SM PR
AND	???		
OR	???		

The descriptions:

①: The instruction name

②: The symbol used in the ladder diagram in ISPSoft

③: The function

④: The operands supported by the instruction

3.2.2 Applied Instructions (Sorted numerically)

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0000	LD=	DLD=	–	= Q S1 S2 / D= Q S1 S2	Comparing the contact types ON: S1 = S2 OFF: S1≠S2
0001	LD<>	DLD<>	–	<> Q S1 S2 / D<> Q S1 S2	Comparing the contact types ON: S1≠S2 OFF: S1 = S2
0002	LD>	DLD>	–	> Q S1 S2 / D> Q S1 S2	Comparing the contact types ON: S1 > S2 OFF: S1≦S2

① ② ③ ④ ⑤ ⑥

API	Instruction code		Pulse Instruction	Symbol	Function
	32-bit	64-bit			
0018	FLD=	DFLD=	–	F= Q S1 S2 / DF= Q S1 S2	Comparing the floating-point number contact types ON: S1 = S2 OFF: S1≠S2

⑦

The descriptions:

①: The applied instruction number

②: The instruction name

③: If the 16-bit instruction can be used as the 32-bit instruction, a D is added in front of the 16-bit instruction to form the 32-bit instruction.

④: ✓ indicates that the instruction can be used as the pulse instruction, whereas ─ indicates that it cannot.

If users want to use the pulse instruction, they only need to add a P in back of the instruction.

⑤: The symbol used in the ladder diagram in ISPSoft

⑥: The function

3.2.3 Applied Instructions (Sorted Alphabetically)

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
E	0501	EI	–	–	Enabling the interrupt
	2208	EIPRW	–	–	Reading and writing via Ethernet/IP connection
	0503	EIX	–	–	Disabling the specific interrupt
	1203	ENCO	–	✓	Encoder
	1905	EPOP	–	✓	Reading the data into the index registers
	1904	EPUSH	–	✓	Storing the contents of the index registers
①	②	③	④	⑤	⑥

The descriptions:

①: The initial of the instruction name

②: The applied instruction number

③ ~ ④ : The instruction names

If the 16-bit instruction can be used as the 32-bit instruction, a D is added in front of the 16-bit instruction to form the 32-bit instruction.

⑤ : ✓ indicates that the instruction can be used as the pulse instruction, whereas – indicates that it cannot.

If users want to use the pulse instruction, they only need to add a P in back of the instruction.

⑥: The function

3.2.4 Device Tables

①　②　③　④

API	Instruction code			Operand	Function
0100	D	+	P	S_1、S_2、D	Addition of binary numbers

⑤

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

⑥

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

⑦

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

⑧

S_1 : Augend

S_2 : Addend

D : Sum

The descriptions:

①: The applied instruction number

②: The instruction name

If the 16-bit instruction can be used as the 32-bit instruction, a D is added in front of the 16-bit instruction to form the 32-bit instruction.

③: The operand

④: The function

⑤: The devices which are supported by the operand

1. The decimal forms are notated by K, but they are entered directly in ISPSoft. For example, the decimal number 30 is entered directly in ISPSoft.
2. The hexadecimal forms are notated by 16#. For example, the decimal number 30 is represented by 16#1E in the hexadecimal system.
3. The floating-point numbers are notated by F/DF, but they are represented by decimal points in ISPSoft. For example, the floating-point number F500 is represented by 500.0 in ISPSoft.
4. The strings are notated by "$", but they are represented by " " in ISPSoft. For example, the string 1234 is represented by "1234" in ISPSoft.
5. ○: The hollow circle

   The device cannot be modified by an index register.

6. ●: The solid circle

The device cannot be modified by an index register.

⑥ : The unit of the operand

⑦ : The format of the instruction

It indicates whether the instruction can be used as the pulse instruction, the 16-bit instruction, or the 32-bit instruction

⑧ : The ladder diagram

3.3 Lists of Basic Instructions

- Contact instructions

Instruction code	Symbol	Function	Operand
LD		Loading contact A/Connecting contact A in series/Connecting contact A in parallel	DX, X, Y, M, SM, S, T, C, HC, D
AND			
OR			
LDI		Loading contact B/Connecting contact B in series/Connecting contact B in parallel	DX, X, Y, M, SM, S, T, C, HC, D
ANI			
ORI			

- Output instructions

Instruction code	Symbol	Function	Execution condition	Operand
OUT		Driving the coil		DY, Y, M, SM, S, T, C, HC, D
SET		Keeping the device on		DY, Y, M, SM, S, T, C, HC, D

- Master control instructions

Instruction code	Symbol	Function	Operand
MC		Setting the master control	N

Instruction code	Symbol	Function	Operand
MCR	MCR N	Resetting the master control	N

- Rising-edge/Falling-edge detection contact instructions

Instruction code	Symbol	Function	Operand	Instruction code
LDP	???	Starting the rising-edge detection/Connecting the rising-edge detection in series/Connecting the rising-edge detection in parallel		DX, X, Y, M, SM, S, T, C, HC, D
ANDP	???			
ORP	???			
PED	???			X, Y,M, SM, S, T, C, HC, D
APED	???			
OPED	???			
LDF	???	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel		DX, X, Y, M, SM, S, T, C, HC, D
ANDF	???			
ORF	???			
NED	???			X, Y,M, SM, S, T, C, HC, D
ANED	???			
ONED	???			

- Rising-edge/Falling-edge differential output instructions

Instruction code	Symbol	Function	Execution condition	Operand
PLS	PLS En D	Rising-edge differential output		Y, M, SM, S
PLF	PLF En D	Falling-edge differential output		Y, M, SM, S

- Other instructions

Instruction code	Symbol	Function	Operand
INV	INV	Inverting the logical operation result	–
NP		The circuit is rising edge-triggered.	–
PN		The circuit is falling edge-triggered.	–
FB_NP	???	The circuit is rising edge-triggered.	Y, M, S, D
FB_PN	???	The circuit is falling edge-triggered.	Y, M, S, D

3.4 Lists of Applied Instructions

3.4.1 Applied Instructions (Sorted numerically)

- Comparison instructions

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0000	LD=	DLD=	–	= Q S1 S2 / D= Q S1 S2	Comparing the values ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0001	LD<>	DLD<>	–	<> Q S1 S2 / D<> Q S1 S2	Comparing the values ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0002	LD>	DLD>	–	> Q S1 S2 / D> Q S1 S2	Comparing the values ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$
0003	LD>=	DLD>=	–	>= Q S1 S2 / D>= Q S1 S2	Comparing the values ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$
0004	LD<	DLD<	–	< Q S1 S2 / D< Q S1 S2	Comparing the values ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0005	LD<=	DLD<=	—	<= Q S1 S2 / D<= Q S1 S2	Comparing the values ON: $S_1 \leqq S_2$ OFF: $S_1 > S_2$
0006	AND=	DAND=	—	= Q S1 S2 / D= Q S1 S2	Comparing the values ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0007	AND<>	DAND<>	—	<> Q S1 S2 / D<> Q S1 S2	Comparing the values ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0008	AND>	DAND>	—	> Q S1 S2 / D> Q S1 S2	Comparing the values ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$
0009	AND>=	DAND>=	—	>= Q S1 S2 / D>= Q S1 S2	Comparing the values ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$
0010	AND<	DAND<	—	< Q S1 S2 / D< Q S1 S2	Comparing the values ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$
0011	AND<=	DAND<=	—	<= Q S1 S2 / D<= Q S1 S2	Comparing the values ON: $S_1 \leqq S_2$ OFF: $S_1 > S_2$
0012	OR=	DOR=	—	= Q S1 S2 / D= Q S1 S2	Comparing the values ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0013	OR<>	DOR<>	—	<> Q S1 S2 / D<> Q S1 S2	Comparing the values ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0014	OR>	DOR>	—	> Q S1 S2 / D> Q S1 S2	Comparing the values ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$
0015	OR>=	DOR>=	—	>= Q S1 S2 / D>= Q S1 S2	Comparing the values ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$
0016	OR<	DOR<	—	< Q S1 S2 / D< Q S1 S2	Comparing the values ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$
0017	OR<=	DOR<=	—	<= Q S1 S2 / D<= Q S1 S2	Comparing the values ON: $S_1 \leqq S_2$ OFF: $S_1 > S_2$
0018	—	FLD=	—	F= Q S1 S2	Comparing the floating-point numbers ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0019	—	FLD<>	—	F<> Q S1 S2	Comparing the floating-point numbers ON: $S_1$1≠S_2 OFF: $S_1 = S_2$
0020	—	FLD>	—	F> Q S1 S2	Comparing the floating-point numbers ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$
0021	—	FLD>=	—	F>= Q S1 S2	Comparing the floating-point numbers ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$
0022	—	FLD<	—	F< Q S1 S2	Comparing the floating-point numbers ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$
0023	—	FLD<=	—	F<= Q S1 S2	Comparing the floating-point numbers ON: $S_1 \leqq S_2$ OFF: $S_1 > S_2$
0024	—	FAND=	—	F= Q S1 S2	Comparing the floating-point numbers ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0025	—	FAND<>	—	F<> Q S1 S2	Comparing the floating-point numbers ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0026	—	FAND>	—	F> Q S1 S2	Comparing the floating-point numbers ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$
0027	—	FAND>=	—	F>= Q S1 S2	Comparing the floating-point numbers ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$
0028	—	FAND<	—	F< Q S1 S2	Comparing the floating-point numbers ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$
0029	—	FAND<=	—	F<= Q S1 S2	Comparing the floating-point numbers ON: $S_1 \leqq S_2$ OFF: $S_1 > S_2$
0030	—	FOR=	—	F= Q S1 S2	Comparing the floating-point numbers ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0031	—	FOR<>	—	F<> Q S1 S2	Comparing the floating-point numbers ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0032	—	FOR>	—	F> Q S1 S2	Comparing the floating-point numbers ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0033	–	FOR>=	–	F>= Q S1 S2	Comparing the floating-point numbers ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$
0034	–	FOR<	–	F< Q S1 S2	Comparing the floating-point numbers ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$
0035	–	FOR<=	–	F<= Q S1 S2	Comparing the floating-point numbers ON: $S_1 \leqq S_2$ OFF: $S_1 > S_2$
0036	LD$=	–	–	$= Q S1 S2	Comparing the strings ON: $S_1 = S_2$ ON: $S_1 \neq S_2$
0037	LD$<>	–	–	$<> Q S1 S2	Comparing the strings ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0042	AND$=	–	–	$= Q S1 S2	Comparing the strings ON: $S_1 = S_2$ OFF $S_1 \neq S_2$
0043	AND$<>	–	–	$<> Q S1 S2	Comparing the strings ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0048	OR$=	–	–	$= Q S1 S2	Comparing the strings ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0049	OR$<>	–	–	$<> Q S1 S2	Comparing the strings ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0054	CMP	DCMP	✓	CMP En S1 S2 D CMPP En S1 S2 D DCMP En S1 S2 D DCMPP En S1 S2 D	Comparing the values
0055	ZCP	DZCP	✓	ZCP En S1 S2 S D ZCPP En S1 S2 S D DZCP En S1 S2 S D DZCPP En S1 S2 S D	Zone comparison
0056	–	FCMP	✓	FCMP En S1 S2 D FCMPP En S1 S2 D	Comparing the floating-point numbers

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0057	–	FZCP	✓	FZCP En S1 D S2 S FZCPP En S1 D S2 S	Floating-point zone comparison
0058	MCMP	–	✓	MCMP En S1 D S2 n MCMPP En S1 D S2 n	Matrix comparison
0059	CMPT=	–	✓	CMPT= En S1 D S2 n CMPT=P En S1 D S2 n	Comparing the tables ON: =
0060	CMPT<>	–	✓	CMPT<>& En S1 D S2 n CMPT<>&P En S1 D S2 n	Comparing the tables ON: ≠
0061	CMPT>	–	✓	CMPT>& En S1 D S2 n CMPT>&P En S1 D S2 n	Comparing the tables ON: >
0062	CMPT>=	–	✓	CMPT>=& En S1 D S2 n CMPT>=&P En S1 D S2 n	Comparing the tables ON: ≧
0063	CMPT<	–	✓	CMPT<& En S1 D S2 n CMPT<&P En S1 D S2 n	Comparing the tables ON: <
0064	CMPT<=	–	✓	CMPT<=& En S1 D S2 n CMPT<=&P En S1 D S2 n	Comparing the tables ON: ≦
0065	CHKADR	–	–	CHKADR En S D n	Checking the address of the contact type of pointer register
0066	LDZ=	DLDZ=	–	Z= Q S1 S2 S3 DZ= Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$
0067	LDZ<>	DLDZ<>	–	Z<>& Q S1 S2 S3 DZ<>& Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$
0068	LDZ>	DLDZ>	–	Z>& Q S1 S2 S3 DZ>& Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0069	LDZ>=	DLDZ>=	—	Z>= Q S1 S2 S3 / DZ>= Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$
0070	LDZ<	DLDZ<	—	Z< Q S1 S2 S3 / DZ< Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$
0071	LDZ<=	DLDZ<=	—	Z<= Q S1 S2 S3 / DZ<= Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$
0072	ANDZ=	DANDZ=	—	Z= Q S1 S2 S3 / DZ= Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$
0073	ANDZ<>	DANDZ<>	—	Z<> Q S1 S2 S3 / DZ<> Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$
0074	ANDZ>	DANDZ>	—	Z> Q S1 S2 S3 / DZ> Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$
0075	ANDZ>=	DANDZ>=	—	Z>= Q S1 S2 S3 / DZ>= Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$
0076	ANDZ<	DANDZ<	—	Z< Q S1 S2 S3 / DZ< Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$
0077	ANDZ<=	DANDZ<=	—	Z<= Q S1 S2 S3 / DZ<= Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$
0078	ORZ=	DORZ=	—	Z= Q S1 S2 S3 / DZ= Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$
0079	ORZ<>	DORZ<>	—	Z<> Q S1 S2 S3 / DZ<> Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0080	ORZ>	DORZ>	—	Z> Q S1 S2 S3 / DZ> Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$
0081	ORZ>=	DORZ>=	—	Z>= Q S1 S2 S3 / DZ>= Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$
0082	ORZ<	DORZ<	—	Z< Q S1 S2 S3 / DZ< Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$
0083	ORZ<=	DORZ<=	—	Z<= Q S1 S2 S3 / DZ<= Q S1 S2 S3	Comparing the absolute result of the contact type ON: $\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$ OFF: $\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$

- Arithmetic instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0100	+	D+	✓	+ En S1 S2 D / +P En S1 S2 D / D+ En S1 S2 D / D+P En S1 S2 D	Addition of binary numbers $S_1+S_2=D$
0101	-	D-	✓	- En S1 S2 D / -P En S1 S2 D / D- En S1 S2 D / D-P En S1 S2 D	Subtraction of binary numbers $S_1-S_2=D$
0102	*	D*	✓	* En S1 S2 D / *P En S1 S2 D / D* En S1 S2 D / D*P En S1 S2 D	Multiplication of binary numbers $S_1*S_2=D$
0103	/	D/	✓	/ En S1 S2 D / /P En S1 S2 D / D/ En S1 S2 D / D/P En S1 S2 D	Division of binary numbers $S_1/S_2=D$

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0104	–	F+	✓	F+ En S1 S2 D F+P En S1 S2 D	Addition of floating-point numbers $S_1+S_2=D$
0105	–	F-	✓	F- En S1 S2 D F-P En S1 S2 D	Subtraction of floating-point numbers $S_1-S_2=D$
0106	–	F*	✓	F* En S1 S2 D F*P En S1 S2 D	Multiplication of floating-point numbers $S_1*S_2=D$
0107	–	F/	✓	F/ En S1 S2 D F/P En S1 S2 D	Division of floating-point numbers $S_1/S_2=D$
0112	BK+	DBK+	✓	BK+ En S1 S2 n D BK+P En S1 S2 n D DBK+ En S1 S2 n D DBK+P En S1 S2 n D	Addition of binary numbers in blocks
0113	BK-	DBK-	✓	BK- En S1 S2 n D BK-P En S1 S2 n D DBK- En S1 S2 n D DBK-P En S1 S2 n D	Subtraction of binary numbers in blocks
0114	$+	–	✓	$+ En S1 S2 D $+P En S1 S2 D	Linking the strings
0115	INC	DINC	✓	INC En D INCP En D DINC En D DINCP En D	Adding one to the binary number
0116	DEC	DDEC	✓	DEC En D DECP En D DDEC En D DDECP En D	Subtracting one from the binary number
0117	MUL16	MUL32	✓	MUL16 En S1 S2 D MUL16P En S1 S2 D MUL32 En S1 S2 D MUL32P En S1 S2 D	Multiplication of binary numbers for 16-bit Multiplication of binary numbers for 32-bit
0118	DIV16	DIV32	✓	DIV16 En S1 S2 D DIV16P En S1 S2 D DIV32 En S1 S2 D DIV32P En S1 S2 D	Division of binary numbers for 16-bit Division of binary numbers for 32-bit

- Data conversion instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0200	BCD	DBCD	✓	BCD En S D BCDP En S D DBCD En S D DBCDP En S D	Converting the binary number into the binary-coded decimal number
0201	BIN	DBIN	✓	BIN En S D BINP En S D DBIN En S D DBINP En S D	Converting the binary-coded decimal number into the binary number
0202	FLT	DFLT	✓	FLT En S D FLTP En S D DFLT En S D DFLTP En S D	Converting the binary integer into the binary floating-point number
0204	INT	DINT	✓	INT En S D INTP En S D DINT En S D DINTP En S D	Converting the 32-bit floating-point number into the binary integer
0206	MMOV	–	✓	MMOV En S D MMOVP En S D	Converting the 16-bit value into the 32-bit value
0207	RMOV	–	✓	RMOV En S D RMOVP En S D	Converting the 32-bit value into the 16-bit value
0208	GRY	DGRY	✓	GRY En S D GRYP En S D DGRY En S D DGRYP En S D	Converting the binary number into the Gray code
0209	GBIN	DGBIN	✓	GBIN En S D GBINP En S D DGBIN En S D DGBINP En S D	Converting the Gray code into the binary number
0210	NEG	DNEG	✓	NEG En D NEGP En D DNEG En D DNEGP En D	Two's complement
0211	–	FNEG	✓	FNEG En D FNEGP En D	Reversing the sign of the 32-bit floating-point number
0212	–	FBCD	✓	FBCD En S D FBCD En S D	Converting the binary floating-point number into the decimal floating-point number
0213	–	FBIN	✓	FBIN En S D FBINP En S D	Converting the decimal floating-point number into the binary floating-point number
0214	BKBCD	–	✓	BKBCD En S D n BKBCDP En S D n	Converting the binary numbers in blocks into the binary-coded decimal numbers in blocks

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0215	BKBIN	—	✓	BKBIN En S D n BKBINP En S D n	Converting the binary numbers in blocks into the binary-coded decimal numbers in blocks
0216	SCAL	DSCAL	✓	SCAL En S1 D S2 S3 SCALP En S1 D S2 S3 DSCAL En S1 D S2 S3 DSCALP En S1 D S2 S3	Scale value operation
0217	SCLP	DSCLP	✓	SCLP En S1 D S2 SCLPP En S1 D S2 DSCLP En S1 D S2 DSCLPP En S1 D S2	Parameter type of scale value operation

- Data transfer instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0300	MOV	DMOV	✓	MOV En S D MOVP En S D DMOV En S D DMOVP En S D	Transferring the data S: Data source D: Data destination
0302	$MOV	—	✓	$MOV En S D $MOVP En S D	Transferring the string
0303	CML	DCML	✓	CML En S D CMLP En S D DCML En S D DCMLP En S D	Inverting the data
0304	BMOV	DBMOV	✓	BMOV En S D n BMOVP En S D n DBMOV En S D n DBMOVP En S D n	Transferring all data
0305	NMOV	DNMOV	✓	NMOV En S D n NMOVP En S D n DNMOVP En S D n DNMOV En S D n	Transferring the data to several devices

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0306	XCH	DXCH	✓	XCH En S1 S2 XCHP En S1 S2 DXCH En S1 S2 DXCHP En S1 S2	Exchanging the data
0307	BXCH	—	✓	BXCH En S1 S2 n BXCHP En S1 S2 n	Exchanging all data
0308	SWAP	DSWAP	✓	SWAP En S SWAPP En S DSWAP En S DSWAPP En S	Exchange the high byte with the low byte
0309	SMOV	—	✓	SMOV En S D m1 m2 n SMOVP En S D m1 m2 n	Transferring the digits
0310	MOVB	—	✓	MOVB En S D m1 m2 n MOVBP En S D m1 m2 n	Transferring several bits

- Jump instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0400	CJ	—	✓	CJ En S CJP En S	Conditional jump
0401	JMP	—	—	JMP En S	Unconditional jump
0402	GOEND	—	—	GOEND	Jumping to END

- Program execution instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0500	DI	—	—	DI	Disabling the interrupt
0501	EI	—	—	EI	Enabling the interrupt
0503	EIX	—	—	EIX En S	Disabling the specific interrupt
0504	DIX	—	—	DIX En S	Enabling the specific interrupt

- I/O refreshing instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0600	REF	–	✓	REF En D n REFP En D n	Refreshing the I/O

- Convenience instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0700	ALT	–	✓	ALT En D ALTP En D	Alternating between ON and OFF
0701	TTMR	–	–	TTMR n D	Teaching timer
0702	STMR	–	–	STMR S D m	Special timer
0703	RAMP	DRAMP	–	RAMP S1 D S2 N DRAMP En S1 D S2 n	Cyclic ramp signal
0704	MTR	–	–	MTR En S D2 D1 N	Matrix input
0705	ABSD	DABSD	–	ABSD S1 D S2 N DABSD En S1 D S2 N	Absolute drum sequencer
0706	INCD	–	–	INCD S1 D S2 N	Incremental drum sequencer
0708	–	DPIDE	–	DPIDE En PID_RUN MV SV PV PID_MODE PID_MAN MOUT_AUTO CYCLE Kc_Kp Ti_Ki Td_Kd Tf PID_BQ PID_DE PID_DIR ERR_DBW MV_MAX MV_MIN MOUT BIAS I_MV	PID algorithm

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0709	XCMP	–	–	XCMP En S1 D S2 S3 S4	Setups for comparing the inputs of multiple work stations
0710	YOUT	–	–	YOUT En S1 D S2 S3	Comparing the outputs of multiple work stations

- Logic instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0800	WAND	DAND	✓	WAND En S1 D S2 WANDP En S1 D S2 DANDP En S1 D S2 DAND En S1 D S2	Logical AND operation
0801	MAND	–	✓	MAND En S1 D S2 N MANDP En S1 D S2 N	Matrix AND operation
0802	WOR	DOR	✓	WOR En S1 D S2 WORP En S1 D S2 DOR En S1 D S2 DORP En S1 D S2	Logical OR operation
0803	MOR	–	✓	MOR En S1 D S2 N MORP En S1 D S2 N	Matrix OR operation
0804	WXOR	DXOR	✓	WXOR En S1 D S2 WXORP En S1 D S2 DXOR En S1 D S2 DXORP En S1 D S2	Logical exclusive OR operation
0805	MXOR	–	✓	MXOR En S1 D S2 N MXORP En S1 D S2 N	Matrix exclusive OR operation
0808	WINV	DINV	✓	WINV En S D WINVP En S D DINV En S D DINVP En S D	Logical reversed INV operation
0809	LD&	DLD&	–	& Q S1 S2 D& Q S1 S2	S_1&S_2
0810	LD|	DLD|	–	| Q S1 S2 D| Q S1 S2	S_1|S_2

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0811	LD^	DLD^	–	^ Q S1 S2 / D^ Q S1 S2	S_1^S_2
0812	AND&	DAND&	–	& Q S1 S2 / D& Q S1 S2	S_1&S_2
0813	AND|	DAND|	–	| Q S1 S2 / D| Q S1 S2	S_1|S_2
0814	AND^	DAND^	–	^ Q S1 S2 / D^ Q S1 S2	S_1^S_2
0815	OR&	DOR&	–	& Q S1 S2 / D& Q S1 S2	S_1&S_2
0816	OR|	DOR|	–	| Q S1 S2 / D| Q S1 S2	S_1|S_2
0817	OR^	DOR^	–	^ Q S1 S2 / D^ Q S1 S2	S_1^S_2

- Rotation instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0900	ROR	DROR	✓	ROR En D n / RORP En D n / DROR En D n / DRORP En D n	Rotating to the right
0901	RCR	DRCR	✓	RCR En D n / RCRP En D n / DRCR En D n / DRCRP En D n	Rotating to the right with the carry flag
0902	ROL	DROL	✓	ROL En D n / ROLP En D n / DROL En D n / DROLP En D n	Rotating to the left

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0903	RCL	DRCL	✓	RCL En D n RCLP En D n DRCL En D n DRCLP En D n	Rotating to the left with the carry flag
0904	MBR	–	✓	MBR En S D n MBRP En S D n	Rotating the matrix bits

- Timer and counter instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1000	RST	DRST	–	Device —(R) DRST En D	Resetting the contact to OFF or clearing the value in the register.
1001	TMR	–	–	TMR En S D	16-bit timer (Unit: 100ms)
1002	TMRH	–	–	TMRH En S D	16-bit timer (Unit: 1ms)
1003	CNT	–	–	CNT En S D	16-bit counter
1004	–	DCNT	–	DCNT En S D	32-bit counter (Including the use of high-speed counters)
1005	–	DHSCS	–	DHSCS En S1 D S2	Setting high-speed comparison
1006	–	DHSCR	–	DHSCR En S1 D S2	Resetting high-speed comparison
1007	–	DHSZ	–	DHSZ En S1 D S2 S	High-speed input zone comparison
1008	–	DSPD	–	DSPD En S1 D S2	Speed detection
1009	PWD	–	–	PWD En S1 D1 S2 D2	Pulse Width Detection
1010	–	DCAP	–	DCAP En S1 D S2	Capturing the high-speed count value in the external input interrupt
1011	TMRM	–	–	TMRM En S1 S2	16-bit timer (Unit: 10ms)

- Shift instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1100	SFTR	–	✓	SFTR En S D n1 n2 SFTRP En S D n1 n2	Shifting the states of the devices to the right
1101	SFTL	–	✓	SFTL En S D n1 n2 SFTLP En S D n1 n2	Shifting the states of the devices to the left
1102	WSFR	–	✓	WSFR En S D n1 n2 WSFRP En S D n1 n2	Shifting the data in the word devices to the right
1103	WSFL	–	✓	WSFL En S D n1 n2 WSFLP En S D n1 n2	Shifting the data in the word devices to the left
1104	SFWR	–	✓	SFWR En S D n SFWRP En S D n	Shifting the data and writing it into the word device
1105	SFRD	–	✓	SFRD En S D n SFRDP En S D n	Shifting the data and reading it from the word device
1106	SFPO	–	✓	SFPO En S D SFPOP En S D	Reading the latest data from the data list
1107	SFDEL	–	✓	SFDEL En S D n SFDELP En S D n	Deleting the data from the data list
1108	SFINS	–	✓	SFINS En S D n SFINSP En S D n	Inserting the data into the data list
1109	MBS	–	✓	MBS En S D n MBSP En S D n	Shifting the matrix bits
1110	SFR	–	✓	SFR En n D SFRP En n D	Shifting the values of the bits in the 16-bit registers by **n** bits to the right
1111	SFL	–	✓	SFL En n D SFLP En n D	Shifting the values of the bits in the 16-bit registers by **n** bits to the left
1112	BSFR	–	✓	BSFR En n D BSFRP En n D	Shifting the states of the **n** bit devices by one bit to the right
1113	BSFL	–	✓	BSFL En n D BSFLP En n D	Shifting the states of the **n** bit devices by one bit to the left

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1114	NSFR	—	✓	NSFR En n D NSFRP En n D	Shifting **n** registers to the right
1115	NSFL	—	✓	NSFL En n D NSFLP En n D	Shifting **n** registers to the left

- Data processing instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1200	SER	DSER	✓	SER En S1 D S2 N DSER En S1 D S2 N SERP En S1 D S2 N DSERP En S1 D S2 N	Searching the data
1201	SUM	DSUM	✓	SUM En S D SUMP En S D DSUM En S D DSUMP En S D	Number of bits whose states are ON
1202	DECO	—	✓	DECO En S D n DECOP En S D n	Decoder
1203	ENCO	—	✓	ENCO En S D n ENCOP En S D n	Encoder
1204	SEGD	—	✓	SEGD En S D SEGDP En S D	Seven-segment decoding
1205	SORT	DSORT	✓	SORT En S D m1 m2 n DSORT En S D m1 m2 n SORTP En S D m1 m2 n DSORTP En S D m1 m2 n	Sorting the data
1206	ZRST	—	✓	ZRST En D1 D2 ZRSTP En D1 D2	Resetting the zone
1207	BON	DBON	✓	BON En S D n BONP En S D n DBON En S D n DBONP En S D n	Checking the state of the bit

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1208	MEAN	DMEAN	✓	MEAN En S n D MEANP En S n D DMEAN En S n D DMEANP En S n D	Mean
1209	CCD	–	✓	CCD En S n D CCDP En S n D	Sum check
1210	ABS	DABS	✓	ABS En D ABSP En D DABS En D DABSP En D	Absolute value
1211	MINV	–	✓	MINV En S n D MINVP En S n D	Inverting the matrix bits
1212	MBRD	–	✓	MBRD En S n D MBRDP En S n D	Reading the matrix bit
1213	MBWR	–	✓	MBWR En S n D MBWRP En S n D	Writing the matrix bit
1214	MBC	–	✓	MBC En S n D MBCP En S n D	Counting the bits with the value 0 or 1
1215	DIS	–	✓	DIS En S n D DISP En S n D	Disuniting the 16-bit data
1216	UNI	–	✓	UNI En S n D UNIP En S n D	Uniting the 16-bit data
1217	WSUM	DWSUM	✓	WSUM En S n D WSUMP En S n D DWSUM En S n D DWSUMP En S n D	Getting the sum
1221	LIMIT	DLIMIT	✓	LIMIT En S1 S2 S3 D LIMITP En S1 S2 S3 D DLIMIT En S1 S2 S3 D DLIMITP En S1 S2 S3 D	Confining the value within the bounds

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1222	BAND	DBAND	✓	BAND En S1 D S2 S3 BANDP En S1 D S2 S3 DBAND En S1 D S2 S3 DBANDP En S1 D S2 S3	Deadband control
1223	ZONE	DZONE	✓	ZONE En S1 D S2 S3 ZONEP En S1 D S2 S3 DZONE En S1 D S2 S3 DZONEP En S1 D S2 S3	Controlling the zone
1224	–	FMEAN	✓	FMEAN En S D n FMEANP En S D n	The mean of the floating points
1225	–	FSUM	✓	FSUM En S D n FSUMP En S D n	The sum of the floating points

- Structure creation instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1300	FOR	–	–	FOR S	Start of the nested loop
1301	NEXT	–	–	NEXT	End of the nested loop
1302	BREAK	–	–	BREAK En D P	Terminating the FOR-NEXT loop

- Module instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1400	FROM	DFROM	✓	FROM En m1 D m2 m3 n FROMP En m1 D m2 m3 n DFROM En m1 D m2 m3 n DFROMP En m1 D m2 m3 n	Reading the data from the control register in the special module

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1401	TO	DTO	✓	TO: En, m1, m2, m3, S, n TOP: En, m1, m2, m3, S, n DTO: En, m1, m2, m3, S, n DTOP: En, m1, m2, m3, S, n	Writing the data into the control register in the special module

- Floating-point number instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1500	–	FSIN	✓	FSIN: En, S, D FSINP: En, S, D	Sine of the floating-point number
1501	–	FCOS	✓	FCOS: En, S, D FCOSP: En, S, D	Cosine of the floating-point number
1502	–	FTAN	✓	FTAN: En, S, D FTANP: En, S, D	Tangent of the floating-point number
1503	–	FASIN	✓	FASIN: En, S, D FASINP: En, S, D	Arcsine of the floating-point number
1504	–	FACOS	✓	FACOS: En, S, D FACOSP: En, S, D	Arccosine of the floating-point number
1505	–	FATAN	✓	FATAN: En, S, D FATANP: En, S, D	Arctangent of the floating-point number
1506	–	FSINH	✓	FSINH: En, S, D FSINHP: En, S, D	Hyperbolic sine of the floating-point number
1507	–	FCOSH	✓	FCOSH: En, S, D FCOSHP: En, S, D	Hyperbolic cosine of the floating-point number
1508	–	FTANH	✓	FTANH: En, S, D FTANHP: En, S, D	Hyperbolic tangent of the floating-point number
1509	–	FRAD	✓	FRAD: En, S, D FRADP: En, S, D	Converting the degree to the radian
1510	–	FDEG	✓	FDEG: En, S, D FDEGP: En, S, D	Converting the radian to the degree
1511	SQR	DSQR	✓	SQR: En, S, D SQRP: En, S, D DSQR: En, S, D DSQRP: En, S, D	Square root of the binary number
1512	–	FSQR	✓	FSQR: En, S, D FSQRP: En, S, D	Square root of the floating-point number

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1513	–	FEXP	✓	FEXP En S D / FEXPP En S D	Exponent of the floating-point number
1514	–	FLOG	✓	FLOG En S1 S2 D / FLOGP En S1 S2 D	Logarithm of the floating-point number
1515	–	FLN	✓	FLN En S D / FLNP En S D	Natural logarithm of the binary floating-point number
1516	–	FPOW	✓	FPOW En S1 S2 D / FPOWP En S1 S2 D	Power of the floating-point number
1517	RAND	–	✓	RAND En S1 S2 D / RANDP En S1 S2 D	Random number

- Real-time clock instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1600	TRD	–	✓	TRD En D / TRDP En D	Reading the time
1601	TWR	–	✓	TWR En S / TWRP En S	Writing the time
1602	T+	–	✓	T+ En S1 S2 D / T+P En S1 S2 D	Adding the time
1603	T-	–	✓	T- En S1 S2 D / T-P En S1 S2 D	Subtracting the time
1604	HOUR	–	–	HOUR En S D1 D2	Running-time meter
1605	TCMP	–	✓	TCMP En S1 S2 S3 S D / TCMPP En S1 S2 S3 S D	Comparing the time
1606	TZCP	–	✓	TZCP En S1 S2 S D / TZCPP En S1 S2 S D	Time zone comparison
1607	DST	–	✓	DST En S S1 S2 S3 S4 S5 D / DSTP En S S1 S2 S3 S4 S5 D	Daylight saving time

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1608	WWON	—	—	WWON En S1 D S2 S3 S4	Weekly working time setup

- Peripheral instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1700	TKY	DTKY	—	TKY En S D1 D2 DTKY En S D1 D2	Ten-key keypad
1701	HKY	DHKY	—	HKY En S1 D1 S2 D2 D3 DHKY En S1 D1 S2 D2 D3	Sixteen-key keypad
1702	DSW	—	—	DSW En S1 D1 S2 D2 n	DIP switch
1703	ARWS	—	—	ARWS En S1 D1 S2 D2 n	Arrow keys
1704	SEGL	—	—	SEGL En S1 D S2 n	Seven-segment display with latches

- Communication instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1806	LRC	—	—	LRC En S D n	Longitudinal parity check
1807	CRC	—	—	CRC En S D n	Cyclic Redundancy Check
1808	MODRW	—	—	MODRW En S1 S2 S3 S n	Reading/Writing the MODBUS data
1812	COMRS	—	—	COMRS En S1 D1 S2 D2 S3	Sending and receiving communication data

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1813	COMDF	—	✓	COMDF En S1 S2 S3 S4 S5 D COMDFP En S1 S2 S3 S4 S5 D	Setting the communication format for a serial communication port
1814	VFDRW	—	—	VFDRW En S1 S2 S3 S	Serial communication instruction exclusive for Delta AC motor drive
1815	ASDRW	—	—	ASDRW En S1 S2 S3 S	Serial communication instruction exclusive for Delta servo drive
1816	CCONF	—	✓	CCONF En S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 CCONFP En S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11	Setting the parameters in the data exchange table of a communication port
1817	MODRWE	—	—	MODRWE En S1 D S2 S3 S4 S n	Reading and writing Modbus data without any flag used

- Other instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1900	WDT	—	✓	WDT En WDTP En	Watchdog timer
1901	DELAY	—	✓	DELAY En S DELAYP En S	Delaying the execution of the program
1902	GPWM	—	—	GPWM En S1 S2 S2	General pulse width modulation
1904	EPUSH	—	✓	EPUSH En D EPUSHP En D	Storing the contents of the index registers
1905	EPOP	—	✓	EPOP En D EPOPP En D	Reading the data into the index registers

- String processing instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2100	BINDA	DBINDA	✓	BINDA En S D BINDAP En S D DBINDA En S D DBINDAP En S D	Converting the singed decimal number into the ASCII code

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2101	BINHA	DBINHA	✓	BINHA En S D BINHAP En S D DBINHA En S D DBINHAP En S D	Converting the binary hexadecimal number into the hexadecimal ASCII code
2102	BCDDA	DBCDDA	✓	BCDDA En S D BCDDAP En S D DBCDDA En S D DBCDDAP En S D	Converting the binary-coded decimal number into the ASCII code
2103	DABIN	DDABIN	✓	DABIN En S D DABINP En S D DDABIN En S D DDABINP En S D	Converting the signed decimal ASCII code into the signed decimal binary number
2104	HABIN	DHABIN	✓	HABIN En S D HABINP En S D DHABIN En S D DHABINP En S D	Converting the hexadecimal ASCII code into the hexadecimal binary number
2105	DABCD	DDABCD	✓	DABCD En S D DABCDP En S D DDABCD En S D DDABCDP En S D	Converting the ASCII code into the binary-coded decimal number
2106	$LEN	–	✓	$LEN En S D $LENP En S D	Calculating the length of the string
2109	$FSTR	–	✓	$FSTR En S1 D S2 $FSTRP En S1 D S2	Converting the floating-point number into the string
2110	$FVAL	–	✓	$FVAL En S D $FVALP En S D	Converting the string into the floating-point number
2111	$RIGHT	–	✓	$RIGHT En S D n $RIGHTP En S D n	The retrieve of the characters in the string begins from the right.
2112	$LEFT	–	✓	$LEFT En S D n $LEFTP En S D n	The retrieve of the characters in the string begins from the left.
2113	$MIDR	–	✓	$MIDR En S1 D S2 $MIDRP En S1 D S2	Retrieving a part of the string
2115	$SER	–	✓	$SER En S1 D S2 N $SERP En S1 D S2 N	Searching the string
2116	$RPLC	–	✓	$RPLC En S1 D S2 S3 S4 $RPLCP En S1 D S2 S3 S4	Replacing the characters in the string
2117	$DEL	–	✓	$DEL En S1 D S2 S3 $DELP En S1 D S2 S3	Deleting the characters in the string
2118	$CLR	–	✓	$CLR En S $CLRP En S	Clearing the string

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2119	$INS	—	✓	$INS En S1 D S2 S3 $INSP En S1 D S2 S3	Inserting the string

- Ethernet instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2200	SOPEN	—	✓	SOPEN En S1 S2 S3 SOPENP En S1 S2 S3	Opening the socket
2201	SSEND	—	✓	SSEND En S1 S2 SSENDP En S1 S2	Sending the data through the socket
2203	SCLOSE	—	✓	SCLOSE En S1 S2 SCLOSEP En S1 S2	Closing the socket
2204	MSEND	—	✓	MSEND En S1 D S2 S3 MSENDP En S1 D S2 S3	Sending the email
2206	INTOA	—	✓	INTOA En S D INTOAP En S D	Converting the IP address of the integer type into the IP address of the string type
2207	IATON	—	✓	IATON En S D IATONP En S D	Converting the IP address of the string type into the IP address of the integer type
2208	EIPRW	—	—	EIPRW En S1 D1 S2 D2 S3 S4 S5 S6 S7 n S	Reading and writing the Ethernet/IP data
2209	SCONF	—	✓	SCONF En S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 SCONFP En S1 S2 S3 S4 S5 S6 S7 S8 S9 S10	Setting TCP/UDP Socket parameters
2210	MCONF	—	✓	MCONF En S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 MCONFP En S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12	Reading/Writing the Modbus TCP data

- Memory card instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2300	MWRIT	–	✓	MWRIT En C S S1 S2 S3 S4 MWRITP En C S S1 S2 S3 S4	Writing the data from the PLC into the memory card
2301	MREAD	–	✓	MREAD En C D S S1 S2 S3 MREADP En C D S S1 S2 S3	Reading the data from the memory card into the PLC
2302	MTWRIT	–	✓	MTWRIT En C S S1 S2 S3 MTWRITP En C S S1 S2 S3	Writing the string into the memory card
2303	MEMW	–	✓	MEMW En S n D MEMWP En S n D	Writing the data into the file register

- Task control instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2400	TKON	–	✓	TKON En S TKONP En S	Enabling the cyclic task
2401	TKOFF	–	✓	TKOFF En S TKOFFP En S	Disabling the cyclic task

- Sequential function charts (SFC) instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2500	SFCRUN	–	–	SFCRUN En S1 S2 S3	Enabling the SFC
2501	SFCPSE	–	–	SFCPSE En S1 S2	Making SFC to pause
2502	SFCSTP	–	–	SFCSTP En S	Stopping the SFC

- High-speed output instrucitons

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2700	–	DPLSY	–	DPLSY En S1 D S2	High-speed pulse output (without ramp-up/down process)

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2701	–	DPLSR	–	DPLSR En S1 D S2 S3	High-speed pulse output (with ramp-up/down process)
2702	PWM	DPWM	–	PWM En S1 D S2 DPWM En S1 D S2	Pulse width modulation
2703	JOG	DJOG	–	JOG En S1 D1 S2 D2 S3 DJOG En S1 D1 S2 D2 S3	JOG output
2704	–	DZRN	–	DZRN En S1 D S2 S3 S4	Zero return
2705	–	DPLSV	–	DPLSV En S D1 D2	Adjustable pulse output
2706	–	DDRVI	–	DDRVI En S1 D1 S2 D2	Relative position control
2707	–	DDRVA	–	DDRVA En S1 D1 S2 D2	Absolute position control
2708	CSFO	–	–	DCSFO En S1 D1 S2 D2 S3	Catch speed and proportional output
2709	–	DDRVM	–	DDRVM En S1 D1 S2 D2 S3 S4 S5 S6	Mark alignment positioning
2710	–	DPPMR	–	DPPMR En S1 D1 S2 D2 S3	2-Axis relative-coordinate point-to-point synchronized motion
2711	–	DPPMA	–	DPPMA En S1 D1 S2 D2 S3	2-Axis absolute-coordinate point-to-point synchronized motion
2712	–	DCICR	–	DCICR En S1 D1 S2 D2 S3 S4 S5	2-Axis relative-position clockwise arc interpolation
2713	–	DCICA	–	DCICA En S1 D1 S2 D2 S3 S4 S5	2-Axis absolute-position clockwise arc interpolation
2714	–	DCICCR	–	DCICCR En S1 D1 S2 D2 S3 S4 S5	2-Axis relative-position counterclockwise arc interpolation

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2715	–	DCICCA	–	DCICCA En S1 D1 S2 D2 S3 S4 S5	2-Axis absolute-position counterclockwise arc interpolation
2716	–	DCCMR	–	DCCMR En S1 D1 S2 D2 S3 S4	The relative-position circle drawing
2717	–	DCCMA	–	DCCMA En S1 D1 S2 D2 S3 S4	The absolute-position circle drawing
2718	TPO	–	–	TPO En S D1 D2	The position planning table controls the output
2719	–	DTPWS	✓	DTPWS En S1 S2 S3 DTPWSP En S1 S2 S3	Setting single-axis output parameters in the position planning table
2720	–	DTPWL	✓	DTPWL En S1 S2 S3 S4 DTPWLP En S1 S2 S3 S4	Setting linear interpolation parameters in the position planning table
2721	–	DPTWC	✓	DTPWC En S1 S2 S3 S4 S5 DTPWCP En S1 S2 S3 S4 S5	Setting arc interpolation parameters in the position planning table
2723	–	DPPGB	–	DPPGB En S1 D1 S2 D2 S3 S4	Point to point go back and forth

- CANopen Communication Instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2800	INITC	–	–	INITC En S	Initializing the servos for the CANopen communication
2801	ASDON	–	–	ASDON En S1 S2	Servo-ON and servo-OFF
2802	CASD	–	–	CASD En S1 S2 S3	Setting the acceleration time and deceleration time of a servo
2803	–	DDRVIC	–	DDRVIC En S1 S2 S3	Servo relative position control
2804	–	DDRVAC	–	DDRVAC En S1 S2 S3	Servo absolute position control
2805	–	DPLSVC	–	DPLSVC En S1 S2	Servo speed control

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2806	ZRNC	–	–	ZRNC En S1 S2 S3	Homing
2807	COPRW	–	–	COPRW En S1 D S2 S3 S4 S5	Writing and reading the CANopen communication data

3.4.2 Applied Instructions (Sorted Alphabetically)

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
Symbol	0114	$+	–	✓	Linking the strings
	2118	$CLR	–	✓	Clearing the string
	2117	$DEL	–	✓	Deleting the characters in the string
	2109	$FSTR	–	✓	Converting the floating-point number into the string
	2110	$FVAL	–	✓	Converting the string into the floating-point number
	2119	$INS	–	✓	Inserting the string
	2112	$LEFT	–	✓	The retrieve of the characters in the string begins from the left.
	2106	$LEN	–	✓	Calculating the length of the string
	2113	$MIDR	–	✓	Retrieving a part of the string
	0302	$MOV	–	✓	Transferring the string
	2111	$RIGHT	–	✓	The retrieve of the characters in the string begins from the right.
	2116	$RPLC	–	✓	Replacing the characters in the string
	2115	$SER	–	✓	Searching the string
	0102	*	D*	✓	Multiplication of binary numbers
	0103	/	D/	✓	Division of binary numbers
	0100	+	D+	✓	Addition of binary numbers
A	1210	ABS	DABS	✓	Absolute value
	0705	ABSD	DABSD	–	Absolute drum sequencer
	0700	ALT	–	✓	Alternating between ON and OFF
	0043	AND$<>	–	–	S1≠S2
	0042	AND$=	–	–	S1 = S2
	0812	AND&	DAND&	–	S1&S2
	0814	AND^	DAND^	–	S1^S2
	0813	AND|	DAND|	–	S1|S2
	0010	AND<	DAND<	–	S1 < S2

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	0011	AND<=	DAND<=	–	S1≦S2
	0007	AND<>	DAND<>	–	S1≠S2
	0006	AND=	DAND=	–	S1 = S2
	0008	AND>	DAND>	–	S1 > S2
	0009	AND>=	DAND>=	–	S1≧S2
	0076	ANDZ<	DANDZ<	–	|S1-S2| < |S3|
	0077	ANDZ<=	DANDZ<=	–	|S1-S2|≦|S3|
	0073	ANDZ<>	DANDZ<>	–	|S1-S2|≠|S3|
	0072	ANDZ=	DANDZ=	–	|S1-S2|=|S3|
	0074	ANDZ>	DANDZ>	–	|S1-S2| > |S3|
	0075	ANDZ>=	DANDZ>=	–	|S1-S2|≧|S3|
	1703	ARWS	–	–	Arrow key input
	2801	ASDON	–	–	Servo-ON and servo-OFF
	1815	ASDRW	–	–	Serial communication instruction exclusive for Delta servo drive
B	1222	BAND	DBAND	✓	Deadband control
	0200	BCD	DBCD	✓	Converting the binary number into the binary-coded decimal number
	2102	BCDDA	DBCDDA	✓	Converting the binary-coded decimal number into the ASCII code
	0201	BIN	DBIN	✓	Converting the binary-coded decimal number into the binary number
	2100	BINDA	DBINDA	✓	Converting the singed decimal number into the ASCII code
	2101	BINHA	DBINHA	✓	Converting the binary hexadecimal number into the hexadecimal ASCII code
	0113	BK-	DBK-	✓	Subtraction of binary numbers in blocks
	0112	BK+	DBK+	✓	Addition of binary numbers in blocks
	0214	BKBCD	–	✓	Converting the binary numbers in blocks into the binary-coded decimal numbers in blocks
	0215	BKBIN	–	✓	Converting the binary numbers in blocks into the binary-coded decimal numbers in blocks
	0304	BMOV	DBMOV	✓	Transferring all data
	1207	BON	DBON	✓	Checking the state of the bit
	1302	BREAK	–	–	Terminating the FOR-NEXT loop
	1113	BSFL	–	✓	Shifting the states of the n bit devices by one bit to the left
	1112	BSFR	–	✓	Shifting the states of the n bit devices by one bit to the right
	0307	BXCH	–	✓	Exchanging all data
C	2802	CASD	–	–	Setting the acceleration time and deceleration time

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
					of a servo
	1209	CCD	–	✓	Sum check
	1816	CCONF	–	✓	Setting the parameters in the data exchange table of a communication port
	0065	CHKADR	–	–	Checking the address of the contact type of pointer register
	0400	CJ	–	✓	Conditional jump
	0303	CML	DCML	✓	Inverting the data
	0054	CMP	DCMP	✓	Comparing the values
	0063	CMPT<	–	✓	Comparing the tables ON: <
	0064	CMPT<=	–	✓	Comparing the tables ON: ≦
	0060	CMPT<>	–	✓	Comparing the tables ON: ≠
	0059	CMPT=	–	✓	Comparing the tables ON: =
	0061	CMPT>	–	✓	Comparing the tables ON: >
	0062	CMPT>=	–	✓	Comparing the tables ON: ≧
	1003	CNT	–	–	16-bit counter
	1813	COMDF	–	✓	Setting the communication format for a serial communication port
	1812	COMRS	–	–	Sending and receiving communication data
	2807	COPRW	–	–	Writing and reading the CANopen communication data
	1807	CRC	–	–	Cyclic Redundancy Check
	2708	CSFO	–	–	Catch speed and proportional output
D	0101	–	D-	✓	Subtraction of binary numbers S_1-S_2=D
	2717	–	DCCMA	–	The absolute-position circle drawing
	2716	–	DCCMR	–	The relative-position circle drawing
	2713	–	DCICA	–	2-Axis absolute-position clockwise arc interpolation
	2715	–	DCICCA	–	2-Axis absolute-position counterclockwise arc interpolation
	2714	–	DCICCR	–	2-Axis relative-position counterclockwise arc interpolation
	2712	–	DCICR	–	2-Axis relative-position clockwise arc interpolation
	1004	–	DCNT	–	32-bit counter

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	2707	–	DDRVA	–	Absolute position control
	2804	–	DDRVAC	–	Servo absolute position control
	2706	–	DDRVI	–	Relative position control
	2803	–	DDRVIC	–	Servo relative position control
	2709	–	DDRVM	–	Mark alignment positioning
	1006	–	DHSCR	–	Resetting high-speed comparison
	1005	–	DHSCS	–	Setting high-speed comparison
	1007	–	DHSZ	–	High-speed input zone comparison
	0708	–	DPIDE	–	PID algorithm
	2701	–	DPLSR	–	High-speed pulse output (with ramp-up/down process)
	2705	–	DPLSV	–	Adjustable pulse output
	2805	–	DPLSVC	–	Servo speed control
	2700	–	DPLSY	–	High-speed pulse output (without ramp-up/down process)
	2723	–	DPPGB	–	Point to point go back and forth
	2711	–	DPPMA	–	2-Axis absolute-coordinate point-to-point synchronized motion
	2710	–	DPPMR	–	2-Axis relative-coordinate point-to-point synchronized motion
	2721	–	DPTWC	✓	Setting arc interpolation parameters in the position planning table
	2720	–	DTPWL	✓	Setting linear interpolation parameters in the position planning table
	2719	–	DTPWS	✓	Setting single-axis output parameters in the position planning table
	2704	–	DZRN	–	Zero return
	2105	DABCD	DDABCD	✓	Converting the ASCII code into the binary-coded decimal number
	2103	DABIN	DDABIN	✓	Converting the signed decimal ASCII code into the signed decimal binary number
	0116	DEC	DDEC	✓	Subtracting one from the binary number
	1202	DECO	–	✓	Decoder
	1901	DELAY	–	✓	Delaying the execution of the program
	0500	DI	–	–	Disabling the interrupt
D	1215	DIS	–	✓	Disuniting the 16-bit data
	0118	DIV16	DIV32	✓	Division of binary numbers for 16-bit Division of binary numbers for 32-bit
	0504	DIX	–	–	Enabling the specific interrupt
	1607	DST	–	✓	Daylight saving time
	1702	DSW	–	–	DIP switch

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
E	0501	EI	–	–	Enabling the interrupt
	2208	EIPRW	–	–	Reading and writing via Ethernet/IP connection
	0503	EIX	–	–	Disabling the specific interrupt
	1203	ENCO	–	✓	Encoder
	1905	EPOP	–	✓	Reading the data into the index registers
	1904	EPUSH	–	✓	Storing the contents of the index registers
F	0105	–	F-	✓	Subtraction of floating-point numbers S_1-S_2=D
	0106	–	F*	✓	Multiplication of floating-point numbers S_1*S_2=D
	0107	–	F/	✓	Division of floating-point numbers S_1/S_2=D
	0104	–	F+	✓	Addition of floating-point numbers S1+S2=D
	1504	–	FACOS	✓	Arccosine of the floating-point number
	0028	–	FAND<	–	S1 < S2
	0029	–	FAND<=	–	S1≦S2
	0025	–	FAND<>	–	S1≠S2
	0024	–	FAND=	–	S1 = S2
	0026	–	FAND>	–	S1 > S2
	0027	–	FAND>=	–	S1≧S2
	1503	–	FASIN	✓	Arcsine of the floating-point number
	1505	–	FATAN	✓	Arctangent of the floating-point number
	0212	–	FBCD	✓	Converting the binary floating-point number into the decimal floating-point number
	0213	–	FBIN	✓	Converting the decimal floating-point number into the binary floating-point number
	0056	–	FCMP	✓	Comparing the floating-point numbers
	1501	–	FCOS	✓	Cosine of the floating-point number
	1507	–	FCOSH	✓	Hyperbolic cosine of the floating-point number
	1510	–	FDEG	✓	Converting the radian to the degree
	1513	–	FEXP	✓	An exponent of the floating-point number
	0022	–	FLD<	–	S1 < S2
	0023	–	FLD<=	–	S1≦S2
	0019	–	FLD<>	–	S1≠S2
F	0018	–	FLD=	–	S1 = S2
	0020	–	FLD>	–	S1 > S2
	0021	–	FLD>=	–	S1≧S2

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	1515	–	FLN	✓	Natural logarithm of the binary floating-point number
	1514	–	FLOG	✓	Logarithm of the floating-point number
	1224	–	FMEAN	✓	The mean of the floating points
	0211	–	FNEG	✓	Reversing the sign of the floating-point number
	0034	–	FOR<	–	S1 < S2
	0035	–	FOR<=	–	S1≦S2
	0031	–	FOR<>	–	S1≠S2
	0030	–	FOR=	–	S1 = S2
	0032	–	FOR>	–	S1 > S2
	0033	–	FOR>=	–	S1≧S2
	1516	–	FPOW	✓	A power of the floating-point number
	1509	–	FRAD	✓	Converting the degree to the radian
	1500	–	FSIN	✓	Sine of the floating-point number
	1506	–	FSINH	✓	Hyperbolic sine of the floating-point number
	1512	–	FSQR	✓	Square root of the floating-point number
	1225	–	FSUM	✓	The sum of the floating points
	1502	–	FTAN	✓	Tangent of the floating-point number
	1508	–	FTANH	✓	Hyperbolic tangent of the floating-point number
	0057	–	FZCP	✓	Floating-point zone comparison
	0202	FLT	DFLT	✓	Converting the binary integer into the binary floating-point number
	1300	FOR	–	–	Start of the nested loop
	1400	FROM	DFROM	✓	Reading the data from the control register in the extension module
G	0209	GBIN	DGBIN	✓	Converting the Gray code into the binary number
	0402	GOEND	–	–	Jumping to END
	1902	GPWM	–	–	General pulse width modulation
	0208	GRY	DGRY	✓	Converting the binary number into the Gray code
H	2104	HABIN	DHABIN	✓	Converting the hexadecimal ASCII code into the hexadecimal binary number
	1701	HKY	DHKY	–	Hexadecimal key input
	1604	HOUR	–	–	Running-time meter
I	2207	IATON	–	✓	Converting the IP address of the string type into the IP address of the integer type
	0115	INC	DINC	✓	Adding one to the binary number
	0706	INCD	–	–	Incremental drum sequencer
I	2800	INITC	–	–	Initializing the servos for the CANopen communication

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	0204	INT	DINT	✓	Converting the 32-bit floating-point number into the binary integer
	2206	INTOA	–	✓	Converting the IP address of the integer type into the IP address of the string type
J	0401	JMP	–	–	Unconditional jump
	2703	JOG	DJOG	–	JOG output
L	0037	LD$<>	–	–	S1≠S2
	0036	LD$=	–	–	S1 = S2
	0809	LD&	DLD&	–	S_1&S_2
	0811	LD^	DLD^	–	S_1^S_2
	0810	LD|	DLD|	–	S_1|S_2
	0004	LD<	DLD<	–	S1 < S2
	0005	LD<=	DLD<=	–	S1≦S2
	0001	LD<>	DLD<>	–	S1≠S2
	0000	LD=	DLD=	–	S1 = S2
	0002	LD>	DLD>	–	S1 > S2
	0003	LD>=	DLD>=	–	S1≧S2
	0070	LDZ<	DLDZ<	–	|S1-S2| < |S3|
	0071	LDZ<=	DLDZ<=	–	|S1-S2|≦|S3|
	0067	LDZ<>	DLDZ<>	–	|S1-S2|≠|S3|
	0066	LDZ=	DLDZ=	–	|S1-S2|=|S3|
	0068	LDZ>	DLDZ>	–	|S1-S2| > |S3|
	0069	LDZ>=	DLDZ>=	–	|S1-S2|≧|S3|
	1221	LIMIT	DLIMIT	✓	Confining the value within the bounds
	1806	LRC	–	–	Longitudinal parity check
M	0801	MAND	–	✓	Matrix AND operation
	1214	MBC	–	✓	Counting the bits with the value 0 or 1
	0904	MBR	–	✓	Rotating the matrix bits
	1212	MBRD	–	✓	Reading the matrix bit
	1109	MBS	–	✓	Shifting the matrix bits
	1213	MBWR	–	✓	Writing the matrix bit
	0058	MCMP	–	✓	Matrix comparison
	2210	MCONF	–	✓	Reading/Writing the Modbus TCP data
	1208	MEAN	DMEAN	✓	Mean
	2303	MEMW	–	✓	Writing the data into the file register
	1211	MINV	–	✓	Inverting the matrix bits

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	0206	MMOV	–	✓	Converting the 16-bit value into the 32-bit value
	1808	MODRW	–	–	Reading/Writing the MODBUS data
	1817	MODRWE	–	–	Reading and writing Modbus data without any flag used
	0803	MOR	–	✓	Matrix OR operation
	0300	MOV	DMOV	✓	Transferring the data
	0310	MOVB	–	✓	Transferring several bits
	2301	MREAD	–	✓	Reading the data from the memory card into the PLC
	2204	MSEND	–	✓	Sending the email
	0704	MTR	–	–	Matrix input
	2302	MTWRIT	–	✓	Writing the string into the memory card
	0117	MUL16	MUL32	✓	Multiplication of binary numbers for 16-bit/32-bit
	2300	MWRIT	–	✓	Writing the data from the PLC into the memory card
	0805	MXOR	–	✓	Matrix exclusive OR operation
N	0210	NEG	DNEG	✓	Two's complement
	1301	NEXT	–	–	End of the nested loop
	0305	NMOV	DNMOV	✓	Transferring the data to several devices
	1115	NSFL	–	✓	Shifting **n** registers to the left
	1114	NSFR	–	✓	Shifting **n** registers to the right
O	0049	OR$<>	–	–	S1≠S2
	0048	OR$=	–	–	S1 = S2
	0815	OR&	DOR&	–	S1&S2
	0817	OR^	DOR^	–	S1^S2
	0816	OR|	DOR|	–	S1|S2
	0016	OR<	DOR<	–	S1 < S2
	0017	OR<=	DOR<=	–	S1≦S2
	0013	OR<>	DOR<>	–	S1 ≠ S2
	0012	OR=	DOR=	–	S1 = S2
	0014	OR>	DOR>	–	S1 > S2
	0015	OR>=	DOR>=	–	S1≧S2
	0082	ORZ<	DORZ<	–	|S1-S2| < |S3|
	0083	ORZ<=	DORZ<=	–	|S1-S2|≦|S3|
	0079	ORZ<>	DORZ<>	–	|S1-S2|≠|S3|
	0078	ORZ=	DORZ=	–	|S1-S2|=|S3|
	0080	ORZ>	DORZ>	–	|S1-S2| > |S3|

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	0081	ORZ>=	DORZ>=	–	|S1-S2|≧|S3|
P	2702	PWM	DPWM	–	Pulse width modulation
R	0703	RAMP	DRAMP	–	Ramp signal
	1517	RAND	–	✓	Random number
	0903	RCL	DRCL	✓	Rotating to the left with the carry flag
	0901	RCR	DRCR	✓	Rotating to the right with the carry flag
	0600	REF	–	✓	Refreshing the I/O
	0207	RMOV	–	✓	Converting the 32-bit value into the 16-bit value
	0902	ROL	DROL	✓	Rotating to the left
	0900	ROR	DROR	✓	Rotating to the right
	1000	RST	DRST	–	Resetting the contact or clearing the register
S	0216	SCAL	DSCAL	✓	Scale value operation
	2203	SCLOSE	–	✓	Closing the socket
	0217	SCLP	DSCLP	✓	Parameter type of scale value operation
	2209	SCONF	–	✓	Setting TCP/UDP Socket parameters
	1204	SEGD	–	✓	Seven-segment decoding
	1704	SEGL	–	–	Seven-segment display with latches
	1200	SER	DSER	✓	Searching the data
	2501	SFCPSE	–	–	Making SFC to pause
	2500	SFCRUN	–	–	Enabling the SFC
	2502	SFCSTP	–	–	Stopping the SFC
	1107	SFDEL	–	✓	Deleting the data from the data list
	1108	SFINS	–	✓	Inserting the data into the data list
	1111	SFL	–	✓	Shifting the values of the bits in the 16-bit registers by n bits to the left
	1106	SFPO	–	✓	Reading the latest data from the data list
	1110	SFR	–	✓	Shifting the values of the bits in the 16-bit registers by n bits to the right
	1105	SFRD	–	✓	Shifting the data and reading it from the word device
	1101	SFTL	–	✓	Shifting the states of the devices to the left
	1100	SFTR	–	✓	Shifting the states of the devices to the right
	1104	SFWR	–	✓	Shifting the data and writing it into the word device
	0309	SMOV	–	✓	Transferring the digits
	2200	SOPEN	–	✓	Opening the socket
	1205	SORT	DSORT	✓	Sorting the data
	1511	SQR	DSQR	✓	Square root of the binary number

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	2201	SSEND	–	✓	Sending the data through the socket
	0702	STMR	–	–	Special timer
	1201	SUM	DSUM	✓	Number of bits whose states are ON
	0308	SWAP	DSWAP	✓	Exchange the high byte with the low byte
T	1603	T-	–	✓	Subtracting the time
	1602	T+	–	✓	Adding the time
	1605	TCMP	–	✓	Comparing the time
	2401	TKOFF	–	✓	Disabling the cyclic task
	2400	TKON	–	✓	Enabling the cyclic task
	1700	TKY	DTKY	–	Ten key input
	1001	TMR	–	–	16-bit timer
	1002	TMRH	–	–	16-bit timer
	1401	TO	DTO	✓	Writing the data into the control register in the special module
	2718	TPO	–	–	The position planning table controls the output
	1600	TRD	–	✓	Reading the time
	0701	TTMR	–	–	Teaching timer
	1601	TWR	–	✓	Writing the time
	1606	TZCP	–	✓	Time zone comparison
U	1216	UNI	–	✓	Uniting the 16-bit data
V	1814	VFDRW	–	–	Serial communication instruction exclusive for Delta AC motor drive
W	0800	WAND	DAND	✓	Logical AND operation
	1608	WWON	–	–	Weekly working time setup
	1900	WDT	–	✓	Watchdog timer
	0808	WINV	DINV	✓	Logical reversed INV operation
	0802	WOR	DOR	✓	Logical OR operation
	1103	WSFL	–	✓	Shifting the data in the word devices to the left
	1102	WSFR	–	✓	Shifting the data in the word devices to the right
	1217	WSUM	DWSUM	✓	Getting the sum
	0804	WXOR	DXOR	✓	Logical exclusive OR operation
X	0306	XCH	DXCH	✓	Exchanging the data
	0709	XCMP	–	–	Setups for comparing the inputs of multiple work stations
Y	0710	YOUT	–	–	Comparing the outputs of multiple work stations
Z	0055	ZCP	DZCP	✓	Zone comparison
	1223	ZONE	DZONE	✓	Controlling the zone

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	2806	ZRNC	–	–	Homing
	1206	ZRST	–	✓	Resetting the zone

MEMO

4

Chapter 4 Instruction Structure

Table of Contents

4.1 API Composition of Applied Instructions 4-2

4.2 Operand Usage Description 4-5

4.3 Restrictions on the Use of the Instructions 4-6

4.4 Index Registers 4-7

4.5 Pointer Registers 4-9

4.6 Pointer Registers of Timers 4-11

4.7 Pointer Registers of 16-bit Counters 4-12

4.8 Pointer Registers of 32-bit Counters 4-14

4.9 File Register 4-15

4.1 API Composition of Applied Instructions

Every instruction has its own instruction code and API number. The API number of the instruction in the following table is 0300, and the instruction code is MOV, whose function is transferring the data.

API	Instruction code			Operand	Function
0300	D	MOV	P	**S**、**D**	Transferring the data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○	○	○	○		○
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●		●		●	●	
D		●	●		●	●	●		●		●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Data source

D : Data destination

1. The devices used by the instruction are listed in the operand column. **S**, **D**, **n**, and **m** are used as the operands according to their functions. When more than one operand is used, and these operands share the same function, they are suffixed with numbers. For example, $\mathbf{S_1}$, $\mathbf{S_2}$, and etc.
2. If the instruction can be used as the pulse instruction, the letter P is added in back of the instruction. If the 16-bit instruction can be used as the 32-bit instruction, the letter D is added in front of the 16-bit instruction to form the 32-bit instruction. For example, “D***P” in which “***” is an instruction code.
3. F in the operand area represents the single precision floating point (32-bit)
4. The solid circle ● indicates that the device can be modified by an index register, and the hollow circle ○ indicates that the device cannot be modified by an index register. For example, the data register designated by the operand **S** can be modified by an index register.
5. The applicable model is indicated in the table. Users can check whether the instruction can be used as the pulse instruction, the 16-bit instruction, the 32-bit instruction, or the 64-bit instruction according to the information in the table.
6. If users want to use an instruction in the function block, and the output, input and data devices are supported among the operands, users have to use the pointer registers. AS for the timer, the 16-bit counter, and the 32-bit counter that are supported among the operands, users have to use the pointer register of the timer, the pointer register of

the 16-bit counter, and the pointer register of the 32-bit counter. Please refer to sections 4.4~4.7 for more information or section 7.2.4 in ISPSoft.

7. The description of the symbols representing the instruction MOV in ISPSoft:

   MOV, MOVP, DMOV, and DMOVP: Instruction codes

   En: Enable

   S: The data source (The applicable format of the operand is a word/double word.)

   D: The data destination (The applicable format of the operand is a word/double word.)

The composition of applied instructions:

Some applied instructions are composed of instruction codes. For example, the instructions EI, DI, WDT, and etc. however, most applied instructions consist of instruction codes and several operands.

Every applied instruction has its own API number and instruction code. For example, the instruction code of API0300 is MOV (transferring the data).

Entering the instruction directly: Users can enter the instruction by means of ISPSoft. For the instruction MOV, users only need to enter the instruction name and the operands to designate “MOV D0 D1”.

Entering the instruction by dragging: Users can drag the instruction MOV from **APIs** in ISPSoft to the area where the ladder diagram can be edited.

Entering the instruction by the toolbar: Users can click **API/FB Selection** on the toolbar in ISPSoft, and then choose **API**. Finally, they can choose the instruction MOV in **Data Transfer**. The operands are extra designated.

S	Source operand If there is more than one source operand, these source operands are represented by $\mathbf{S_1}$, $\mathbf{S_2}$, and etc.
D	Destination operand If there is more than one destination operand, these destination operand is represented by $\mathbf{D_1}$, $\mathbf{D_2}$, and etc.
If the operand only can designate the constant K/H or the register, it is represented by **m**, $\mathbf{m_1}$, $\mathbf{m_2}$, **n**, $\mathbf{n_1}$, or $\mathbf{n_2}$.	

The length of the operand (the 16-bit instruction, the 32-bit instruction, or the floating-point number instruction):

The 16-bit instruction or the 32-bit instruction

The values of the operands can be divided into the 16-bit values and the 32-bit values. Accordingly, in order to process data of difference lengths, the instructions are divided into the 16-bit instructions and the 32-bit instructions. To separate the 32-bit instruction from the 16-bit one, a D is added in front of the 16-bit instruction.

16-bit instruction MOV

When M1 is ON, the data in D0 is transferred to D1.

32-bit instruction DMOV

When M1 is ON, the data in (D1, D0) is transferred to (D3, D2).

The floating-point number instruction

The floating-point number instructions can support the 32-bit floating-point number instructions which correspond to the single-precision floating-point number instructions. Users can refer to chapter 2 for more information about the floating-point numbers.

32-bit single-precision floating-point number instruction F+

When X0.0 is ON, the data in (D11, D10) and (D21, D20) is transferred to (D31, D30).

The continuous execution of the instruction and the pulse execution of the instruction:

1. The execution of the instructions can be divided into the continuous execution and the pulse execution. When the instruction is not executed, the time needed to execute the program is shorter. Therefore, using the pulse instruction in the program can lessen the scan cycle.
2. The pulse function allows the related instruction to enable the rising edge-triggered control input. The instruction is ON within one scan cycle.
3. If the control input stays ON, and the related instruction is not executed, the control input has to be switched from OFF to ON again in order to execute the instruction.
4. The pulse instruction:

Pulse execution

When M1 is switched from OFF to ON, the instruction MOVP is executed once. The instruction is not executed any more within the scan cycle. Therefore, it is called the pulse instruction.

Continuous execution

Whenever M1 is ON during the scan cycle, the instruction MOV is executed once. Therefore, the instruction is called the continuous instruction.

When the conditonal contact M1 is OFF, the instruction is not executed, and the value in the destination operand D does not change.

4.2 Operand Usage Description

There are 2 types of operands in AS series; one type is for users to designate and the other one is for system to define.

Operands for users to designate are listed below:

1. Input relay: X0.0 ~ X63.15 or X0 ~ X63
2. Output relay: Y0.0 ~ Y63.15 or Y0 ~ Y63
3. Internal relay: M0 ~ M8191
4. Stepping relay: S0 ~ S2047
5. Timer: T0 ~ T511
6. 16-bit counter: C0 ~ C511
7. 32-bit counter: HC0 ~ HC255
8. Data register: D0 ~ D29999 or D0.0 ~ D29999.15
9. File register: FR0 ~ FR65535
10. Special auxiliary flag: SM0 ~ SM2047
11. Special data register: SR0 ~ SR2047
12. Index register: E0 ~ E9
13. Constant: The decimal constants are notated by K, and the hexadecimal constants are notated by 16#.
14. String: “$”
15. Floating-point number: The single-precision floating-point numbers are notated by F
16. The length of the data in one register is generally 16 bits. If users want to store the 32-bit data in the register, they have to designate two consecutive registers.
17. If the operand used in the 32-bit instruction designates D0, the 32-bit data register composed of (D1, D0) is occupied. D1 represents the higher 16 bits, and D0 represents the lower 16 bits. The same rule applies to the timer and the 16-bit counter. 。
18. When the 32-bit counter HC is used as the data register, it is only can be designated by the operand used in the 32-bit instruction.
19. The index registers can only be used in the 16-bit instruction.

Please refer to chapter 2 Device for more information.

Operands for the system to define are listed below:

1. The system assigns the variables to declare such as Bool, WORD, INT and so on: U0 ~ U16387 and W0 ~ W29999.
2. To start or stop a task: TK0 ~ TK31
3. Pointer type variable symbols, the supporting devices and usage are listed below.

Pointer type	Usage	
General pointer (Pointer)	Device range	PR0 ~ PR15，PR0.0~PR15.15
	Maximum quantity	Up to 16 pointers can be used in each function block
	Can be assigned to	Variable symbols of WORD/DWORD/LWORD/INT/DINT/LINT types or data register, input relay or output relay devices (e.g. X0, Y0, etc.)
Pointer for a timer (T_POINTER)	Device range	TR0 ~ TR7
	Maximum quantity	Up to 8 pointers can be used in each function block
	Can be assigned to	Variable symbols of timer type or timer type devices
Pointer for a counter (C_POINTER)	Device range	CR0 ~ CR7
	Maximum quantity	Up to 8 pointers can be used in each function block
	Can be assigned to	Variable symbols of counter type or counter type devices
Pointer for a high-speed counter (HC_POINTER)	Device range	HCR0 ~ HCR7
	Maximum quantity	Up to 8 pointers can be used in each function block
	Can be assigned to	Variable symbols of 32-bit counter type or 32-bit counter type devices

4.3 Restrictions on the Use of the Instructions

- The instructions which only can be used in the function blocks

  API0065 CHKADR, FB_NP, FB_PN, NED, ANED, ONED, PED, APED, OPED

- The instructions which cannot be used in the interrupt tasks

  GOEND

- The instructions which are not supported in the function blocks

  LDP, ANDP, ORP, LDF, ANDF, ORF, PLS, PLF, NP, PN, MC/MCR, GOEND and all pulse instructions in applied instructions.

  If users want to use some of the instructions mentioned above, they can use the substitute instructions.

Instruction which cannot be used in the function block	Substitute instruction in the function block
LDP/ANDP/ORP	NED/ANED/ONED
LDF/ANDF/ORF	PED/APED/OPED
PLS	-
PLF	-
NP	FB_NP
PN	FB_PN
MC	-

Instruction which cannot be used in the function block	Substitute instruction in the function block
MCR	-
All pulse instructions in applied commands	*1

*1: Pulse instructions cannot be used in the function blocks. If users want to get the function of the pulse instruction in the function block, they can refer to the following example.

Example:

1. First, declare 10 bit variables tempBit[10] used in the system.
2. When StartBit1 is switched from OFF to ON, method 1 (network 1) and method 2 (network 2) can only execute the instruction MOV once; users can choose one to use.
3. The variable tempBit used in the system cannot be used repeatedly.

4.4 Index Registers

The index register is the 16-bit data register. It is like the general register in that the data can be read from it and written into it. However, it is mainly used as the index register. The range of index registers is E0~E9.

The index register is used as follows.

1. Using the register name to modify the device

   When M0 is ON, E0=10, D0@E0=D (0+10)=D10, and D1=D10.

When M0 is ON, E0=10, E1=17, D1@E0=D (1+10)=D11, D11 is ON.

NOTE 1: AS Series supports using the register name to modify the device for example D0.1@E0 but not supporting 2-layered modification for example, D0@E1.1@E0.

NOTE 2: When E0=17, D0.1@E0=D0.(1+17)=D1.2, and D1.2 will be ON. And the bit part 1@E0=(1+17)=18. However, the maximum bit number is 15. Since m=18/16=1 and the remainder is 2, the last modification result is D (0+1).2=D1.2. D1.2 will be ON.

When M0 is ON, E0=10, and M1@E0=M (1+10)=M11. M11 is ON.

2. Declaring the variables first, and then modifying the device

- Declare the three variables StartBit, Var1, and Var2 in ISPSoft.

  The type of StartBit is the Boolean array, and its size is 2 bits. The range is from StartBit[0] to StartBit[1].

  The type of Var1 is the word array, and its size is 11 words. The range is from Var1[0] to Var1[10].

  The type of Var2 is the word, and its size is one word.

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
VAR	StartBit	N/A [Auto]	ARRAY [2] OF BOOL	N/A	
VAR	Var1	N/A [Auto]	ARRAY [11] OF WORD	N/A	
VAR	Var2	N/A [Auto]	WORD	N/A	

- When StartBit[0] is ON, E0=10, E1=1, Var1[0]@E0=Var1[10], Var2=Var1[10], and StrartBit[0]@E1=StartBit[1]. StartBit[1] is ON.

Additional remark: When users declare the variables in ISPSoft, and the variables are added to the contents of the registers to form the addresses to the actual data, users must note the addresses to prevent the program from being executed wrongly.

4.5 Pointer Registers

- ISPSoft supports the function blocks. When the variable declaration type is VAR_IN_OUT, and the data type is POINTER, the variable is the pointer register. The value in the pointer register can refer directly to the value stored in the device X, Y, or D and the pointer register can point to the address associated with the variable set automatically in ISPSoft.
- Users can declare 16 pointer registers in every function block. The range is PR0~PR15, or PR0.0~PR15.15.

Example:

1. Establish a program organization unit (POU) in ISPSoft first.
2. Establish a function block which is called FB0.

3. The program in the function block FB0

4. Declare the varaible in the function block FB0.

   Choose VAR_IN_OUT as the declaration type, Point1 as the identifier, POINTER as the data type. The variable is the pointer register.

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
VAR	VarBit1	N/A [Auto]	BOOL	FALSE	
VAR	Var1	N/A [Auto]	WORD	0	
VAR	Var2	N/A [Auto]	WORD	0	
VAR_IN_OUT	Point1	N/A [Auto]	POINTER	N/A	

5. Declare the variable in the program organization unit (POU).

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
VAR	StartBit	N/A [Auto]	ARRAY [2] OF BOOL	N/A	
VAR	CVar1	N/A [Auto]	ARRAY [2] OF WORD	N/A	
VAR	MyFB0	N/A [Auto]	FB0	N/A	

6. Call the function block FB0 in the program organization unit (POU).

7. The program in the program organization unit (POU)

   Network 1: When StartBit[0] is ON, the address of D0 is transmitted to Point 1 in FB0.

   When VarBit1 in FB0 is ON, E0=1, Var1=D0, Point1@E0=D (0+1)=D1, and Var2=D1.

   Network 2: When StartBit[1] is ON, the address of CVar1[0] is transmitted to Point1 in FB0.

   Var2=CVar1[1]。When VarBit1 in FB0 is ON, E0=1, Var1=CVar1[0], Point1@E0=CVar1 (0+1)=Cvar1[1], and Var2=CVar1[1].

4.6 Pointer Registers of Timers

- ISPSoft supports the function blocks. If users want to use the timer in the function block, they have to declare a pointer register of the timer in the function block. The address of the timer is transmitted to the pointer register of the timer when the function block is called.
- When the variable declaration type is VAR_IN_OUT, and the data type is T_POINTER, the variable is the pointer register of the timer. The value in the pointer register of the timer can refer directly to the value stored in the device T or in the variable which is the timer in ISPSoft.
- Users can declare 8 pointer registers of the timers in every function block. The range is TR0~TR7.
- If users want to use an instruction in the function block, and the timer is supported among the operands, users have to use the pointer register of the timer.

Example:

1. Establish a program organization unit (POU) in ISPSoft first.
2. Establish a function block which is called FB0.

3. Declare the varaible in the function block FB0.

   Choose VAR_IN_OUT as the declaration type, TPoint1 as the identifier, T_POINTER as the data type. The variable is the pointer register of the timer.

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
VAR	VarBit1	N/A [Auto]	BOOL	FALSE	
VAR_IN_OUT	TPoint1	N/A [Auto]	T_POINTER	N/A	
VAR	VarOut	N/A [Auto]	BOOL	FALSE	

4. The program in the function block FB0

5. Declare the variable in the program organization unit (POU).

   The data type of CVar1 should be TIMER.

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
VAR	StartBit	N/A [Auto]	ARRAY [2] OF BOOL	[2(FALSE)]	
VAR	CVar1	T0	TIMER	N/A	
VAR	MyFB0	N/A [Auto]	FB0	N/A	

6. Call the function block FB0 in the program organization unit (POU).

7. The program in the program organization unit (POU)

Network 1: When StartBit[0] is ON, the address of T511 is transmitted to TPoint1 in FB0.

When VarBit1 in the FB0 is ON, the instruction TMR is executed, and TPoint1 (T511) starts counting. When the value of TPoint1 matches the setting value, VarOut is ON.

Network 2: When StartBit[1] is ON, the address of CVar1[0] is transmitted to TPoint1 in FB0.

When VarBit1 in FB0 is ON, the instruction TMR is executed, and TPoint (CVar1) starts counting. When the value of TPoint1 matches the setting value, VarOut is ON.

4.7 Pointer Registers of 16-bit Counters

- ISPSoft supports the function blocks. If users want to use the 16-bit counter in the function block, they have to declare a pointer register of the 16-bit counter in the function block. The address of the 16-bit counter is transmitted to the pointer register of the 16-bit counter when the function block is called.
- When the variable declaration type is VAR_IN_OUT, and the data type is C_POINTE, the variable is the pointer register of the 16-bit counter. The value in the pointer register of the 16-bit counter can refer directly to the value stored in the device T or in the variable which is the counter in ISPSoft.
- Users can declare 8 pointer registers of the 16-bit counters in every function block. The range is CR0~CR7.
- If users want to use an instruction in the function block, and the counter is supported among the operands, users have to use the pointer register of the 16-bit counter.

Example:

1. Establish a program organization unit (POU) in ISPSoft first.
2. Establish a function block which is called FB0.

3. Declare the varaible in the function block FB0.

   Choose VAR_IN_OUT as the declaration type, CPoint1 as the identifier, C_POINTER as the data type. The variable is the pointer register of the 16-bit counter.

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
VAR	VarBit1	N/A [Auto]	BOOL	FALSE	
VAR_IN_OUT	CPoint1	N/A [Auto]	C_POINTER	N/A	

4. The program in the function block FB0

5. Declare the variable in the program organization unit (POU).

   The data type of CVar1 should be COUNTER.

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
VAR	StartBit	N/A [Auto]	ARRAY [2] OF BOOL	[2(FALSE)]	
VAR	CVar1	C1	COUNTER	N/A	
VAR	MyFB0	N/A [Auto]	FB0	N/A	

6. Call the function block FB0 in the program organization unit (POU).
7. The program in the program organization unit (POU)

   Network 1: When StartBit[0] is ON, the address of C0 is transmitted to CPoint1 in FB0.

When VarBit1 in FB0 is ON, CPoint1 (C0) is ON.

Network 2: When StartBit[1] is ON, the address of CVar1 is transmitted to CPoint1 in FB0.

When VarBit1 in FB0 is ON, CPoint1 (CVar1) is ON.

4.8 Pointer Registers of 32-bit Counters

- ISPSoft supports the function blocks. If users want to use the 32-bit counter in the function block, they have to declare a pointer register of the 32-bit counter in the function block. The address of the 32-bit counter is transmitted to the pointer register of the 32-bit counter when the function block is called.
- When the variable declaration type is VAR_IN_OUT, and the data type is HC_POINTER, the variable is the pointer register of the 32-bit counter. The value in the pointer register of the 32-bit counter can refer directly to the value stored in the device HC or in the variable which is the counter in ISPSoft.
- Users can declare 8 pointer registers of the 32-bit counters in every function block. The range is HCR0~HCR7.
- If users want to use an instruction in the function block, and the 32-bit counter is supported among the operands, users have to use the pointer register of the 32-bit counter.

Example:

1. Establish a function block which is called FB0.

2. Declare the varaible in the function block FB0.

   Choose VAR_IN_OUT as the declaration type, HCPoint1 as the identifier, HC_POINTER as the data type. The variable is the pointer register of the 32-bit counter.

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
VAR	VarBit1	N/A [Auto]	BOOL	FALSE	
VAR_IN_OUT	HCPoint1	N/A [Auto]	HC_POINTER	N/A	

3. The program in the function block FB0

4. Declare the variable in the program organization unit (POU).

   The data type of CVar1 should be COUNTER, and users have to fill in the address column with the practical address of the 32-bit counter.

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
VAR	StartBit	N/A [Auto]	ARRAY [2] OF BOOL	[2(FALSE)]	
VAR	CVar1	HC1	COUNTER	N/A	
VAR	MyFB0	N/A [Auto]	FB0	N/A	

5. Call the function block FB0 in the program organization unit (POU).

6. The program in the program organization unit (POU)

Network 1: When StartBit[0] is ON, the address of HC0 is transmitted to HCPoint1 in FB0.

When VarBit1 in FB0 is ON, HCPoint1 (HC0) is ON.

Network: When StartBit[1] is ON, the address of CVar1 is transmitted to HCPoint1 in FB0.

When VarBit1 in FB0 is ON, HCPoint1 (CVar1) is ON.

4.9 File Register

- AS series PLC provides users with File Registers for storing larger amount of parameters.
- Users can edit, upload, download the parameters in the file registers via ISPSoft.
- The values in FR can be read while operating the PLC. Please refer to API2303 MEMW in AS Series programming manual for more information about how to write in FR.

MEMO

5

Chapter 5 Basic Instructions

Table of Contents

5.1 List of Basic Instructions 5-2

5.2 Basic Instructions 5-3

5.1 List of Basic Instructions

Instruction code	Function	Operand	Operation time (μs)
LD/AND/OR	Loading contact A/Connecting contact A in series/Connecting contact A in parallel	DX, X, Y, M, SM, S, T, C, HC, D	0.025
LDI/ANI/ORI	Loading contact B/Connecting contact B in series/Connecting contact B in parallel	DX, X, Y, M, SM, S, T, C, HC, D	0.03
OUT	Driving the coil	DY, Y, M, SM, S, T, C, HC, D	0.04
SET	Keeping the device on	DY, Y, M, SM, S, T, C, HC, D	0.04
MC/MCR	Setting/Resetting the master control	N	0.24
LDP/ANDP/ORP	Starting the rising-edge detection/Connecting the rising-edge detection in series/Connecting the rising-edge detection in parallel	DX, X, Y, M, SM, S, T, C, HC, D	0.22
LDF/ANDF/ORF	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel	DX, X, Y, M, SM, S, T, C, HC, D	0.22
PED/APED/OPED	Starting the rising-edge detection/Connecting the rising edge-detection in series/Connecting the rising-edge detection in parallel	X, Y, M, SM, S, T, C, HC, D	0.22
NED/ANED/ONED	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel	X, Y, M, SM, S, T, C, HC, D	0.22
PLS	Rising-edge output	Y, M, SM, S	0.22
PLF	Falling-edge output	Y, M, SM, S	0.22
INV	Inverting the logical operation result	–	0.22
NP	The circuit is rising edge-triggered.	–	0.24
PN	The circuit is falling edge-triggered.	–	0.24
FB_NP	The circuit is rising edge-triggered.	Y, M, S, D	0.24
FB_PN	The circuit is falling edge-triggered.	Y, M, S, D	0.24

5.2 Basic Instructions

Instruction code	Operand	Function
LD/AND/OR	**S**	Loading contact A/Connecting contact A in series/Connecting contact A in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S	●		●	●	●	○	●	●	●	●	●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol:

LD

AND

OR

S : Specified device

Explanation:

1. The instruction LD applies to contact A which starts from the mother line or contact A which is the start of a contact circuit. It functions to reserve the current contents, and store the contact state which is acquired in the accumulative register.
2. The instruction AND is used to connect contact A in series. It functions to read the state of the contact which is specified to be connected in series, and perform the AND operation with the previous logical operation result. The final result is stored in the accumulative register.
3. The instruction OR is used to connect contact A in parallel. It functions to read the state of the contact which is specified to be connected in parallel, and perform the OR operation with the previous logical operation result. The final result is stored in the accumulative register.

Example:

1. Contact A of X0.0 is loaded, contact A of X0.1 is connected in series, contact A of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are ON, or when X0.2 is ON, Y0.0 is ON.

Instruction code	Operand	Function
LDI/ANI/ORI	**S**	Loading contact B/Connecting contact B in series/Connecting contact B in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S	●		●	●	●	○	●	●	●	●	●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol:

LDI

ANI

ORI

S : Specified device

Explanation:

1. The instruction LDI applies to contact B which starts from the mother line or contact B which is the start of a contact circuit. It functions to reserve the current contents, and store the contact state which is acquired in the accumulative register.
2. The instruction ANI is used to connect contact B in series. It functions to read the state of the contact which is specified to be connected in series, and perform the AND operation with the previous logical operation result. The final result is stored in the accumulative register.
3. The instruction ORI is used to connect contact B in parallel. It functions to read the state of the contact which is specified to be connected in parallel, and perform the OR operation with the previous logical operation result. The final result is stored in the accumulative register.

Example:

1. Contact B of X0.0 is loaded, contact B of X0.1 is connected in series, contact B of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are ON, or when X0.2 is ON, Y0.0 is ON.

Instruction code	Operand	Function
OUT	**D**	Driving the coil

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
D		●		●	●	○	●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												

Symbol:

D : Specified device

Explanation:

1. The logical operation result prior to the application of the instruction OUT is output into the specified device.
2. The action of the coil contact:

Operation result	OUT		
	Coil	Contact	
		Contact A (normally open)	Contact B (normally closed)
False	OFF	OFF	ON
True	ON	ON	OFF

Example:

1. Contact B of X0.0 is loaded, contact A of X0.1 is connected in series, and the coil Y0.0 is driven.
2. When X0.0 is OFF, and X0.1 is ON, Y0.0 is ON.

Instruction code	Operand	Function
SET	**D**	Keeping the device on

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
D		●		●	●	○	●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												

Symbol:

D : Specified device

Explanation:

When the instruction SET is driven, the specified device is set to ON. No matter the instruction SET is still driven, the specified device keeps ON. Users can set the specified device to OFF by means of the instruction RST.

Example:

1. Contact B of X0.0 is loaded, contact A of Y0.0 is connected in series, and Y0.1 keeps ON.
2. When X0.0 is OFF, and Y0.0 is ON, Y0.1 is ON. Even if the operation result changes, Y0.1 still keeps ON.

Instruction code	Operand	Function
MC/MCR	**N**	Setting/Resetting the master control

Symbol:

MC

MCR

N : Level of the nested program structure N0~N31

Explanation:

1. The instruction MCR is used to set the master control. When the instruction MC is executed, the instructions between MC and MCR are executed as usual. When the instruction MC is OFF, the actions of the instructions between MC and MCR are as follows.

Instruction type	Description
General-purpose timer	The timer value is reset to zero. The coil and the contact are OFF.
Timer used in the function block	The timer value is reset to zero. The coil and the contact are OFF.
Accumulative timer	The coil is OFF. The timer value and the state of the contact remains the same.
Counter	The coil is OFF. The timer value and the state of the contact remains the same.
Coils driven by OUT	All coils are OFF.
Devices driven by SET and RST	The states of the devices remain the same.
Applied instruction	All applied instructions are not executed. The FOR/NEXT loop is still repeated N times, but the actions of the instructions inside the FOR/NEXT loop follow those of the instructions between MC and MR.

2. The instruction MCR is used to reset the master control, and is placed at the end of the master control program. There should not be any contact instruction before MCR.

3. MC/MCR supports the nested program structure. There are at most 32 levels of nested program structures (N0~N31). Please refer to the example below.

Example:

NETWORK 1
X0.0
MC
En
N0
N
NETWORK 2
X0.1
Y0.0
NETWORK 3
X0.2
MC
En
N1
N
NETWORK 4
X0.3
Y0.1
NETWORK 5
MCR
N1
N
NETWORK 6
MCR
N0
N
NETWORK 7
X0.4
MC
En
N0
N
NETWORK 8
X0.5
Y0.2
NETWORK 9
MCR
N0
N

Instruction code	Operand	Function
LDP/ANDP/ORP	**S**	Starting the rising-edge detection/Connecting the rising-edge detection in series/Connecting the rising-edge detection in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S	●		●	●	●	○	●	●	●	●	●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol:

LDP

ANDP

ORP

S : Specified device

Explanation:

1. The instruction LDP functions to reserve the current contents, and store the rising-edge detection of the contact in the accumulative register.
2. The instruction ANDP is used to connect the rising-edge detection of the contact in series.
3. The instruction ORP is used to connect the rising-edge detection of the contact in parallel.
4. Only when LDP/ANDP/ORP is scanned can the state of the device be gotten, and not until LDP/ANDP/ORP is scanned next time can whether the state of the device changes be judged.
5. Please use the instructions PED, APED, and OPED in the subroutine.

Example:

1. The rising-edge detection of X0.0 starts, the rising-edge detection of X0.1 is connected in series, the rising-edge detection of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are switched from OFF to ON, or when X0.2 is switched from OFF to ON, Y0.0 is ON for a scan cycle.

Instruction code	Operand	Function
LDF/ANDF/ORF	**S**	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S	●		●	●	●	○	●	●	●	●	●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol:

LDF

ANDF

ORF

S : Specified device

_5

Explanation:

1. The instruction LDF functions to reserve the current contents, and store the falling-edge detection of the contact in the accumulative register.
2. The instruction ANDF is used to connect the falling-edge detection of the contact in series.
3. The instruction ORP is used to connect the falling-edge detection of the contact in parallel.
4. Only when LDF/ANDF/ORF is scanned can the state of the device be gotten, and not until LDF/ANDF/ORF is scanned next time can whether the state of the device changes be judged.
5. Please use the instructions NED, ANED, and ONED in the subroutine.

Example:

1. The falling-edge detection of X0.0 starts, the falling-edge detection of X0.1 is connected in series, the falling-edge detection of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are switched from OFF to ON, or when X0.2 is switched from OFF to ON, Y0.0 is ON for a scan cycle.

Instruction code	Operand	Function
PED/APED/OPED	S_1 , S_2	Starting the rising-edge detection/Connecting the rising edge-detection in series/Connecting the rising-edge detection in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S_1			●	●	●	○	●	●	●	●	●
S_2				●	●		●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol:

PED

APED

OPED

S_1 : Specified device

S_2 : For internal use

Q : Output the state of the operation result

Explanation:

1. PED/APED/OPED corresponds to LDP/ANDP/ORP. The only difference between PED/APED/OPED and LDP/ANDP/ORP lies in the fact that users need to specify the bit device S_2 in which the previous state of the contact is stored when PED/APED/OPED is executed. Please do not use the device S_2 repeatedly in the program. Otherwise, the wrong execution result will appear.
2. The instruction APED is used to connect the rising-edge detection of the contact in series.
3. The instruction OPED is used to connect the rising-edge detection of the contact in parallel.
4. Only when PED/APED/OPED is scanned can the state of the device be gotten, and not until PED/APED/OPED is scanned next time can whether the state of the device changes be judged.
5. PED/APED/OPED only can be used in the function block.
6. The state of the operation result will be outputted automatically after the instruction is executed. Users do not need to input device for this.

Example:

1. The rising-edge detection of X0.0 starts, the rising-edge detection of X0.1 is connected in series, the rising-edge detection of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are switched from OFF to ON, or when X0.2 is switched from OFF to ON, Y0.0 is ON for a scan cycle.

Network 1
PED
Q
X0.0
S1
M0
S2
PED
Q
X0.1
S1
M1
S2
Y0.0
PED
Q
X0.2
S1
M2
S2

Instruction code	Operand	Function
NED/ANED/ONED	S_1 · S_2	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S_1			●	●	●	○	●	●	●	●	●
S_2				●	●		●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol:

NED

ANED

ONED

S_1 : Specified device

S_2 : For internal use

Q : Output the state of the operation result

Explanation:

1. NED/ANED/ONED corresponds to LDF/ANDF/ORF. The only difference between NED/ANED/ONED and LDF/ANDF/ORF lies in the fact that users need to specify the bit device S_2 in which the previous state of the contact is stored when NED/ANED/ONED is executed. Please do not use the device S_2 repeatedly in the program. Otherwise, the wrong execution result will appear.
2. The instruction ANED is used to connect the falling-edge detection of the contact in series.
3. The instruction ONED is used to connect the falling-edge detection of the contact in parallel.
4. Only when NED/ANED/ONED is scanned can the state of the device be gotten, and not until NED/ANED/ONED is scanned next time can whether the state of the device changes be judged.
5. NED/ANED/ONED only can be used in the function block.
6. The state of the operation result will be outputted automatically after the instruction is executed. Users do not need to input device for this.

Example:

1. The falling -edge detection of X0.0 starts, the falling -edge detection of X0.1 is connected in series, the falling -edge detection of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are switched from OFF to ON, or when X0.2 is switched from OFF to ON, Y0.0 is ON for a scan cycle.

Network 1
NED
Q
X0.0
S1
M0
S2
NED
Q
X0.1
S1
M1
S2
Y0.0
NED
Q
X0.2
S1
M2
S2

Instruction code	Operand	Function
PLS	**D**	Rising-edge output

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
D				●	●	○	●				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												

Symbol:

PLS | **D** : Specified device

Explanation:

1. When the conditional contact is switched from OFF to ON, the instruction PLS is executed, and the device D sends out a pulse for a scan cycle.
2. Please do not use the instruction PLS in the function block.

Example:

When X0.0 is ON, M0 is ON for a pulse time. When M0 is ON, Y0.0 is set to ON.

Timing diagram:

Instruction code	Operand	Function
PLF	**D**	Falling-edge output

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
D				●	●	○	●				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol:

PLF | **D** :Specified device

Explanation:

1. When the conditional contact is switched from ON to OFF, the instruction PLF is executed, and the device D sends out a pulse for a scan cycle.
2. Please do not use the instruction PLS in the function block.

Example:

When X0.0 is ON, M0 is ON for a pulse time. When M0 is ON, Y0.0 is set to ON.

Timing chart:

Instruction code	Operand	Function
INV	-	Inverting the logical operation result

Symbol:

Explanation:

The logical operation result preceding the instruction INV is inverted, and the inversion result stored in the accumulative register.

Example:

When X0.0 is ON, Y0.0 is OFF. When X0.0 is OFF, Y0.0 is ON.

Instruction code	Operand	Function
NP	-	The circuit is rising edge-triggered.

Symbol:

Explanation:

1. When the value in the accumulative register turns from 0 to 1, the instruction NP keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. Please use the instruction FB_NP in the function block.

Example:

Instruction:		Operation:
LD	M0	Contact A of M0 is loaded.
AND	M1	Contact A of M1 is connected in series.
NP		The circuit is rising edge-triggered.
OUT	Y0.0	The coil Y0.0 is driven.

Timing diagram:

Instruction code	Operand	Function
PN	-	The circuit is falling edge-triggered.

Symbol:

Explanation:

1. When the value in the accumulative register turns from 1 to 0, the instruction PN keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. Please use the instruction FB_ PN in the function block.

Example:

Instruction:		**Operation: :**
LD	M0	Contact A of M0 is loaded.
AND	M1	Contact A of M1 is connected in series.
PN		The circuit is falling edge-triggered.
OUT	Y0.0	The coil Y0.0 is driven.

Timing diagram:

Instruction code	Operand	Function
FB_NP	**S**	The circuit is rising edge-triggered.

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S				●	●		●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol:

S ：For internal use

Explanation:

1. When the value in the accumulative register turns from 0 to 1, the instruction FB_NP keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. The previous state of the contact is stored in the bit device **S**. Please do not use **S** repeatedly in the program. Otherwise, the wrong execution result will appear.
3. The instruction FB_NP only can be used in the function block.

Example:

Instruction:		**Operation:**
LD	M0	Contact A of M0 is loaded.
AND	M1	Contact A of M1 is connected in series.
FB_NP	D0.0	The circuit is rising edge-triggered.
OUT	Y0.0	The coil Y0.0 is driven.

Timing diagram:

Instruction code	Operand	Function
FB_PN	**S**	The circuit is falling edge-triggered.

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S				●	●		●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol:

S ：For internal use

Explanation:

1. When the value in the accumulative register turns from 1 to 0, the instruction FB_PN keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. The previous state of the contact is stored in the bit device **S**. Please do not use **S** repeatedly in the program. Otherwise, the wrong execution result will appear.
3. The instruction FB_PN only can be used in the function block.

Example:

Instruction:		**Operation:**
LD	M0	Contact A of M0 is loaded.
AND	M1	Contact A of M1 is connected in series.
FB_PN	D0.0	The circuit is falling edge-triggered.
OUT	Y0.0	The coil Y0.0 is driven.

Timing diagram:

MEMO

6

Chapter 6 Applied Instructions

Table of Contents

6.1 Comparison Instructions 6-4
6.1.1 List of Comparison Instructions 6-4
6.1.2 Explanation of Comparison Instructions 6-7

6.2 Arithmetic Instructions 6-46
6.2.1 List of Arithmetic Instructions 6-46
6.2.2 Explanation of Arithmetic Instructions 6-47

6.3 Data Conversion Instructions 6-78
6.3.1 List of Data Conversion Instructions 6-78
6.3.2 Explanation of Data Conversion Instructions 6-79

6.4 Data Transfer Instructions 6-117
6.4.1 List of Data Transfer Instructions 6-117
6.4.2 Explanation of Data Transfer Instructions 6-118

6.5 Jump Instructions 6-145
6.5.1 List of Jump Instructions 6-145
6.5.2 Explanation of Jump Instructions 6-146

6.6 Program Execution Instructions 6-154
6.6.1 List of Program Execution Instructions 6-154
6.6.2 Explanation of Program Execution Instructions 6-155

6.7 IO Refreshing Instructions 6-167
6.7.1 List of IO Refreshing Instructions 6-167
6.7.2 Explanation of IO Refreshing Instructions 6-168

6.8 Convenience Instructions 6-170
6.8.1 List of Convenience Instructions 6-170
6.8.2 Explanation of Convenience Instructions 6-171

6.9 Logic Instructions 6-218
6.9.1 List of Logic Instructions 6-218
6.9.2 Explanation of Logic Instructions 6-219

6.10 Rotation Instructions 6-240
6.10.1 List of Rotation Instructions 6-240
6.10.2 Explanation of Rotation Instructions 6-241

6.11 Timer and Counter Instructions 6-252
6.11.1 List of Timer and Counter Instructions 6-252
6.11.2 Explanation of Timer and Counter Instructions 6-253

6.12 Shift Instructions 6-285
6.12.1 List of Shift Instructions 6-285
6.12.2 Explanation of Shift Instructions 6-286

6.13 Data Processing Instructions 6-323
6.13.1 List of Data Processing Instructions 6-323
6.13.2 Explanation of Data Processing Instructions 6-324

6.14 Structure Creation Instructions 6-377
6.14.1 List of Structure Creation Instructions 6-377
6.14.2 Explanation of Structure Creation Instructions 6-378

6.15 Module Instructions 6-386
6.15.1 List of Module Instructions 6-386
6.15.2 Explanation of Module Instructions 6-387

6.16 Floating-point Number Instructions 6-393
6.16.1 List of Floating-point Number Instructions 6-393
6.16.2 Explanation of Floating-point Number Instructions 6-394

6.17 Real-time Clock Instructions 6-429
6.17.1 List of Real-time Clock Instructions 6-429
6.17.2 Explanation of Real-time Clock Instructions 6-430

6.18 Peripheral Instructions 6-453
6.18.1 List of Peripheral Instructions 6-453
6.18.2 Explanation of Peripheral Instructions 6-454

6.19 Communication Instructions 6-470
6.19.1 List of Communication Instructions 6-470
6.19.2 Explanation of Communication Instructions 6-471
6.19.3 Descriptions on the Communication-related Flags and Registers 6-539

6.20 Other Instructions 6-542
6.20.1 List of Other Instructions 6-542
6.20.2 Explanation of Other Instructions 6-543

6.21 String Processing Instructions 6-553
6.21.1 List of String Processing Instructions 6-553
6.21.2 Explanation of String Processing Instructions 6-554

6.22 Ethernet Instructions 6-608
6.22.1 List of Ethernet Instructions 6-608
6.22.2 Explanation of Ethernet Instructions 6-609

6.23 Memory Card Instructions 6-643

6.23.1 List of Memory Card Instructions 6-643
6.23.2 Explanation of Memory Card Instructions 6-644

6.24 Task Control Instructions 6-660
6.24.1 List of Task Control Instructions 6-660
6.24.2 Explanation of Task Control Instructions 6-661

6.25 SFC Instructions 6-665
6.25.1 List of SFC Instructions 6-665
6.25.2 Explanation of SFC Instructions 6-666

6.26 High-speed Output Instructions 6-673
6.26.1 List of High-speed Output Instructions 6-673
6.26.2 Explanation of High-speed Output Instructions 6-675

6.27 Delta CANopen Communication Instructions 6-760
6.27.1 List of Delta CANopen Communication Instructions 6-760
6.27.2 Explanation of Delta CANopen Communication Instructions 6-761

6.1 Comparison Instructions

6.1.1 List of Comparison Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0000	LD=	DLD=	–	S1 = S2
0001	LD<>	DLD<>	–	S1≠S2
0002	LD>	DLD>	–	S1 > S2
0003	LD>=	DLD>=	–	S1≧S2
0004	LD<	DLD<	–	S1 < S2
0005	LD<=	DLD<=	–	S1≦S2
0006	AND=	DAND=	–	S1 = S2
0007	AND<>	DAND<>	–	S1≠S2
0008	AND>	DAND>	–	S1 > S2
0009	AND>=	DAND>=	–	S1≧S2
0010	AND<	DAND<	–	S1 < S2
0011	AND<=	DAND<=	–	S1≦S2
0012	OR=	DOR=	–	S1 = S2
0013	OR<>	DOR<>	–	S1 ≠ S2
0014	OR>	DOR>	–	S1 > S2
0015	OR>=	DOR>=	–	S1≧S2
0016	OR<	DOR<	–	S1 < S2
0017	OR<=	DOR<=	–	S1≦S2
0018	–	FLD=	–	S1 = S2
0019	–	FLD<>	–	S1≠S2
0020	–	FLD>	–	S1 > S2
0021	–	FLD>=	–	S1≧S2
0022	–	FLD<	–	S1 < S2
0023	–	FLD<=	–	S1≦S2
0024	–	FAND=	–	S1 = S2
0025	–	FAND<>	–	S1≠S2
0026	–	FAND>	–	S1 > S2
0027	–	FAND>=	–	S1≧S2
0028	–	FAND<	–	S1 < S2
0029	–	FAND<=	–	S1≦S2
0030	–	FOR=	–	S1 = S2
0031	–	FOR<>	–	S1≠S2
0032	–	FOR>	–	S1 > S2

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0033	–	FOR>=	–	S1≧S2
0034	–	FOR<	–	S1 < S2
0035	–	FOR<=	–	S1≦S2
0036	LD$=	–	–	S1 = S2
0037	LD$<>	–	–	S1≠S2
0042	AND$=	–	–	S1 = S2
0043	AND$<>	–	–	S1≠S2
0048	OR$=	–	–	S1 = S2
0049	OR$<>	–	–	S1≠S2
0054	CMP	DCMP	✓	Comparing the values
0055	ZCP	DZCP	✓	Zone comparison
0056	–	FCMP	✓	Comparing the floating-point numbers
0057	–	FZCP	✓	Floating-point zone comparison
0058	MCMP	–	✓	Matrix comparison
0059	CMPT=	–	✓	Comparing the tables ON: =
0060	CMPT<>	–	✓	Comparing the tables ON: <>
0061	CMPT>	–	✓	Comparing the tables ON: >
0062	CMPT>=	–	✓	Comparing the tables ON: ≧
0063	CMPT<	–	✓	Comparing the tables ON: <
0064	CMPT<=	–	✓	Comparing the tables ON: ≦
0065	CHKADR	–	–	Checking the address of the contact type of pointer register
0066	LDZ=	DLDZ=	–	|S1-S2|=|S3|
0067	LDZ<>	DLDZ<>	–	|S1-S2|≠|S3|
0068	LDZ>	DLDZ>	–	|S1-S2| > |S3|
0069	LDZ>=	DLDZ>=	–	|S1-S2|≧|S3|
0070	LDZ<	DLDZ<	–	|S1-S2| < |S3|
0071	LDZ<=	DLDZ<=	–	|S1-S2|≦|S3|
0072	ANDZ=	DANDZ=	–	|S1-S2|=|S3|
0073	ANDZ<>	DANDZ<>	–	|S1-S2|≠|S3|
0074	ANDZ>	DANDZ>	–	|S1-S2| > |S3|
0075	ANDZ>=	DANDZ>=	–	|S1-S2|≧|S3|
0076	ANDZ<	DANDZ<	–	|S1-S2| < |S3|
0077	ANDZ<=	DANDZ<=	–	|S1-S2|≦|S3|

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0078	ORZ=	DORZ=	–	|S1-S2|=|S3|
0079	ORZ<>	DORZ<>	–	|S1-S2|≠|S3|
0080	ORZ>	DORZ>	–	|S1-S2| > |S3|
0081	ORZ>=	DORZ>=	–	|S1-S2|≧|S3|
0082	ORZ<	DORZ<	–	|S1-S2| < |S3|
0083	ORZ<=	DORZ<=	–	|S1-S2|≦|S3|

6.1.2 Explanation of Comparison Instructions

API	Instruction code			Operand	Function
0000~ 0005	D	LD※		S_1 · S_2	Comparing the values LD※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol:

Taking LD= and DLD= for example

S_1 : Data source 1

S_2 : Data source 1

Explanation:

1. The instructions are used to compare the value in S_1 with that in S_2. Take the instruction LD= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

2. Only the 32-bit instruction can use the 32-bit counter, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0000	LD =	**D**LD =	$S_1 = S_2$	$S_1 \neq S_2$
0001	LD < >	**D**LD < >	$S_1 \neq S_2$	$S_1 = S_2$
0002	LD >	**D**LD >	$S_1 > S_2$	$S_1 \leq S_2$
0003	LD > =	**D**LD > =	$S_1 \geq S_2$	$S_1 < S_2$
0004	LD <	**D**LD <	$S_1 < S_2$	$S_1 \geq S_2$
0005	LD < =	**D**LD < =	$S_1 \leq S_2$	$S_1 > S_2$

Example:

1. When the value in C10 is equal to 200, Y0.10 is ON.
2. When the value in D200 is greater than -30, Y0.11 keeps ON.
3. When the value in (C201, C200) is less than 678,493, or when M3 is ON, M50 is ON.

NETWORK 1
=
Q
200
S1
C10
S2
Y0.10
NETWORK 2
>
Q
D200
S1
-30
S2
Y0.11
S
NETWORK 3
D>
Q
678493
S1
C200
S2
M50
M3

API	Instruction code			Operand	Function
0006~ 0011	D	AND※		S_1 , S_2	Comparing the values AND※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol:

Taking AND= and DAND= for example

S_1 : Data source 1

S_2 : Data source 2

Explanation:

1. The instructions are used to compare the value in S_1 with that in S_2. Take the instruction AND= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

2. Only the 32-bit instruction can use the 32-bit counter, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0006	AND =	**D**AND =	$S_1 = S_2$	$S_1 \neq S_2$
0007	AND < >	**D**AND < >	$S_1 \neq S_2$	$S_1 = S_2$
0008	AND >	**D**AND >	$S_1 > S_2$	$S_1 \leqq S_2$
0009	AND > =	**D**AND > =	$S_1 \geqq S_2$	$S_1 < S_2$
0010	AND <	**D**AND <	$S_1 < S_2$	$S_1 \geqq S_2$
0011	AND < =	**D**AND < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example:

1. When X0.0 is ON and the current value in C10 is equal to 100, Y0.10 is ON.

2. When X0.1 is OFF and the value in D0 is not equal to -10, Y0.11 keeps ON.

3. When X0.2 is ON and the value in (D11, D10) is less than 678,493, or when M3 is ON, M50 is ON.

NETWORK 1
X0.0
=
Q
Y0.10
100
S1
C10
S2
NETWORK 2
X0.1
<>
Q
Y0.11
-10
S1
D0
S2
NETWORK 3
X0.2
D>
Q
M50
678493
S1
D10
S2
M3

API	Instruction code			Operand	Function
0012~0017	D	OR※		S_1 · S_2	Comparing the values OR※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse Instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol:

Taking OR= and DOR= for example

S_1 : Data source 1

S_2 : Data source 2

Explanation:

1. The instructions are used to compare the value in S_1 with that in S_2. Take the instruction OR= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

2. Only the 32-bit instruction can use the 32-bit counter, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0012	OR =	**D**OR =	$S_1 = S_2$	$S_1 \neq S_2$
0013	OR < >	**D**OR < >	$S_1 \neq S_2$	$S_1 = S_2$
0014	OR >	**D**OR >	$S_1 > S_2$	$S_1 \leq S_2$
0015	OR > =	**D**OR > =	$S_1 \geq S_2$	$S_1 < S_2$
0016	OR <	**D**OR <	$S_1 < S_2$	$S_1 \geq S_2$
0017	OR < =	**D**OR < =	$S_1 \leq S_2$	$S_1 > S_2$

Example:

1. When X0.1 is ON, or when the current value in C10 is equal to 100, Y0.10 is ON.

2. When both X0.2 and M30 are ON, or when the value in (D101, D100) is greater than or equal to 1000,000, M60 is ON.

NETWORK 1
X0.1
Y0.10
=
Q
100
S1
C10
S2
NETWORK 2
X0.2
M30
M60
D>=
Q
D100
S1
1000000
S2

API	Instruction code			Operand	Function
0018~ 0023		FLD※		S_1 · S_2	Comparing the floating-point numbers LD※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○					○
S_2	●	●			●	●	●	●	●		○					○

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				

Pulse Instruction	16-bit instruction	32-bit instruction
-	-	AS

Symbol:

Taking FLD= and DFLD= for example

S_1 : Data source 1

S_2 : Data source 2

Explanation:

1. This instruction is a 32-bit single precision floating point comparison instruction.
2. The instructions are used to compare the value in S_1 with that in S_2, and the values compared are floating-point numbers. Take the instruction FLD= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

API number	32-bit instruction	Continuity condition	Discontinuity condition
0018	FLD =	$S_1 = S_2$	$S_1 \neq S_2$
0019	FLD < >	$S_1 \neq S_2$	$S_1 = S_2$
0020	FLD >	$S_1 > S_2$	$S_1 \leqq S_2$
0021	FLD > =	$S_1 \geqq S_2$	$S_1 < S_2$
0022	FLD <	$S_1 < S_2$	$S_1 \geqq S_2$
0023	FLD < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example:

Take the instruction FLD = for example. When the value in D0 is equal to that in D2, Y0.0 is ON.

Additional remark:

1. If the value in $\mathbf{S_1}$ or $\mathbf{S_2}$ exceeds the range of values which can be represented by the floating-point numbers, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.

API	Instruction code	Operand	Function
0024~0029	FAND※	S_1 · S_2	Comparing the floating-point numbers AND※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○					○
S_2	●	●			●	●	●	●	●		○					○

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				

Pulse Instruction	16-bit instruction	32-bit instruction
-	-	AS

Symbol:

Taking FAND= and DFAND= for example

S_1 : Data source 1

S_2 : Data source 2

Explanation:

1. This instruction is a 32-bit single precision floating point comparison instruction.
2. The instructions are used to compare the value in S_1 with that in S_2, and the values compared are floating-point numbers. Take the instruction FAND= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

API number	32-bit instruction	Continuity condition	Discontinuity condition
0024	FAND =	$S_1 = S_2$	$S_1 \neq S_2$
0025	FAND < >	$S_1 \neq S_2$	$S_1 = S_2$
0026	FAND >	$S_1 > S_2$	$S_1 \leqq S_2$
0027	FAND > =	$S_1 \geqq S_2$	$S_1 < S_2$
0028	FAND <	$S_1 < S_2$	$S_1 \geqq S_2$
0029	FAND < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example:

Take the instruction FAND = for example. When X1.0 is ON and the value in D1 is equal to that in D2, Y1.0 is ON.

Additional remark:

1. If the value in $\mathbf{S_1}$ or $\mathbf{S_2}$ exceeds the range of values which can be represented by the floating-point numbers, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0030~ 0035		FOR※		S_1 · S_2	Comparing the floating-point numbers OR※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○					○
S_2	●	●			●	●	●	●	●		○					○

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				

Pulse Instruction	16-bit instruction	32-bit instruction
-	-	AS

Symbol:

S_1 : Data source1

S_2 : Data source2

Taking FOR= and DFOR= for example

Explanation:

1. This instruction is a 32-bit single precision floating point comparison instruction.

2. The instructions are used to compare the value in S_1 with that in S_2, and the values compared are floating-point numbers. Take the instruction FOR= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

API number	32-bit instruction	Continuity condition	Discontinuity condition
0030	FOR =	$S_1 = S_2$	$S_1 \neq S_2$
0031	FOR < >	$S_1 \neq S_2$	$S_1 = S_2$
0032	FOR >	$S_1 > S_2$	$S_1 \leqq S_2$
0033	FOR > =	$S_1 \geqq S_2$	$S_1 < S_2$
0034	FOR <	$S_1 < S_2$	$S_1 \geqq S_2$
0035	FOR < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example:

When X1.0 is ON, or when the value in D1 is equal to that in D2, Y1.0 is ON.

Additional remark:

1. If the value in $\mathbf{S_1}$ or $\mathbf{S_2}$ exceeds the range of values which can be represented by the floating-point numbers, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0036~ 0037		LD$※		S_1 · S_2	Comparing the strings LD$※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●						○	
S_2	●	●			●	●		●	●						○	

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													●
S_2													●

Pulse Instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

Taking LD$= for example

S_1 : Data source1

S_2 : Data source2

Explanation:

1. The instructions are used to compare the data in S_1 with that in S_2, and the data compared is strings.
2. Up to 256 characters can be inputted in the S_1 and S_2 (16#00 the end symbol is included.)
3. Take the instruction LD$= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the **continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2**, the discontinuity condition of the instruction is met.

API number	16-bit instruction	Continuity condition	Discontinuity condition
0036	LD$ =	$S_1 = S_2$	$S_1 \neq S_2$
0037	LD$ < >	$S_1 \neq S_2$	$S_1 = S_2$

4. Only when the data in **S**~**S**+n (n indicates the nth device, up to 256 characters can be inputted) includes 16#00 can the data be judged as a complete string. For example:

	b15 ~ b8	b7 ~ b0
S	16#32(2)	16#31(1)
S+1	16#34(4)	16#33(3)
S+2	16#00	16#35(5)

" 12345"

5. When two strings are the same, the corresponding comparison operation results of the instructions are listed below. For example:

	b15 ~ b8	b7 ~ b0
S1	16#42(B)	16#41(A)
S1 +1	16#44(D)	16#43(C)
S1 +2	16#00	16#45(E)

" ABCDE"

Comparison sign

	b15 ~ b8	b7 ~ b0
S2	16#42(B)	16#41(A)
S2 +1	16#44(D)	16#43(C)
S2 +2	16#00	16#45(E)

" ABCDE"

Comparison symbol	Comparison operation result
$ =	Continuity
$ < >	Discontinuity

Example:

When the string starting with the data in D0~16#00 is equal to the string staring with D2~16#00, Y0.0 is ON.

Additional remark:

1. If the string contains more than 256 characters or the string does not end with 16#00, the instruction will not be executed, SM is ON, and the error code in SR0 is 16#200E.
2. During the string comparison, whenever the end symbol 16#00 is shown, the length of the string ends there.

API	Instruction code			Operand	Function
0042~ 0043		AND$※		S_1 · S_2	Comparing the strings AND$※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●						○	
S_2	●	●			●	●		●	●						○	

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													●
S_2													●

Pulse Instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

Taking AND$= for example

S_1 : Data source1

S_2 : Data source2

Explanation:

1. The instructions are used to compare the data in S_1 with that in S_2, and the data compared is strings.
2. Up to 256 characters can be inputted in the S_1 and S_2 (16#00 the end symbol is included.)
3. Take the instruction AND$= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the **continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2**, the discontinuity condition of the instruction is met.

API number	16-bit instruction	Continuity condition	Discontinuity condition
0042	AND$ =	$S_1 = S_2$	$S_1 \neq S_2$
0043	AND$ < >	$S_1 \neq S_2$	$S_1 = S_2$

4. Only when the data in **S**~**S**+n (n indicates the nth device, up to 256 characters can be inputted) includes 16#00 can the data be judged as a complete string. For example:
5. When two strings are the same, the corresponding comparison operation results of the instructions are listed below. For example:

Comparison symbol	Comparison operation result
$ =	Continuity
$ < >	Discontinuity

Example:

When the string starting with the data in D0~16#00 is equal to the string staring with D2~16#00, Y0.0 is ON.

Additional remark:

1. If the string contains more than 256 characters or the string does not end with 16#00, the instruction will not be executed, SM is ON, and the error code in SR0 is 16#200E.
2. During the string comparison, whenever the end symbol 16#00 is shown, the length of the string ends there.

API	Instruction code			Operand	Function
0048~ 0049		OR$※		S_1 · S_2	Comparing the strings OR$※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●						○	
S_2	●	●			●	●		●	●						○	

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													●
S_2													●

Pulse Instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol：

Taking OR$= for example

S_1 : Data source1

S_2 : Data source2

Explanation:

1. The instructions are used to compare the data in S_1 with that in S_2, and the data compared is strings.
2. Up to 256 characters can be inputted in the S_1 and S_2 (16#00 the end symbol is included.)
3. Take the instruction OR$= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the **continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2**, the discontinuity condition of the instruction is met.

API number	16-bit instruction	Continuity condition	Discontinuity condition
0048	OR$ =	$S_1 = S_2$	$S_1 \neq S_2$
0049	OR$ < >	$S_1 \neq S_2$	$S_1 = S_2$

4. Only when the data in **S**~**S**+n (n indicates the nth device, up to 256 characters can be inputted) includes 16#00 can the data be judged as a complete string. For example:
5. When two strings are the same, the corresponding comparison operation results of the instructions are listed below. For example:

Comparison symbol	Comparison operation result
$ =	Continuity
$ < >	Discontinuity

Example:

When the string starting with the data in D0~16#00 is equal to the string staring with D2~16#00, Y0.0 is ON.

Additional remark:

1. If the string contains more than 256 characters or the string does not end with 16#00, the instruction will not be executed, SM is ON, and the error code in SR0 is 16#200E.
2. During the string comparison, whenever the end symbol 16#00 is shown, the length of the string ends there.

API	Instruction code			Operand	Function
0054	D	CMP	P	S_1 · S_2 · **D**	Comparing the values

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol：

S1 : Comparison value1

S2 : Comparison value2

D : Comparison result

Explanation：

1. The instruction is used to compare the value in S_1 with that in S_2, and the values compared are singed decimal numbers. The comparison results are stored in **D**.
2. The operand **D** occupies 3 consecutive devices. The comparison results are stored in **D**, **D**+1, and **D**+2. If the comparison value in S_1 is greater than the comparison value in S_2, **D** will be ON. If the comparison value in S_1 is equal to the comparison value in S_2, **D**+1 is ON. If the comparison value in S_1 is less than the comparison value in S_2, **D**+2 will be ON.
3. Only the instructions **D**CMP and **D**CMPP can use the 32-bit counter, but not the device E.

Example:

1. If the operand **D** is M0, the comparison results will be stored in M0, M1 and M2, as shown below.
2. When X0.0 is ON, the instruction CMP is executed. M0, M1, or M2 is ON. When X0.0 is OFF, the execution of the instruction CMP stops. The state of M0, the state of M1, and the state of M1 remain unchanged.

3. If users need to clear the comparison result, they can use the instruction RST or ZRST.

Additional remark:

1. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [3] of BOOL.
2. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0055	D	ZCP	P	**S_1 · S_2 · S · D**	Zone comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
S	●	●			●	●	●	●	●		○	○	○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
S		●	●		●	●	●				●	●	
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol：

S1 : Minimum value of the zone comparison

S2 : Maximum value of the zone comparison

S : Comparison value

D : Comparison result

Explanation：

1. The instruction is used to compare the value in **S** with that in **S_1**, and compare the value in **S** with that in **S_2**. The values compared are singed decimal numbers, and the comparison results are stored in **D**.
2. The value in **S_1** must be less than that in **S_2**. If the value in **S_1** is larger than that in **S_2**, **S_1** will be taken as the maximum/minimum value during the execution of the instruction ZCP.
3. The operand **D** occupies three consecutive devices. The comparison results are stored in **D**, **D**+1, and **D**+2. If the comparison value in **S_1** is less than the comparison value in **S**, **D** will be ON. If the comparison value in **S** is within the range between the value in **S_1** and the value in **S_2**, **D**+1 will ON. If the comparison value in **S** is greater than the value in **S_2**, **D**+2 will be ON.
4. Only the instructions **D**ZCP and **D**ZCPP can use the 32-bit counter, but not the device E.

Example:

1. If the operand **D** is M0, the comparison results will be stored in M0, M1 and M2, as shown below.
2. When X0.0 is ON, the instruction ZCP is executed. M0, M1, or M2 is ON. When X0.0 is OFF, the instruction ZCP is not executed. The state of M0, the state of M1, and the state of M2 remain the same as those before X0.0's being OFF.

3. If users need to clear the comparison result, they can use the instruction RST or ZRST.

Additional remark:

1. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [3] of BOOL.
2. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0056		FCMP	P	S_1 · S_2 · **D**	Comparing the floating-point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○					○
S_2	●	●			●	●	●	●	●		○					○
D		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol：

S_1 ： Floating-point comparison value1

S_2 ： Floating-point comparison value2

D ： Comparison result

Explanation：

1. The instruction FCMP is used to compare the floating-point number in S_1 with the floating-point number in S_2. The comparison results (> · = · <) are stored in **D**.

2. The operand **D** occupies three consecutive devices. The comparison results are stored in **D**, **D**+1, and **D**+2. If the comparison value in S_1 is greater than the comparison value in S_1, **D** will be ON. If the comparison value in S_1 is equal to the value in S_2, **D**+1 will ON. If the comparison value in S_1 is less than the value in S_2, **D**+2 will be ON.

Example:

1. If the operand **D** is M10, the comparison results will be stored in M10, M11 and M12, as shown below.

2. When X0.0 is ON, the instruction FCMP is executed. M10, M11, or M12 is ON. When X0.0 is OFF, the instruction FCMP is not executed. The state of M10, the state of M11, and the state of M12 remain the same as those before X0.0's being OFF.

3. If users want to get the comparison result ≧, ≦, or ≠, they can connect M10~M12 is series or in parallel.

4. If users want to clear the comparison result, they can use the instruction RST or ZRST.

Additional remark:

1. If the value in **S_1** or **S_2** exceeds the range of values which can be represented by the floating-point numbers, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.
2. If users declare the operand **D** in ISPSoft, the data type will ARRAY [3] of BOOL.
3. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0057		FZCP	P	S_1 · S_2 · **S** · **D**	Floating-point zone comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○					○
S_2	●	●			●	●	●	●	●		○					○
S	●	●			●	●	●	●	●		○					○
D		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
S									●				
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol：

S_1 : Minimum value of the zone comparison

S_2 : Maximum value of the zone comparison

S : Comparison value

D : Comparison result

Explanation：

1. The instruction is used to compare the value in **S** with that in S_1, and compare the value in **S** with that in S_2. The values compared are floating-point numbers, and the comparison results are stored in **D**.
2. The value in S_1 must be less than that in S_2. If the value in S_1 is larger than that in S_2, S_1 will be taken as the maximum/minimum value during the execution of the instruction FZCP.
3. The operand **D** occupies three consecutive devices. The comparison results are stored in **D**, **D**+1, and **D**+2. If the comparison value in S_1 is greater than the comparison value in **S**, **D** will be ON. If the comparison value in **S** is within the range between the value in S_1 and the value in S_2, **D**+1 will be ON. If the compared value in S_2 is less than the value in **S**, **D**+2 will be ON.

Example:

1. If the operand **D** is M0, the comparison results will be stored in M0, M1 and M2.
2. When X0.0 is ON, the instruction FZCP is executed. M0, M1, or M2 is ON. When X0.0 is OFF, the instruction FZCP is not executed. The state of M0, the state of M1, and the state of M2 remain the same as those before X0.0's being OFF.
3. If users want to clear the comparison result, they can use the instruction RST or ZRST.

Additional remark:

1. If the value in $\mathbf{S_1}$ or $\mathbf{S_2}$ or **S** exceeds the range of values which can be represented by the floating-point numbers, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.
2. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [3] of BOOL.
3. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0058		MCMP	P	S_1 · S_2 · **n** · **D**	Matrix comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●		○					
S_2	●	●			●	●		●	●		○					
n	●	●			●	●		●	●		○		○	○		
D		●			●	●		●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol：

S_1 : Matrix source device 1

S_2 : Matrix source device 2

n : Length of the array

D : Pointer

Explanation：

1. The search for the bits whose states are different starts from the bits specified by the number gotten from the addition of one to the current value in **D**. After the bits whose states are different are found, the bit number is stored in **D**, and the comparison is finished.
2. The operand **n** should be within the range between 1 and 256.
3. When SM607 is ON, the equivalent values are compared. When SM607 is OFF, the different values are compared. When the matching bits are compared, the comparison stops immediately, and SM610 is ON. When the last bits are compared, SM608 is ON, and the bit number is stored in **D**. The comparison starts from the 0th bits in the next scan cycle, and SM609 is ON. When the value in **D** exceeds the range, SM611 is ON.
4. When the instruction MCMP is executed, users need a 16-bit register to specify a certain bit among the 16**n** bits in the matrix for the operation. The register is called the pointer, and is specified by users. The value in the register is within the range between 0 and 16**n**-1, and corresponds to the bit within the range between b0 and b16**n**-1. During the operation, users should be prevented from altering the value of the pointer in case the search for the matching bits is affected. If the value of the pointer exceeds the range, SM611 will be ON, and the instruction MCMP will not be executed.
5. If SM608 and SM610 occur simultaneously, they will be ON simultaneously.

Example:

1. When X0.0 is switched from OFF to ON, SM609 is OFF. The search for the bits whose states are different (SM607 is

OFF) starts from the bits specified by the number gotten from the addition of one to the current value of the pointer.

2. Suppose the current value in D20 is 2. When X0.0 is switched from OFF to ON four times, users can get the following execution results.

- The value in D20 is 5, SM610 is ON, and SM608 is OFF.
- The value in D20 is 45, SM610 is ON, and SM608 is OFF.
- The value in D20 is 47, SM610 is OFF, and SM608 is ON.
- The value in D20 is 1, SM610 is ON, and SM608 is OFF.

Additional remark:

1. The description of the operation error code:

   If the devices **S_1+n**-1 and **S_2+n**-1 exceed the range, the instruction MCMP is not executed, SM is ON, and the error code in SR0 is 16#2003.

   If the value in the operand **n** is not within the range between 1 and 256, the instruction MCMP is not executed, SM is ON, and the error code in SR0 is 16#200B.

2. The description of the flags:

SM607:	It is the matrix comparison flag. ON: Comparing the equivalent values OFF: Comparing the different values
SM608:	The matrix comparison comes to an end. When the last bits are compared, SM608 is ON.
SM609:	When SM609 is ON, the comparison starts from bit 0.
SM610:	It is the matrix bit search flag. When the matching bits are compared, the comparison stops immediately, and SM610 is ON.
SM611:	It is the matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.

API	Instruction code			Operand	Function
0059~ 0064		CMPT※	P	**S_1 · S_2 · n · D**	Comparing the tables

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●		○		○	○		
S_2	●	●			●	●		●	●		○					
n	●	●			●	●		●	●		○	○	○	○		
D		●	●	●				●		●						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
n		●			●	●							
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol：

S_1 : Source device 1

S_2 : Source device 2

n : Data length

D : Comparison result

Explanation:

1. The instruction is used to compare **n** pieces of data in devices starting from **S_1** with those in devices starting from **S_2**. The values compared are signed decimal numbers, and the comparison results are stored in **D**.
2. The operand **n** should be within the range between 1 and 256.
3. The value which is written into the operand **D** is a one-bit value.
4. When the results gotten from the comparison by using the instruction CMPT# are that all devices are ON, SM620 is ON. Otherwise, SM620 is OFF.
5. If the operand **S_1** is a device, the comparison will be as shown below.

6. If the operand **S_1** is a constant within the range between -32768 and 32767, the comparison will be as shown below.

7. The corresponding comparison operation results of the instructions are listed below.

API number	16-bit instruction	Comparison operation result	
		ON	OFF
0059	CMPT =	$S_1 = S_2$	$S_1 \neq S_2$
0060	CMPT < >	$S_1 \neq S_2$	$S_1 = S_2$
0061	CMPT >	$S_1 > S_2$	$S_1 \leqq S_2$
0062	CMPT > =	$S_1 \geqq S_2$	$S_1 < S_2$
0063	CMPT <	$S_1 < S_2$	$S_1 \geqq S_2$
0064	CMPT < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example:

The data in D0~D3 are compared with that in D10~D13. If the comparison result is that the data in D0~D3 is the same as that in D10~D13, Y0.1~Y0.4 will be ON.

Additional remark:

1. If the value in the operand **n** is not within the range between 1 and 256, the instruction is not executed, SM is ON, and the error code in SR0 is 16#200B.
2. If the number of devices specified by **S_1~S_1+n**, **S_2~S_2+n**, or **D** is insufficient, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0065		CHKADR		**S · n · D**	Checking the address of the contact type of pointer register

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S																
n	●	●			●	●		●	●		○	○	○	○		
D		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													
n		●			●	●							
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol：

S : Pointer register

n : Number of devices

D : Check result

Explanation：

1. The instruction CHKADR is used to check whether the value in **S** and (the value in **S**)+**n**-1 exceed the device range. If the check result is that the value in **S** and (the value in **S**)+**n**-1 do not exceed the device range, the device **D** will be ON. Otherwise, it will be OFF.
2. **S** supports the pointer registers D, T, C, HC (POINTER/T_POINTER/C_POINTER/HC_POINTER).
3. The operand **n** should be within the range between 1 and 1024.
4. The instruction CHKADR only can be used in the function block. It is used for the initial program development phase or when not sure if the device range will be exceeded. After the program is written, this instruction can be deleted.

Example:

1. Establish a program (Prog0) and a function block (FB0) in ISPSoft.

Declare two variables in the program.

Local Symbols				
Declaration Type	Identifiers	Address	Type...	Initial Value
VAR	myfb0	N/A [Auto]	FB0	N/A
VAR	StartBit	N/A [Auto]	BOOL	FALSE

2. Declare VarPR1, VarTR1, VarCR1, and VarHCR1 in the function block, and assign the data types POINTER, T_POINTER, C_POINTER, and HC_POINTER to them respectively.

Local Symbols				
Declaration Type	Identifiers	Address	Type...	Initial Value
VAR_IN_OUT	VarPR1	N/A [Auto]	POINTER	N/A
VAR_IN_OUT	VarTR1	N/A [Auto]	T_POINTER	N/A
VAR_IN_OUT	VarCR1	N/A [Auto]	C_POINTER	N/A
VAR_IN_OUT	VarHCR1	N/A [Auto]	HC_POINTER	N/A
VAR	PR_ChkBit	N/A [Auto]	BOOL	FALSE
VAR	TR_ChkBit	N/A [Auto]	BOOL	FALSE
VAR	CR_ChkBit	N/A [Auto]	BOOL	FALSE
VAR	HCR_ChkBit	N/A [Auto]	BOOL	FALSE
VAR	chkPR	N/A [Auto]	BOOL	N/A
VAR	chkTR	N/A [Auto]	BOOL	N/A
VAR	chkCR	N/A [Auto]	BOOL	N/A
VAR	chkHCR	N/A [Auto]	BOOL	N/A

3. Call the function block FB0 in the program, and assign D29999, T0, C511, and HC50 to VarPR1, VarTR1, VarCR1, and VarHCR1 in FB0 respectively.

4. Use the instruction CHKADR to check whether VarPR1, VarTR1, VarCR1, and VarHCR1 exceed the range.

5. When chkPR is ON, the practical device represented by VarPR1 is D29999. Since the legal range of devices is from D0 to D29999, and D29999+10-1=D30008, which exceeds the range, PR_ChkBit is OFF.

6. When chkTR is ON, the practical device represented by VarTR1 is T0. Since the legal range of devices is from T0 to T511, and T0+10-1=T9, which does not exceed the range, TR_ChkBit is ON.

7. When chkCR is ON, the practical device represented by C511. Since the legal range of devices is from C0 to C511, and C511+10-1=C520, which exceeds the range, CR_ChkBit is OFF.

8. When chkHCR is ON, the practical device represented by HC50 is VarHCR1. Since the legal range of deices is from HC0 to HC255, and HC50+10-1=HC59, which does not exceed the range, HCR_ChkBit is ON.

Additional remark:

1. If the value (the practical device address) in **S** exceeds the device range, the instruction CHKADR is not executed, SM is ON, and the error code in SR0 is 16#2003.
2. If the value in the operand **n** is not within the range between 1 and 1024, the instruction CHKADR is not executed, SM is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
0066~ 0071	D	LDZ※		S_1 · S_2 · S_3	Comparing contact type absolute values LDZ※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
S_3	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
S_3		●	●		●	●	●				●	●	

Pulse Instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol：

S_1 ： Data source1

S_2 ： Data source2

S_3 ： Comparison result

Taking LDZ= and DLDZ= for example

Explanation:

1. The absolute value of the difference between S_1 and S_2 is compared with the absolute value of S_3. Take LDZ= for example. If the comparison result is that the absolute value of the differencebetween S_1 and S_2 is equal to the absolute value of S_3, the continuity condition of the instruction is met. If the comparison result is that the absolute value of the difference between S_1 and S_2 is not equalto the absolute value of S_3, the discontinuity condition of the instruction is met.

2. Only the 32-bit instruction can use the 32-bit HC device, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0066	LDZ =	**D**LDZ =	$\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$
0067	LDZ < >	**D**LDZ < >	$\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$
0068	LDZ >	**D**LDZ >	$\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$
0069	LDZ > =	**D**LDZ > =	$\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$
0070	LDZ <	**D**LDZ <	$\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$
0071	LDZ < =	**D**LDZ < =	$\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$

Example:

1. When the absolute difference of D10 and D11 is greater than 200, Y0.0 is ON. While the absolute difference is less than 200, Y0.0 is OFF.

API	Instruction code			Operand	Function
0072~ 0077	D	ANDZ※		S_1 · S_2 · S_3	Comparing contact type absolute values ANDZ※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
S_3	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
S_3		●	●		●	●	●				●	●	

Pulse Instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol：

Taking ANDZ= and DANDZ= for example

S_1 ： Data source1

S_2 ： Data source2

S_3 ： Comparison result

6

Explanation:

1. The absolute value of the difference between S_1 and S_2 is compared with the absolute value of S_3. Take ANDZ= for example. If the comparison result is that the absolute value of the differencebetween S_1 and S_2 is equal to the absolute value of S_3, the continuity condition of the instruction is met. If the comparison result is that the absolute value of the difference between S_1 and S_2 is not equalto the absolute value of S_3, the discontinuity condition of the instruction is met.

2. Only the 32-bit instruction can use the 32-bit HC device, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0072	ANDZ =	DANDZ =	$\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$
0073	ANDZ < >	DANDZ < >	$\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$
0074	ANDZ >	DANDZ >	$\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$
0075	ANDZ > =	DANDZ > =	$\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$
0076	ANDZ <	DANDZ <	$\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$
0077	ANDZ < =	DANDZ < =	$\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$

Example:

1. When M0 is ON and the absolute difference of D10 and D11 is greater than 200, Y0.0 is ON. While the absolute difference is less than 200, Y0.0 is OFF.

API	Instruction code			Operand	Function
0078~0083	D	ORZ※		S_1 · S_2 · S_3	Comparing contact type absolute values ORZ※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
S_3	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
S_3		●	●		●	●	●				●	●	

Pulse Instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol：

Taking ORZ= and DORZ= for example

S_1 : Data source1

S_2 : Data source2

S_3 : Comparison result

6

Explanation:

1. The absolute value of the difference between S_1 and S_2 is compared with the absolute value of S_3. Take ORZ= for example. If the comparison result is that the absolute value of the differencebetween S_1 and S_2 is equal to the absolute value of S_3, the continuity condition of the instruction is met. If the comparison result is that the absolute value of the difference between S_1 and S_2 is not equalto the absolute value of S_3, the discontinuity condition of the instruction is met.

2. Only the 32-bit instruction can use the 32-bit HC device, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0078	ORZ =	**D**ORZ =	$\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$
0079	ORZ < >	**D**ORZ < >	$\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$
0080	ORZ >	**D**ORZ >	$\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$
0081	ORZ > =	**D**ORZ > =	$\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$
0082	ORZ <	**D**ORZ <	$\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$
0083	ORZ < =	**D**ORZ < =	$\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$

Example:

1. When M0 is ON and the absolute difference of D10 and D11 is greater than 200, Y0.0 is ON. While the absolute difference is less than 200, Y0.0 is OFF.

6.2 Arithmetic Instructions

6.2.1 List of Arithmetic Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0100	+	D+	✓	Addition of binary numbers
0101	-	D-	✓	Subtraction of binary numbers
0102	*	D*	✓	Multiplication of binary numbers
0103	/	D/	✓	Division of binary numbers
0104	–	F+	✓	Addition of floating-point numbers
0105	–	F-	✓	Subtraction of floating-point numbers
0106	–	F*	✓	Multiplication of floating-point numbers
0107	–	F/	✓	Division of floating-point numbers
0112	BK+	DBK+	✓	Binary number block addition
0113	BK-	DBK-	✓	Binary number block subtraction
0114	$+	–	✓	Linking the strings
0115	INC	DINC	✓	Adding one to the binary number
0116	DEC	DDEC	✓	Subtracting one from the binary number
0117	MUL16	MUL32	✓	MUL16: Multiplication of binary numbers for 16-bit instructions MUL32: Multiplication of binary numbers for 32-bit instructions
0118	DIV16	DIV32	✓	DIV16: Division of binary numbers for 16-bit instructions DIV32: Division of binary numbers for 32-bit instructions

6.2.2 Explanation of Arithmetic Instructions

API	Instruction code			Operand	Function
0100	D	+	P	S_1 · S_2 · **D**	Addition of binary numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Augend

S_2 : Addend

D : Sum

Explanation:

1. The binary value in S_2 is added to the binary value in S_1, and the sum is stored in **D**.

2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

3. The Flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

4. When the operation result is zero, SM600 is ON. Otherwise, it is OFF.

5. The addition of 16-bit binary values:

   When the operation result exceeds the range of 16-bit binary values, SM602 is ON. Otherwise, it is OFF.

6. The addition of 32-bit binary values:

When the operation result exceeds the range of 32-bit binary values, SM602 is ON. Otherwise, it is OFF.

Example 1:

The addition of 16-bit binary values: When X0.0 is ON, the addend in D10 is added to the augend in D0, and sum is stored in D20.

- When the values in D0 and D10 are 100 and 10 respectively, D0 plus D10 equals 110, and 110 is stored in D20.
- When the values in D0 and D10 are 16#7FFF and 16#1 respectively, D0 plus D10 equals 16#8000, and 16#8000 is stored in D20.
- When the values in D0 and D10 are 16#FFFF and 16#1 respectively, D0 plus D10 equals 16#10000. Since the operation result exceeds the range of 16-bit binary values, SM602 is ON, and the value stored in D20 is 16#0. Besides, since the operation result is 16#0, SM600 is ON.

Example 2:

The addition of 32-bit binary values: When X0.0 is ON, the addend in (D41, D40) is added to the augend in (D31, D30), and sum is stored in (D51, D50). (The data in D30, D40, and D50 is the lower 16-bit data, whereas the data in D31, D41, and D51 is the higher 16-bit data).

- When the values in (D31, D30) and (D41, D40) are 11111111 and 44444444 respectively, (D31, D30) plus (D41, D40) equals 55555555, and 55555555 is stored in (D51, D50).
- When the values in (D31, D30) and (D41, D40) are 16#80000000 and 16#FFFFFFFF respectively, (D31, D30) plus (D41, D40) equals 16#17FFFFFFF. Since the operation result exceeds the range of 32-bit binary values, SM602 is ON, and the value stored in (D51, D50) is 16#7FFFFFFF.

Flag:

The 16-bit instruction:

1. If the operation result is zero, SM600 will be set to ON.

2. If the operation result exceeds 65,535, SM602 will be set to ON.

The 32-bit instruction:

1. If the operation result is zero, SM600 will be set to ON.

2. If the operation result exceeds 4,294,967,295, SM602 will be set to ON.

The 16-bit instruction:

The 32-bit instruction:

API	Instruction code			Operand	Function
0101	D	-	P	S_1 · S_2 · **D**	Subtraction of binary numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Minuend

S_2 : Subtrahend

D : Difference

Explanation:

1. The binary value in S_2 is subtracted from the binary value in S_1, and the difference is stored in **D**.
2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.
3. The Flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)
4. When the operation result is zero, SM600 is ON. Otherwise, it is OFF.
5. When a borrow occurs during the arithmetic, SM601 is ON. Otherwise, it is OFF.

Example 1:

The subtraction of 16-bit binary values: When X0.0 is ON, the subtrahend in D10 is subtracted from the minuend in D0, and the difference is stored in D20.

- When the values in D0 and D10 are 100 and 10 respectively, D0 minus D10 leaves 90, and 90 is stored in D20.
- When the values in D0 and D10 are 16#8000 and 16#1 respectively, D0 minus D10 leaves 16#7FFF, and 16#7FFF is stored in D20.
- When the values in D0 and D10 are 16#1 and 16#2 respectively, D0 minus D10 leaves 16#FFFF. Since the borrow occurs during the arithmetic, SM601 is ON, and the value stored in D20 is 16#FFFF.
- When the values in D0 and D10 are 16#0 and 16#FFFF respectively, D0 minus D10 leaves 16#F0001. Since the borrow occurs during the arithmetic, SM601 is ON, and the value stored in D20 is 16#1.

Example 2: :

The addition of 32-bit binary values: When X0.0 is ON, the subtrahend in (D41, D40) is subtracted from the minuend in (D31, D30), and sum is stored in (D51, D50). (The data in D30, D40, and D50 is the lower 16-bit data, whereas the data in D31, D41, and D51 is the higher 16-bit data).

- When the values in (D31, D30) and (D41, D40) are 55555555 and 11111111 respectively, (D31, D30) minus (D41, D40) D10 leaves 44444444, and 44444444 is stored in (D51, D50).
- When the values in (D31, D30) and (D41, D40) are 16#80000000 and 16#FFFFFFFF respectively, (D31, D30) minus (D41, D40) leaves 16#F80000001. Since the borrow occurs during the arithmetic, SM601 is ON, and the value stored in (D51, D50) is 16#80000001.

API	Instruction code			Operand	Function
0102	D	*	P	**S_1 · S_2 · D**	Multiplication of binary numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Multiplicand

S_2 : Multiplier

D : Product

Explanation:

1. The signed binary value in **S_1** is multiplied by the singed binary value in **S_2**, and the product is stored in **D**.

2. Only the instruction D* can use the 32-bit counter.

3. The multiplication of 16-bit binary values:

The product is a 32-bit value, and is stored in the register (D+1, D), which is composed of 32 bits. When the sign bit b31 is 0, the product is a positive value. When the sign bit b31 is 1, the product is a negative value.

4. The multiplication of 32-bit binary values:

The product is a 64-bit value, and is stored in the register (D+3, D+2, D+1, D0), which is composed of 64 bits. When the sign bit b63 is 0, the product is a positive value. When the sign bit b63 is 1, the product is a negative value.

Example:

The 16-bit value in D0 is multiplied by the 16-bit value in D10, and the 32-bit product is stored in (D21, D20). The data in D21 is the higher 16-bit data, whereas the data in D20 is the lower 16-bit data. Whether the result is a positive value or a negative value depends on the state of the highest bit b31. When b31 is OFF, the result is a positive value. When b31 is ON, the result is a negative value.

D0×D10=(D21, D20)

16-bit value×16-bit value=32-bit value

API	Instruction code			Operand	Function
0103	D	/	P	**S_1 · S_2 · D**	Division of binary numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Dividend

S_2 : Divisor

D : Quotient; remainder

Explanation:

1. The singed binary value in **S_1** is divided by the signed binary value in **S_2**. The quotient and the remainder are stored in **D**.

2. Only the 32-bit instructions can use the 32-bit counter.

3. When the sign bit is 0, the value is a positive one. When the sign bit is 1, the value is a negative one.

4. The division of 16-bit values:

The operand **D** occupies two consecutive devices. The quotient is stored in **D**, and the remainder is stored in **D**+1.

5. The division of 32-bit values:

The operand **D** occupies two devices. The quotient is stored in (**D**+1, **D)**, and the remainder is stored in (**D**+3, **D**+2).

Example:

When X0.0 is ON, the dividend in D0 is divided by the divisor in D10, the quotient is stored in D20, and the remainder is stored in D21. Whether the result is a positive value or a negative value depends on the state of the highest bit.

Additional remark:

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the divisor is 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2012.
3. If the operand **D** used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [2] of WORD/INT.
4. If the operand **D** used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [2] of DWORD/DINT.

API	Instruction code			Operand	Function
0104		F+	P	S_1 · S_2 · **D**	Addition of floating-point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●							○
S_2	●	●			●	●	●	●	●							○
D		●			●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S_1 : Augend

S_2 : Addend

D : Sum

6

Explanation:

1. This instruction is an operation for the addition of 32-bit single-precision floating-point numbers.

2. The floating-point number in S_2 is added to the floating-point number in S_1, and the sum is stored in **D**.

3. The Flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

   - When the operation result is zero, SM600 is ON. Otherwise, it is OFF.
   - When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point number, the value in **D** is 16#FF800000 and SM601 is ON.
   - When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point number, the value in **D** is 16#7F800000 and SM602 is ON.

Example:

The addition of single-precision floating-point numbers: When X0.0 is ON, the addend 16#4046B852 in (D21, D20) is added to the augend 16#3FB9999A in (D11, D10), and the sum 16#4091C28F is stored in (D31, D30). 16#4046B852, 16#3FB9999A, and 16#4091C28F represent the floating point numbers 3.105, 1.450, and 4.555 respectively.

Additional remark:

If the value in $\mathbf{S_1}$ or the value in $\mathbf{S_2}$ exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0105		F-	P	S_1 , S_2 , D	Subtraction of floating-point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●							○
S_2	●	●			●	●	●	●	●							○
D		●			●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S_1 : Minuend

S_2 : Subtrahend

D : Difference

6

Explanation:

1. This instruction is an operation for the subtraction of 32-bit single-precision floating-point numbers.

2. The floating-point number in **S_2** is subtracted from the floating-point number in **S_1**, and the difference is store in **D**.

3. The Flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

   - When the operation result is zero, SM600 is ON.
   - When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point number, the value in **D** is 16#FF800000 and SM601 is ON.
   - When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point number, the value in **D** is 16#7F800000 and SM602 is ON.

Example:

The subtraction of 32-bit single-precision floating-point numbers: When X0.0 is ON, the subtrahend in (D21, D20) is subtracted from the minuend in (D21, D20), and the difference is stored in (D31, D30).

Additional remark:

If the value in $\mathbf{S_1}$ or the value in $\mathbf{S_2}$ exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0106		F*	P	S_1 · S_2 · **D**	Multiplication of floating-point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●							○
S_2	●	●			●	●	●	●	●							○
D		●			●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S_1 : Multiplicand

S_2 : Multiplier

D : Product

6

Explanation:

1. This instruction is an operation for the multiplication of 32-bit single-precision floating-point numbers.

2. The floating-point number in S_1 is multiplied by the floating-point number in S_2, and the product is stored in **D**.

3. The Flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

   - When the operation result is zero, SM600 is ON.
   - When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point number, the value in **D** is 16#FF800000 and SM601 is ON.
   - When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point number, the value in **D** is 16#7F800000 and SM602 is ON.

Example:

The multiplication of 32-bit single-precision floating-point numbers: When X0.0 is ON, the multiplicand 32.5 is multiplied by the multiplier in (D1, D0), and the product is stored in (D11, D10).

Additional remark:

If the value in $\mathbf{S_1}$ or the value in $\mathbf{S_2}$ exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0107		F/	P	S_1 · S_2 · **D**	Division of floating-point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●							○
S_2	●	●			●	●	●	●	●							○
D		●			●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S_1 : Dividend

S_2 : Divisor

D : Quotient

6

Explanation:

1. This instruction is an operation for the division of 32-bit single-precision floating-point numbers.

2. The single-precision floating-point number in S_1 is divided by the single-precision flaoting-point number in S_2. The quotient is stored in **D**.

3. The Flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

   - When the operation result is zero, SM600 is ON.
   - When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point number, the value in **D** is 16#FF800000 and SM601 is ON.
   - When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point number, the value in **D** is 16#7F800000 and SM602 is ON.

Example:

The division of 32-bit single-precision floating-point numbers: When X0.0 is ON, the dividend in (D1, D0) is divided by the divisor 100.7, and the quotient is stored in (D11, D10).

Additional remark:

1. If the divisor is 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2012.
2. If the value in $\mathbf{S_1}$ or the value in $\mathbf{S_2}$ exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0112	D	BK+	P	S_1 · S_2 · n · D	Addition of binary values in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●							
S_2	●	●			●	●	●	●	●				○	○		
n	●	●			●	●	●	●	●				○	○		
D		●			●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●					●	●	
S_2		●			●	●					●	●	
n		●			●	●					●	●	
D		●			●	●					●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Augend

S_2 : Addend

n : Data length

D : Sum

Explanation:

1. **n** pieces of data in devices starting from S_2 are added to those in devices starting from S_1. The augends and the addends are binary values, and the sums are stored in **D**.
2. The operand n should be within the range between 1 and 256.
3. Only the 32-bit instructions can use the 32-bit counter.
4. When the operation result is zero, SM600 is ON.
5. For the 16-bit instructions, when the operation result is less than –32,768, SM601 is ON.
6. For the 16-bit instructions, when the operation result is larger than 32,767, SM602 is ON.
7. For the 32-bit instructions, when the operation result is less than–21,474,836,488, SM601 is ON.
8. For the 32-bit instructions, when the operation result is larger than 2,147,483,647, SM602 is ON.

9. Take the 16-bit instruction as an example, when the operand **S_2** is a device (not a constant or a hexadecimal value):

10. Take the 16-bit instruction as an example, when the operand **S_2** is a constant or a hexadecimal value:

Example 1:

When X0.0 is ON, the binary values in D10~D14 are added to the binary values in D0~D4, and the sums are stored in D100~D104.

Example 2:

When X0.0 is ON, the addend 10 is added to the binary values in D0~D4, and the sums are stored in D100~D104.

Execution result

Additional remark:

1. For the 16-bit instructions, if the devices **S_1~S_1+n**-1, **S_2~S_2+n**-1, or **D~D+n**-1 exceed the device range, the instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.

2. For the 32-bit instructions, if the devices **S_1~S_1**+2*n-1, **S_2~S_2**+2*n-1, or **D~D**+2*n-1 exceed the device range, the instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.

3. If n<1 or n>256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. For the 16-bit instructions, if **S_1~S_1+n**-1 overlap **D~D+n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

5. For the 32-bit instructions, if **S_1~S_1**+2*n-1 overlap **D~D**+2*n-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

6. For the 16-bit instructions, if **S_2~S_2+n**-1 overlap **D~D+n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

7. For the 32-bit instructions, if **S_2~S_2**+2*n-1 overlap **D~D**+2*n-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

API	Instruction code			Operand	Function
0113	D	BK-	P	S_1 , S_2 , **n** , **D**	Subtraction of binary values in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●							
S_2	●	●			●	●	●	●	●				○	○		
n	●	●			●	●	●	●	●				○	○		
D		●			●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●					●	●	
S_2		●			●	●					●	●	
n		●			●	●					●	●	
D		●			●	●					●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Minuend

S_2 : Subtrahend

n : Data length

D : Difference

Explanation:

1. **n** pieces of data in devices starting from S_2 are subtracted to those in devices starting from S_1. The minuends and the subtrahends are binary values, and the differences are stored in **D**.

2. The operand n should be within the range between 1 and 256.

3. Only the 32-bit instructions can use the 32-bit counter.

4. When the operation result is zero, SM600 is ON.

5. For the 16-bit instructions, when the operation result is less than –32,768, SM601 is ON.

6. For the 16-bit instructions, when the operation result is larger than 32,767, SM602 is ON.

7. For the 32-bit instructions, when the operation result is less than–21,474,836,488, SM601 is ON.

8. For the 32-bit instructions, when the operation result is larger than 2,147,483,647, SM602 is ON.

9. Take the 16-bit instruction as an example, when the operand **S_2** is a device (not a constant or a hexadecimal value):

10. Take the 16-bit instruction as an example, when the operand **S_2** is a constant or a hexadecimal value:

Example 1:

When X0.0 is ON, the binary values in D10~D14 are subtracted from the binary values in D0~D4, and the differences are stored in D100~D104.

Example 2:

When X0.0 is ON, the subtrahend 1 is subtracted from the binary values in D0~D4, and the differences are stored in D100~D104.

Execution result

D0	10		1		D100	9
D1	9		1		D101	8
D2	8	-	1	⇨	D102	7
D3	7		1		D103	6
D4	6		1		D104	5

Additional remark:

1. For the 16-bit instructions, if the devices **S_1**~**S_1**+**n**-1, **S_2**~**S_2**+**n**-1, or **D~D**+**n**-1 exceed the device range, the instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.

2. For the 32-bit instructions, if the devices **S_1**~**S_1**+2*__n__-1, **S_2**~**S_2**+2*__n__-1, or **D~D**+2*__n__-1 exceed the device range, the instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.

3. If n<1 or n>256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. For the 16-bit instructions, if **S_1**~**S_1**+**n**-1 overlap **D~D**+**n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

5. For the 32-bit instructions, if **S_1**~**S_1**+2*__n__-1 overlap **D**~**D**+2*__n__-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

6. For the 16-bit instructions, if **S_2**~**S_2**+**n**-1 overlap **D~D**+**n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

7. For the 32-bit instructions, if **S_2**~**S_2**+2*__n__-1 overlap **D**~**D**+2*__n__-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

API	Instruction code			Operand	Function
0114		$+	P	S_1 , S_2 , D	Linking the strings

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●						○	
S_2	●	●			●	●		●	●						○	
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													●
S_2													●
D													●

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : String 1

S_2 : String 2

D : Device in which the string is stored

Explanation:

6

1. When the instruction is executed, the string starting with the data in the device specified by **S_1** (exclusive of 16#00), and the string starting with the data in the device specified by **S_2** (exclusive of 16#00) are linked and moved to the operand D. Besides, the code 16#00 is added to the end of the linked string in the operand D. When the instruction is not executed, the data in D is unchanged.

2. When the data source for **S_1, S_2** or Operand D is not a string ($), the content of the data source is up to 256 characters (the ending code 16#00 included)

3. The string in the operand **S_1** and the string in the operand **S_2** are linked and moved to the operand D, as illustrated below.

S_1	B(16#62)	A(16#61)
S_1+1	D(16#64)	C(16#63)
S_1+2	E(16#65)	(16#00)

S_2	B(16#42)	A(16#41)
S_2+1	D(16#44)	C(16#43)
S_2+2	(16#00)	(16#00)

=

D	d(16#62)	a(16#61)
D +1	d(16#64)	c(16#63)
D +2	B(16#42)	A(16#41)
D +3	D(16#44)	C(16#43)
D +4	(16#00)	(16#00)

Turning into 16#00 automatically

4. When the data source for **S_1, S_2** or Operand D is not a string ($),the ending code 16#00 should be added to the end of the data which is moved.

5. Suppose **S_1** or **S_2** is not a string. When the instruction is executed and the first character is the code 16#00, 16#00 is still linked and moved.

6. The string “abcde” in the Operand **S_1** is shown as below.

S_1	b(16#62)	a(16#61)
S_1+1	d(16#64)	c(16#63)
S_1+2	(16#00)	e(16#65)

Example:

Suppose **S_1** is the string “ab” and **S_2** is the string “c”. After the conditional contact M0 is enabled, the data in D65534 is 16#6261 and the data in D65535 is16#0063.

Additional remark:

1. If **S_1** or **S_2** is a string, at most 31 characters can be moved.

2. If D is not sufficient to contain the string composed of the strings in **S_1** and **S_2**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the string of **S_1+S_2** is with more than 256 character (the ending code 16#00 included), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If **S_1** or **S_2** overlaps D, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

5. If the string in **S_1** or **S_2** does not end with 16#00, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200E.

API	Instruction code			Operand	Function
0115	D	INC	P	**D**	Adding one to the binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

D : Destination device

Explanation:

1. One is added to the value in D.
2. Only the instruction DINC can use the 32-bit counter.
3. When the 16-bit operation is performed, 32,767 plus 1 equals -32,768. When the 32-bit operation is performed, 2,147,483,647 plus 1 equals -2,147,483,648.

Example:

When X0.0 is switched from OFF to ON, the value in D0 increases by one.

API	Instruction code			Operand	Function
0116	D	DEC	P	**D**	Subtracting one from the binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

D : Destination device

Explanation:

1. One is subtracted from the value in D.
2. Only the instruction DDEC can use the 32-bit counter.
3. When the 16-bit operation is performed, -32,768 minus 1 leaves 32,767. When the 32-bit operation is performed, -2,147,483,648 minus 1 leaves 2,147,483,647.

Example:

When X0.0 is switched from OFF to ON, the value in D0 decreases by one.

API	Instruction code			Operand	Function
0117		MUL16 MUL32	P	S_1 · S_2 · D	16-bit binary multiplication 32-bit binary multiplication

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Multiplicand

S_2 : Multiplier

D : Product

6

Explanation:

1. The signed binary value in **S_1** is multiplied by the singed binary value in **S_2**, and the product is stored in D.

2. Only MUL32 can use an HC device.

3. 16-bit binary multiplication:

The product gotten is a 16-bit value. It is stored in **D** which is a 16-bit register. If b15 in **D** is 0, the product stored in **D** is a positive value. If b15 in **D** is 1, the product stored in **D** is a negative value.

4. 32-bit binary multiplication:

The product gotten is a 32-bit value. It is stored in (**D**, **D**+1**)** which is a 32-bit register. If b31 in **D** is 0, the product stored in (**D**, **D**+1**)** is a positive value. If b31 in **D** is 1, the product stored in (**D**, **D**+1**)** is a negative value.

Example:

The 16-bit value in D0 is multiplied by the 16-bit value in D10, and the product is stored in D20. Whether the product is a positive value or a negative value depends on the leftmost bit (bit 15) in D20. If bit 15 in D20 is 0, the product stored in D20 is a positive value. If bit 15 in D20 is 1, the product stored in D20 is a negative value.

D0×D10=D20

16-bit value×16-bit value=16-bit value

Additional remark:

1. If the product of a 16-bit multiplication is not a 16-bit signed value available, and is greater than the maximum 16-bit positive number K32767, or less than the minimum negative number K-32768, the carry flag SM602 will be ON, and only the low 16 bits will be written.

2. If users need the complete result of a 16-bit multiplication (a 32-bit value), they have to use API 0102 */*P. Please refer to the explanation of API 0102 */*P for more information.

3. If the product of a 32-bit multiplication is not a 32-bit signed value available, and is greater than the maximum 32-bit positive number K2147483647, or less than the minimum negative number K-2147483648, the carry flag SM602 will be ON, and only the low 32 bits will be written.

4. If users need the complete result of a 32-bit multiplication (a 64-bit value), they have to use API 0102 D*/D*P. Please refer to the explanation of API 0102 D*/D*P for more information.

API	Instruction code			Operand	Function
0118		DIV16 DIV32	P	S_1 · S_2 · D	16-bit binary division 32-bit binary division

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	●	●		
S_2	●	●			●	●	●	●	●		○	○	●	●		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Dividend

S_2 : Divisor

D : Quotient; remainder

6

Explanation:

1. The singed binary value in S_1 is divided by the signed binary value in S_2. The quotient is stored in D.

2. Only the 32-bit instruction can use an HC device.

3. Sign bit=0 (Positive number); sign bit =1 (Negative number)

4. 16-bit binary division:

The quotient is stored in **D**.

5. 32-bit binary division:

D occupies two consecutive devices. The quotient is stored in (**D**+1, **D**).

Example:

When X0.0 is ON, the dividend in D0 is divided by the divisor in D10, and the quotient is stored in D20. Whether the quotient is a positive value or a negative value depends on the leftmost bit in D20.

Additional remark:

1. If the device used is not available, the instruction will not be executed, SM0 will be ON, and the error code stored in SR0 will be 16#2003.

2. If the divisor used is 0, the instruction will not be executed, SM0 will be ON, and the error code stored in SR0 will be 16#2012.

3. If you want to store the remainder gotten, you have to use "/" instruction (Division of binary values). Please refer to the explanation of API0103 "/" instruction for more information.

6.3 Data Conversion Instructions

6.3.1 List of Data Conversion Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0200	BCD	DBCD	✓	Converting the binary number into the binary-coded decimal number
0201	BIN	DBIN	✓	Converting the binary-coded decimal number into the binary number
0202	FLT	DFLT	✓	Converting the binary integer into the binary floating-point number
0204	INT	DINT	✓	Converting the 32-bit floating-point number into the binary integer
0206	MMOV	–	✓	Converting the 16-bit value into the 32-bit value
0207	RMOV	–	✓	Converting the 32-bit value into the 16-bit value
0208	GRY	DGRY	✓	Converting the binary number into the Gray code
0209	GBIN	DGBIN	✓	Converting the Gray code into the binary number
0210	NEG	DNEG	✓	Two's complement
0211	–	FNEG	✓	Reversing the sign of the 32-bit floating-point number
0212	–	FBCD	✓	Converting the binary floating-point number into the decimal floating-point number
0213	–	FBIN	✓	Converting the decimal floating-point number into the binary floating-point number
0214	BKBCD	–	✓	Converting the binary numbers in blocks into the binary-coded decimal numbers in blocks
0215	BKBIN	–	✓	Converting the binary numbers in blocks into the binary-coded decimal numbers in blocks
0216	SCAL	DSCAL	✓	Scale value operation
0217	SCLP	DSCLP	✓	Parameter type of scale value operation

6.3.2 Explanation of Data Conversion Instructions

API	Instruction code			Operand	Function
0200	D	BCD	P	**S · D**	Converting the binary number into the binary-coded decimal number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○				
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source device

D : Conversion result

Explanation:

1. The binary value in **S** is converted into the binary-coded decimal value, and the conversion result is stored in **D**.

2. Only the instruction DBCD can use the 32-bit counter, but not the device E.

3. The four fundamental operations of arithmetic in the PLC, the instruction INC, and the instruction DEC all involve binary numbers. To show the decimal value on the display, users can use the instruction BCD to convert the binary value into the binary-coded decimal value

Example:

1. When X0.0 is ON, the binary value in D10 is converted into the binary-code decimal value, and the conversion result is stored in D100.

2. If D10=16#04D2=1234, the conversion result will be that D100=16#1234.

Additional remark:

1. If the conversion result exceeds the range between 0 and 9,999, the instruction BCD is not executed, SM0 is ON, and the error code in SR0 is 16#200D (The binary-coded decimal value is represented by the hexadecimal value, but one of digits is not within the range between 0 and 9.).

2. If the conversion result exceeds the range between 0 and 99,999,999, the instruction DBCD is not executed, SM0 is ON, and the error code in SR0 is 16#200D (The binary-coded decimal value is represented by the hexadecimal value, but one of digits is not within the range between 0 and 9.).

API	Instruction code			Operand	Function
0201	D	BIN	P	**S · D**	Converting the binary-coded decimal number into the binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○				
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source device

D : Conversion result

Explanation:

1. The binary-coded decimal value in **S** is converted into the binary value, and the conversion result is stored in **D**.

2. The 16-bit binary-coded decimal value in **S** should be within the range between 0 and 9,999, and the 32-bit binary-coded decimal value in **S** should be within the range between 0 and 99,999,999.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

4. Constants and hexadecimal values are converted into binary values automatically. Therefore, users do not need to use the instruction.

Example:

When X0.0 is ON, the binary-coded decimal value in D0 is converted into the binary value, and the conversion result is stored in D10.

Additional remark:

1. If the value in **S** is not the binary-coded decimal value, the operation error occurs, SM0 is ON, and the error code in SR0 is 16#200D (The binary-coded decimal value is represented by the hexadecimal value, but one of digits is not within the range between 0 and 9.).

2. The application of the instructions BCD and BIN:

   - Before the value of the binary-coded decimal type of DIP switch is read into the PLC, users have to use the instruction BIN to convert the data into the binary value and store the conversion result in the PLC.
   - If users want to display the data stored inside the PLC in a seven-segment display of the binary-coded decimal type, they have to use the instruction BCD to convert the data into the binary-coded decimal value before the data is sent to the seven-segment display.
   - When X1.0 is ON, the binary-coded decimal value in X0.0~X0.15 is converted into the binary value, and the conversion result is stored in D100. Subsequently, the binary value in D100 is converted into the binary-coded decimal value, and the conversion result is stored in Y0.0~Y0.15.

API	Instruction code			Operand	Function
0202	D	FLT	P	**S · D**	Converting the binary integer into the binary floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○				
D		●					●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source device

D : Conversion result

Explanation:

1. The instruction is used to convert the binary integer into the single-precision floating-point number.

2. The operand **S** used in the instruction FLT cannot be the 32-bit counter, but not the device E.

3. The source device **S** used in the instruction FLT occupies one register, and **D** used in FLT occupies two registers.

4. The source device **S** used in the instruction DFLT occupies two registers, and **D** used in DFLT also occupies two registers.

   - When the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, SM602 is ON, and the maximum floating-point number is stored in **D**.

   - When the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, SM601 is ON, and the minimum floating-point number is stored in **D**.

   - When the conversion result is zero, SM600 is ON.

Example 1:

1. When X0.0 is ON, the binary integer in D0 is converted into the single-precision floating-point number, and the conversion result is stored in (D13, D12).

2. When X0.1 is ON, the binary integer in (D1, D0) is converted into the single-precision floating-point number, and the conversion result is stored in (D21, D20).

3. Suppose the value in D0 is 10. When X0.0 is ON, 10 is converted into the single-precision floating-point number 16#41200000, and 16#41200000 is stored in the 32-bit register (D13, D12).

4. Suppose the value in the 32-bit register (D1, D0) is 100,000. When X0.1 is ON, 100,000 is converted into the single-precision floating-point number 16#47C35000, 16#47C35000 is stored in the 32-bit register (D21, D20).

Example 2:

Users can use the applied instructions to perform the following calculation.

- The binary integer in D10 is converted into the single-precision floating-point number, and the conversion result is stored in (D101, D100).
- The binary-coded decimal value in X0.0~X0.15 is converted into the binary value, and the conversion result is stored in D200.
- The binary integer in D200 is converted into the single-precision floating-point number, and the conversion result is stored in (D203, D202).
- The constant 615 is divided by the constant 10, and the quotient which is the single-precision floating-point number is stored in (D301, D300).
- The single-precision floating-point number in (D101, D100) is divided by the single-precision floating-point number in (D203, D202), and the quotient which is the single-precision floating-point number is stored in (D401, D400).
- The single-precision floating-point number in (D401, D400) is multiplied by the single-precision floating-point number in (D301, D300), and the product which is the single-precision floating-point number is stored in (D21, D20).
- The single-precision floating-point number in (D21, D20) is converted into the decimal floating-point number, and the conversion result is stored in (D31, D30).

- The single-precision floating-point number in (D21, D20) is converted into the binary integer, and the conversion result is stored in (D41, D40).

F/
En
D100
S1
D
D400
D202
S2
F*
En
D400
S1
D
D20
D300
S2
FBCD
En
D20
S
D
D30
DINT
En
D20
S
D
D40

API	Instruction code			Operand	Function
0204	D	INT	P	**S · D**	Converting the 32-bit floating-point number into the binary integer

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source device

D : Conversion result

Explanation:

1. The single-precision floating-point number in the register specified by **S** is converted into the binary integer. The binary floating-point number is rounded down to the nearest whole digit, and becomes the bianry integer. The binary integer is stored in the register specified by **D**.

2. The source device **S** used in the instruction INT occupies two registers, and **D** used in INT occupies one register.

3. The source device **S** used in the instruction DINT occupies two registers, and **D** used in DINT also occupies two registers.

4. The operand **D** used in the instruction INT cannot be the 32-bit counter, but not the device E.

5. The instruction INT is the opposite of the instruction FLT.

6. When the conversion result is zero, SM600 is ON.

7. During the conversion, if the floating-point number is rounded down to the nearest whole digit, SM601 will be ON.

8. When the conversion result exceeds the range, SM602 is ON.

9. For the instruction INT/IINTP, the range of conversion results is between -32,768 and 32,767.

10. For the instruction DINT/DINTP, the range of conversion results is between -2,147,483,648 and 2,147,483,647.

Example:

1. When X0.0 is ON, the single-precision floating-point number in (D1, D0) is converted into the binary integer, and the conversion result is stored in D10. The binay floating-point number is rounded down to the nearest whole digit.

2. When X0.1 is ON, the single-precision floating-point number in (D21, D20) is converted into the binary integer, and the conversion result is stored in (D31, D30). The binary floating-point number is rounded down to the nearest whole digit.

Additional remark:

If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0206		MMOV	P	**S · D**	Converting the 16-bit value into the 32-bit value

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●		○	○	○	○		
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●	●				●	●	
D			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Source device

D : Conversion result

Explanation:

The data in the 16-bit device **S** is transmitted to the 32-bit device **D**. The sing bit which is specified is copied repeatedly to the destination.

Example:

When X0.0 is ON, the value of b15 in D4 is transmitted to b15~b31 in (D7, D6). The data in (D7, D6) becomes a negative value.

API	Instruction code			Operand	Function
0207		RMOV	P	**S · D**	Converting the 32-bit value into the 16-bit value

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○		○	○		
D		●			●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S			●				●						
D		●			●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Source device

D : Conversion result

Explanation:

The data in the 32-bit device **S** is transmitted to the 16-bit device **D**. The sing bit which is specified is retained.

6

Example:

When X0.0 is ON, the value of b31 in D7 is transmitted to b15 in D4, the values of b0~b14 are transmitted to the corresponding bits, and the values of b15~b30 are ignored.

API	Instruction code			Operand	Function
0208	D	GRY	P	**S · D**	Converting the binary number into the Gray code

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source device

D : Conversion result

Explanation:

1. The binary value in the device specified by **S** is converted into the Gray code, and the conversion result is stored in the device specified by **D**.

2. Only the instruction DGRY can use the 32-counter, but not the device E.

3. The value in the operand **S** should be within the available range.

   The value in the operand **S** used in the 16-bit instruction should be within the range between 0 and 32,767.

   The value in the operand **S** used in the 32-bit instruction should be within the range between 0 and 2,147,483,647.

Example:

When X0.0 is ON, the constant 6513 is converted into the Gray code, and the conversion result is stored in Y1.0~Y1.15.

Additional remark:

If the value in **S** is less than 0, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0209	D	GBIN	P	**S · D**	Converting the Gray code into the binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source device

D : Conversion result

Explanation:

1. The Gray code in the device specified by **S** is converted into the binary value, and the conversion result is stored in the device specified by **D**.

2. The instruction is used to convert the Gary code in the absolute position encoder which is connected to the input terminal of the PLC to the binary value, and the conversion result is stored in the register which is specified.

3. Only the instruction DGBIN can use the 32-counter, but not the device E.

4. The value in the device **D** should be within the available range.

   The value in the device **D** used in the 16-bit instruction should be within the range between 0 and 32,767.

   The value in the device **D** used in the 32-bit instruction should be within the range between 0 and 2,147,483,647.

Example:

When X0.0 is ON, the Gary code in the absolute position encoder which is connected to the inputs X0.0~X0.15 is converted into the binary value, and the conversion result is stored in D10.

Additional remark:

If the value in **S** is less than 0, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0210	D	NEG	P	**D**	Two's complement

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

D : Device in which the two's complement is stored

Explanation:

1. The instruction is used to convert the negative binary value into the absolute value.

2. Only the instruction DNEG can use the 32-bit counter.

3. Generally, the pulse instructions NEGP and DNEGP are used.

Example 1:

When X0.0 is switched from OFF to ON, all bits in D0 are inverted (0 becomes 1, and 1 becomes 0), and 1 is added to the result. The final value is stored in the original register D10.

Example 2:

The absolute value of the negative value:

1. When the value of the 15th bit in D0 is 1, M0 is ON. (The value in D0 is a negative value.)

2. When M0 is ON, the instruction NEG is used to obtain the two's complement of the negative value in D0. (The

corresponding positive value is obtained.)

Example 3:

The absolute value of the difference between two values:

Suppose X0.0 is ON.

1. When the value in D0 is greater than that in D2, M0 is ON.
2. When the value in D0 is equal to that in D2, M1 is ON.
3. When the value in D0 is less than that in D2, M2 is ON.
4. The value in D4 is a positive value.

Additional remark:

The representation of the value and its absolute value:

1. Whether the data is a positive value or a negative value depends on the value of the highest bit in the register. If the value of the highest in the register is 0, the data is a positive value. If it is 1, the data is a negative value.

2. The negative value can be converted into its absolute value by means of the instruction NEG.

(D0)=2

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

(D0)=1

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(D0)=0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

API	Instruction code			Operand	Function
0211		FNEG	P	**D**	Reversing the sign of the 32-bit floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

D : Device in which the sign of the value is reversed

Explanation:

The sign of the single-precision floating-point number in the device **D** is reversed.

Example:

Before the instruction is executed, the value in (D1, D0) is the negative value 16#AE0F9000. When X0.0 is switched from OFF to ON, the sign of the single-precision floating-point number in (D1, D0) is reversed. In other words, after the instruction is executed, the value in (D1, D0) is the positive value 16#2E0F9000.

Before the instruction is executed, the value in (D1, D0) is the positive value 16#2E0F9000. When X0.0 is switched from OFF to ON, the sign of the single-precision floating-point number in (D1, D0) is reversed. In other words, after the instruction is executed, the value in (D1, D0) is the negative value 16#AE0F9000.

API	Instruction code			Operand	Function
0212		FBCD	P	**S · D**	Converting the binary floating-point number into the decimal floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●		●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Data source

D : Conversion result

Explanation:

1. The single-precision floating-point number in the register specified by **S** is converted into the decimal floating-point number, and the conversion result is stored in the register specified by **D**.

2. The floating-point operation in the PLC is based on the single-precision floating-point numbers, and the instruction FBCD is used to convert the single-precision floating-point number into the decimal floating-point number.

3. The Flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

   When the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, SM602 is ON.

   When the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, SM601 is ON.

   When the conversion result is zero, SM600 is ON.

Example:

When X0.0 is ON, the single-precision floating-point number in (D1, D0) is converted into the decimal floating-point number, and the conversion result is stored in (D3, D2).

Additional remark:

If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0213		FBIN	P	**S · D**	Converting the decimal floating-point number into the binary floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●		●					
D		●			●	●	●	●			●					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Data source

D : Conversion result

Explanation:

1. The decimal floating-point number in the register specified by **S** is converted into the single-precision floating-point number, and the conversion result is stored in the register specified by **D**.

2. Suppose the value in **S** is 1234, and the value in **S**+1 is 3. The value in **S** is converted into 1.234×10^6.

3. The value in **D** should be a single-precision floating-point number, and the values in **S** and **S**+1 represent the decimal real number and the decimal exponent respectively.

4. The instruction FBIN is used to convert the decimal floating-point number into the single-precision floating-point number.

5. The real number of decimal floating-point numbers range from -9,999 to +9,999, the exponents of decimal floating-point numbers range from -41 to +35, and the practical range of decimal floating-point numbers in PLC is between ±1175×10-41 and ±3402×10+35. When the operation result is zero, SM600 is ON.

Example 1:

1. When X0.0 is ON, the decimal floating-point number in the register in (D1, D0) is converted into the single-precision floating-point number, and the conversion result is stored in (D3, D2).

Example 2:

1. Before the floating-point operation is performed, users have to use the instruction FLT to convert the binary integer into the single-precision floating-point number. The premise of the conversion is that the value converted in the binary integer. However, the instruction FBIN can be used to convert the floating-point number into the single-precision floating-point number.

2. When X0.0 is ON, K314 and K-2 are moved to D0 and D1 respectively, and combine into the decimal floating-point number ($3.14=314\times10^{-2}$).

Additional remark:

If the real number of the decimal floating-point number in the operand **S** is not within the range between -9,999 and +9,999, or if the exponent of the decimal floating-point number in the operand **S** is not within the range between -41 and +35, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0214		BKBCD	P	**S · n · D**	Converting the binary numbers in blocks into the binary-coded decimal numbers in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●							
n	●	●			●	●		●	●				○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Data source

n : Data length

D : Conversion result

Explanation:

1. **n** pieces of data (the binary values) in devices starting from **S** are converted into the binary-coded decimal values, and the conversion results are stored in **D**.

2. The operand **n** should be within the range between 1 and 256.

Example:

When M1 is ON, the binary values in D0 and D1 are converted into the binary-coded decimal values, and the conversion results are stored in D4 and D5.

Additional remark:

1. If **n** is less than 1, or when **n** is larger than 256, the instruction is not execute, SM0 is ON, and the error code in SR0 is 16#200B.

2. If the devices specified by **S**+**n**-1 and **D**+**n**-1 exceed the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the conversion result is not within the range between 0 and 9,999, the instruction is not executed, and the error code in SR0 is 16#200D (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.).

4. If **S**~**S**+**n**-1 overlap **D**~**D+n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

API	Instruction code			Operand	Function
0215		BKBIN	P	**S · n · D**	Converting the binary numbers in blocks into the binary-coded decimal numbers in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●							
n	●	●			●	●		●	●				○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Data source

n : Data length

D : Conversion result

Explanation:

1. **n** pieces of data (the binary-coded decimal values) in devices starting from **S** are converted into the binary values, and the conversion results are stored in **D**.

2. The binary-coded decimal value in **S** should be within the range between 0 and 9,999.

3. The operand **n** should be within the rang between 1 and 256.

Example:

When M1 is ON, the binary-code decimal values in D0 and D1 are converted into the binary values, and the conversion results are stored in D4 and D5.

Additional remark:

1. If **n** is less than 1, or when **n** is larger than 256, the instruction is not execute, SM0 is ON, and the error code in SR0 is 16#200B.

2. If the devices specified by **S**+**n**-1 and **D**+**n**-1 exceed the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the data in **S** is not the binary-coded decimal, the instruction is not executed, and the error code in SR0 is 16#200D (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.).

4. If **S**~**S**+**n**-1 overlap **D**~**D+n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

API	Instruction code			Operand	Function
0216		SCAL	P	S_1 , S_2 , S_3 , **D**	Scale value operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●		○	○	○	○		
S_2	●	●			●	●		●	●		○	○	○	○		
S_3	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Data source

S_2 : Slope

S_3 : Offset

D : Destination device

Explanation:

1. The operation equation in the instruction: **D**=(S_1×S_2)÷1,000+S_3

2. To obtain the values in S_2 and S_3, users have to use the slope equation and the offset equation below first, and then round off the results to the nearest whole digit. The final 16-bit values are entered into S_2 and S_3.

   The slope equation: S_2=[(Maximum destination value–Minimum destination value)÷(Maximum source value–Minimum source value)]×1,000

   The offset equation: S_3=Minimum destination value–Minimum source value×S_2÷1,000

   The output curve is as shown below:

Example 1:

1. Suppose the values in S_1, S_2, and S_3 are 500, 168, and -4 respectively. When X0.0 is ON, the instruction SCAL is executed, and the scale value is stored in D0.

2. The operation equation: D0=(500×168)÷1,000+(-4)=80

Example 2:

1. Suppose the values in S_1, S_2, and S_3 are 500, -168, and 534 respectively. When X0.0 is ON, the instruction SCAL is

executed, and the scale value is stored in D10.

2. The operation equation: D10=(500×-168)÷1,000+534=450

Additional remark:

1. Only when the slope and the offset are known can the instruction SCAL be used. If the slope and the offset are unknown, users are suggested to use the instruction SCLP to perform the operation.

2. When the 16-bit instruction is performed, the value entered into $\mathbf{S_2}$ should be within the range between –32,768 and 32,767. (The practical value is within the range between –32,768 and 32,767. If the value in $\mathbf{S_2}$ exceeds the range, please use the instruction SCLP to do the operation.

3. When the 32-bit instruction is performed, the value entered into $\mathbf{S_2}$ should be within the range between -2,147,483,648 and 2,147,483,647. (The practical value is within the range between -2,147,483,648 and 2,147,483,647. If the value in $\mathbf{S_2}$ exceeds the range, please use the instruction SCLP to do the operation.

4. When users use the slope equation, they have to notice that the maximum source value should be larger than the minimum source value. However, the maximum destination value is not necessarily larger than the minimum destination value.

5. When the 16-bit instruction is performed, if the value in **D** is larger than 32,767, the value stored in **D** will be 32,767. If the value in **D** is less than -32,768, the value stored in **D** will be -32,768.

6. When the 32-bit instruction is performed, if the value in **D** is larger than 2,147,483,647, the value stored in **D** will be 2,147,483,6477. If the value in **D** is less than -2,147,483,648, the value stored in **D** will be -2,147,483,648.

API	Instruction code			Operand	Function
0217	D	SCLP	P	S_1 · S_2 · S_3 · **D**	Parameter type of scale value operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		○
S_2	●	●			●	●	●	●	●							
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●		●				
S_2		●	●		●	●	●		●				
D		●	●		●	●	●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Data source

S_2 : Parameter

D : Destination device

Explanation:

1. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

2. Constant usage for the operand S_1

Constant	16-bit instruction	32-bit instruction	
		SM685 ON	SM685 OFF
Constant	○	X	○
hexadecimal	○	X	○
Floating number	X	○	X

The flag SM685 (whether to use floating point operation or not) can only be used for 32-bit instructions.

3. The operand **S_2** used in the 16-bit instruction is set as follows.

Device number	Parameter	Setting range
S_2	Maximum source value	-32,768~32,767
S_2+1	Minimum source value	-32,768~32,767
S_2+2	Maximum destination value	-32,768~32,767
S_2+3	Minimum destination value	-32,768~32,767

4. The operand **S_2** used in the 16-bit instruction occupies four devices.

5. The operand **S_2** used in the 32-bit instruction is set as follows.

Device number	Parameter	Setting range	
		Integer	Floating-point number
S_2、**S_2**+1	Maximum source value	-2,147,483,648~ 2,147,483,647	The range of 32-bit floating-point numbers
S_2+2、3	Minimum source value		
S_2+4、5	Maximum destination value		
S_2+6、7	Minimum destination value		

6. The operand **S_2** used in the 32-bit instruction occupies eight devices.

7. If the values used in the 32-bit instruction are floating-point numbers, SM658 can be set to ON. If the values are decimal integers, SM685 can be set to OFF.

8. The operation equation in the instruction: **D**=[(**S_1**–Minimum source value)×(Maximum destination value–Minimum destination value)]÷(Maximum source value)+Minimum destination value

9. The operational relation between the source value and the destination value:

   y=kx+b

   y=Destination value (D)

   k=Slope=(Maximum destination value–Minimum destination value)÷(Maximum source value–Minimum source value)

   x=Source value (**S_1**)

   b=Offset =Minimum destination value–Minimum source value×Slope

The parameters above are being substituted for y, k, x, and b in the equation y=kx+b, and the operation equation in the instruction is obtained.

y=kx+b=D=k$\mathbf{S_1}$+b=Slope×$\mathbf{S_1}$ + Offset = Slope×$\mathbf{S_1}$ + Minimum destination value–Minimum source value×Slope=Slope×($\mathbf{S_1}$–Minimum source value) + Minimum destination value =($\mathbf{S_1}$–Minimum source value)×(Maximum destination value–Minimum destination value)÷(Maximum source value–Minimum source value) + Minimum destination value

10. If $\mathbf{S_1}$ is larger than the maximum source value, the maximum source value will be the value in $\mathbf{S_1}$. If $\mathbf{S_1}$ is less than the minimum source value, the minimum source value will be the value in $\mathbf{S_1}$. After the input values and the parameters are set, the output curve is as shown below.

Example 1:

1. Suppose the value in $\mathbf{S_1}$ is 500, the maximum source value in D0 is 3,000, the minimum source value in D1 is 200, the maximum destination value in D2 is 500, and the minimum destination value in D3 is 30. When X0.0 is ON, the instruction SCLP is executed, and the scale value is stored in D10.

2. The operation equation: D10=[(500–200)×(500–30)]÷(3,000–200)+30=80.35

   80.35 is rounded off to the nearest whole digit, and becomes 80. 80 is stored in D10.

Example 2:

1. Suppose the value in $\mathbf{S_1}$ is 500, the maximum source value in D0 is 3,000, the minimum source value in D1 is 200, the maximum destination value in D2 is 30, and the minimum destination value in D3 is 500. When X0.0 is ON, the instruction SCLP is executed, and the scale value is stored in D10.

2. The operation equation: D10=[(500–200)×(30–500)]÷(3,000–200)+500=449.64

   449.64 is rounded off to the nearest whole digit, and becomes 450. 450 is stored in D10.

Example 3:

1. Suppose the value in $\mathbf{S_1}$ is 500.0, the maximum source value in D0 is 3000.0, the minimum source value in D2 is 200.0, the maximum destination value in D4 is 500.0, and the minimum destination value in D6 is 30.0. When X0.0 is ON, SM685 is set to ON, the instruction DSCLP is executed, and the scale value is stored in D10.

2. The operation equation: D10=[(500.0–200.0)×(500.0–30.0)]÷(3000.0–200.0)+30.0=80.35

   80.35 is rounded off to the nearest whole digit, and becomes 80.0. 80.0 is stored in D10.

Additional remark:

1. The value in $\mathbf{S_1}$ which is used in the 16-bit instruction should be within the range between the minimum source value and the maximum source value, i.e. between -32,768 and 32,767. If the value exceeds the boundary value, the boundary value is used in the operation.

2. The integer in **S_1** which is used in the 32-bit instruction should be within the range between the minimum source value and the maximum source value, i.e. between -2,147,483,648 and 2,147,483,647. If the integer exceeds the boundary value, the boundary value is used in the operation.

3. The floating-point number in **S_1** which is used in the 32-bit instruction should be within the range between the minimum source value and the maximum source value, i.e. within the range of floating-point numbers. If the floating-point number exceeds the boundary value, the boundary value is used in the operation.

4. When users use the instruction, they have to notice that the maximum source value should be larger than the minimum source value. However, the maximum destination value is not necessarily larger than the minimum destination value.

5. When the maximum source value is the same as the minimum source value, the instruction will not be executed and it will be seen as an operation error; SM0 will be ON and the error code in SR0 is 16#2012.

6. If the operand **S_2** used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [4] of WORD.

7. If the operand **S_2** used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [4] of DWORD.

6.4 Data Transfer Instructions

6.4.1 List of Data Transfer Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0300	MOV	DMOV	✓	Transferring the data
0302	$MOV	–	✓	Transferring the string
0303	CML	DCML	✓	Inverting the data
0304	BMOV	DBMOV	✓	Transferring all data
0305	NMOV	DNMOV	✓	Transferring the data to several devices
0306	XCH	DXCH	✓	Exchanging the data
0307	BXCH	–	✓	Exchanging all data
0308	SWAP	DSWAP	✓	Exchange the high byte with the low byte
0309	SMOV	–	✓	Transferring the digits
0310	MOVB	–	✓	Transferring several bits

6.4.2 Explanation of Data Transfer Instructions

API	Instruction code			Operand	Function
0300	D	MOV	P	**S · D**	Transferring the data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○	○	○		○
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●		●		●	●	
D		●	●		●	●	●		●		●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Data source

D : Data destination

Explanation:

1. When the instruction is executed, the data in **S** is transferred to **D**. When the instruction is not executed, the data in **D** is unchanged.

2. Only the data in **S** which is used in the 32-bit instruction can be the floating-point number.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example:

1. To transfer the 16-bit data, users should use the instruction MOV.

   - When X0.0 is OFF, the data in D0 is unchanged. When X0.0 is ON, the value 10 is transferred to the data register D0.

   - When X0.1 is OFF, the data in D10 is unchanged. When X0.1 is ON, the current value of T0 is transferred to the data register D10.

2. To transfer the 32-bit data, users should use the instruction DMOV.

   - When X0.0 is OFF, the data in (D31, D30) and (D41, D40) is unchanged. When X0.2 is ON, the current value in (D21, D20) is transferred to (D31, D30), and the current value of HC0 is transferred to (D41, D40).

3. To transfer the floating-point number, users should use the instruction DMOV.

   - When X0.3 is OFF, the data in (D51, D50) is unchanged. When X0.3 is ON, the floating-point number 3.450 is converted into the binary floating-point number, and the conversion result is transferred to (D51, D50).

API	Instruction code			Operand	Function
0302		$MOV	P	**S · D**	Transferring the string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●						○	
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													●
D													●

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Data source

D : Data destination

Explanation:

1. When the operand **S** is a string and the insturction is executed, the string is transferred to **D**, and the code 16#00 is added to the end of the data

2. When the operand **S** is not a string, the code 16#00 should be added to the end of the data transferred.

3. When the ending code16#00 cannot be found in **S** for 256 characters in a row or even beyond the device range, the instruction is not executed; SM0 is ON and the error code in SR0 is 16#200E.

4. When the the operand **S** is not a string and the instruction is executed, the string starting with the data in the device specified by **S** (including 16#00) is transferred to **D**. When the instruction is not executed, the data in **D** is unchanged.

5. If D is not sufficient to contain the string composed of the strings in **S**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

6. Suppose the operand **S** is not a string. When the instruction is executed and the first character is the code 16#00, 16#00 is still transferred to **D**.

7. When 16#00 appears in the low byte, the execution of the instruction is as follows.

Before the instruction is executed:

	b15~b8	b7~b0
S	16#31	16#30
S+1	16#33	16#32
S+2	16#35	16#34
S+3	16#30	16#00

	B15~b8	b7~b0
D	16#38	16#39
D+1	16#36	16#37
D+2	16#34	16#35
D+3	16#32	16#33

After the instruction is executed:

	b15~b8	b7~b0
S	16#31	16#30
S+1	16#33	16#32
S+2	16#35	16#34
S+3	16#30	16#00

	b15~b8	b7~b0
D	16#31	16#30
D+1	16#33	16#32
D+2	16#35	16#34
D+3	16#00	16#00

16#30 in the high byte is not transferred.

16#32 in the high byte turns into 16#00.

8. When 16#00 appears in the high byte, the execution of the instruction is as follows.

Before the instruction is executed:

	b15~b8	b7~b0
S	16#31	16#30
S+1	16#33	16#32
S+2	16#00	16#34
S+3	16#37	16#36

	b15~b8	b7~b0
D	16#38	16#39
D+1	16#36	16#37
D+2	16#34	16#35
D+3	16#32	16#33

After the instruction is executed:

	b15~b8	b7~b0
S	16#31	16#30
S+1	16#33	16#32
S+2	16#00	16#34
S+3	16#37	16#36

	b15~b8	b7~b0
D	16#31	16#30
D+1	16#33	16#32
D+2	16#00	16#34
D+3	16#32	16#33

9. When **S** overlaps **D** and the device number of **S** is less than the device number of **D**, the transfer of the data to **D** starts form the ending code 16#00.

Before the instruction is executed:

	b15~b8	b7~b0
D0	16#31	16#30
D1	16#33	16#32
D2	16#35	16#34
D3	16#30	16#00

	b15~b8	b7~b0
D1	16#33	16#32
D2	16#35	16#34
D3	16#30	16#00
D4	16#38	16#37

After the instruction is executed:

	b15~b8	b7~b0
D0	16#31	16#30
D1	16#33	16#32
D2	16#35	16#34
D3	16#30	16#00

	b15~b8	b7~b0
D1	16#31	16#30
D2	16#33	16#32
D3	16#35	16#34
D4	16#00	16#00

Example 1:

Suppose the operand **S** is the even string “1234”. When the conditional contact X0.0 is enabled, the data in D0~D3 is as follows.

The operand **S**:

String	‘1’	‘2’	‘3’	‘4’
Hexadecimal value	16#31	16#32	16#33	16#34

After the instruction is executed, the data in the operand **D** is as follows.

Device	**High byte**	**Low byte**	**Note**
D0	16#32	16#31	‘1’=16#31; ‘2’=16#32
D1	16#34	16#33	‘3’=16#33; ‘4’=16#34
D2	16#00	16#00	The ending code 16#00 is in the low byte. 16#00 is automatically added in the high byte.
D3	Unchanged	Unchanged	

Example 2:

Suppose the operand **S** is the odd string “12345”. When the conditional contact X0.0 is enabled, the data in D0~D3 is as follows.

The operand **S**:

String	‘1’	‘2’	‘3’	‘4’	‘5’
Hexadecimal value	16#31	16#32	16#33	16#34	16#35

After the instruction is executed, the data in the operand **D** is as follows.

Device	**High byte**	**Low byte**	**Note**
D0	16#32	16#31	‘1’=16#31; ‘2’=16#32
D1	16#34	16#33	‘3’=16#33; ‘4’=16#34
D2	16#00	16#35	The ending code 16#00 is in the high byte.
D3	Unchanged	Unchanged	

Example 3:

When the operand **S** is not a string and the ending code 16#00 appears in the low byte, the execution of the instruction is as follows.

The operand **S**:

Device	**High byte**	**Low byte**	**Note**
D100	16#31	16#30	‘1’=16#31; ‘0’=16#30
D101	16#33	16#32	‘3’=16#33; ‘2’=16#32
D102	16#35	16#34	‘5’=16#35; ‘4’=16#34
D103	16#30	16#00	‘0’=16#30; 16#00 is the ending code.

After the instruction is executed, the data in the operand **D** is as follows.

Device	High byte	Low byte	Note
D0	16#31	16#30	‘1’=16#31; ‘0’=16#30
D1	16#33	16#32	‘3’=16#33; ‘2’=16#32
D2	16#35	16#34	‘5’=16#35; ‘4’=16#34
D3	16#00	16#00	The ending code 16#00 is in the low byte. 16#00 is automatically added in the high byte.
D4	Unchanged	Unchanged	

Example 4:

When the operand **S** is not a string and the ending code 16#00 appears in the high byte, the execution of the instruction is as follows.

The operand **S**:

Device	High byte	Low byte	Note
D100	16#31	16#30	‘1’=16#31; ‘0’=16#30
D101	16#33	16#32	‘3’=16#33; ‘2’=16#32
D102	16#00	16#34	16#00 is the ending code. ‘4’=16#34
D103	16#37	16#36	‘7’=16#37; ‘6’=16#36

After the instruction is executed, the data in the operand **D** is as follows.

Device	High byte	Low byte	Note
D0	16#31	16#30	‘1’=16#31; ‘0’=16#30
D1	16#33	16#32	‘3’=16#33; ‘2’=16#32
D2	16#00	16#34	16#00 is the ending code. ‘4’=16#34
D3	Unchanged	Unchanged	

Example 5:

When **S** overlaps **D**, and the device number of **S** is less than the device number of **D**, the transfer of the data to **D** starts form the ending code 16#00.

The operand **S**:

Device	High byte	Low byte	Note
D0	16#31	16#30	‘1’=16#31; ‘0’=16#30
D1	16#33	16#32	‘3’=16#33; ‘2’=16#32
D2	16#35	16#34	‘5’=16#35; ‘4’=16#34
D3	16#30	16#00	‘0’=16#30; 16#00 is the ending code.
D4	16#38	16#37	‘8’=16#38; ‘7’=16#37

After the instruction is executed, the data in the operand **D** is as follows.

Device	High byte	Low byte	Note
D1	16#31	16#30	‘1’=16#31; ‘0’=16#30
D2	16#33	16#32	‘3’=16#33; ‘2’=16#32
D3	16#35	16#34	‘5’=16#35; ‘4’=16#34
D4	16#00	16#00	The ending code 16#00 is in the low byte. 16#00 is automatically added in the high byte.
D5	Unchanged	Unchanged	

API	Instruction code			Operand	Function
0303	D	CML	P	**S · D**	Inverting the data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Data source

D : Data destination

Explanation:

1. The instruction is used to invert all bits in **S**, i.e. 0 becomes 1, and 1 becomes 0. The inversion result is stored in **D**. If the data in **S** is the constant, the constant will be converted into the binary value.

2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example 1:

When X0.0 is ON, all bit in D1 are inverted, and the conversion result is stored in Y0.0~Y0.15.

Example 2:

The circuits below can be represented by means of the instruction CML.

API	Instruction code			Operand	Function
0304	**D**	BMOV	P	**S · D · n**	Transferring all data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●							
D		●			●	●	●	●								
n	●	●			●	●	●	●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Data source

D : Data destination

n : Data length

Explanation:

1. **n** pieces of data in devices starting from the device specified by **S** are transferred to the devices starting from the device specified by **D**.

2. The operand **n** should be within the range between 1 and 256.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

4. In order to prevent the error which results from the overlap between the source devices and the destination devices, the data is transferred in the following way (using the 16-bit instruction as an example).

   When the device number of **S** is larger than the device number of **D**, the data is transferred in the order from ① to ③.

When the device number of **S** is less than the device number of **D**, the data is transferred in the order from ③ to ①.

Example 1:

When X0.0 is ON, the data in D0~D3 is transferred to D20~D23.

Example 2:

In order to prevent the error which results from the overlap between the source devices and the destination devices, the data is transferred in the following way.

1. When the device number of **S** is larger than the device number of **D**, the data is transferred in the order from ① to ③.

2. When the device number of **S** is less than the device number of **D**, the data is transferred in the order from ③ to ①.

Additional remark:

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If **n** is larger than 256, or if **n** is less than 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
0305	D	NMOV	P	**S · D · n**	Transferring the data to several devices

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●				○	○		
D		●			●	●	●	●								
n	●	●			●	●	●	●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Data source

D : Data destination

n : Data length

Explanation:

1. When the instruciton is executed, the data in **S** is transferred to the **n** devices starting from the device specified by **D**. When the instruction is not executed, the data in **D** is unchanged.

2. Only the 32-bit instructions can use the 32-bit counter.

3. The operand **n** used in the instruction NMOV should be within the range between 1 and 256.

Example:

When M0 is ON, 100 is transferred to D0~D9.

Additional remark:

1. If **D**~**D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the operand **n** used in the 16-bit instruciton is larger than 256 or less than 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
0306	D	XCH	P	S_1 · S_2	Exchanging the data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1		●			●	●	●	●				○				
S_2		●			●	●	●	●				○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Data which will be exchanged

S_2 : Data which will be exchanged

Explanation:

1. The data in the device specified by S_1 is exchanged with the data in the device specified by S_2.
2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example 1:

When X0.0 is switched from OFF to ON, the data in D20 is exchanged with the data in D40.

Example 2:

When X0.0 is switched from OFF to ON, the data in D100 is exchanged with the data in D200.

API	Instruction code			Operand	Function
0307		BXCH	P	S_1 · S_2 · n	Exchanging all data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1		●			●	●		●								
S_2		●			●	●		●								
n	●	●			●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Data which will be exchanged

S_2 : Data which will be exchanged

n : Data length

6

Explanation:

1. The data in the devices specified by S_1~S_1+n-1 is exchnaged with the data in the devices specified by S_2~S_2+n-1.

2. The operand **n** used in the instruction should be within the range between 1 and 256.

Example:

When X0.0 is ON, the data in D10~D14 is exchanged with the data in D100~D104.

D10	D11	D12	D13	D14
1	2	3	4	5

D100	D101	D102	D103	D104
16	17	18	19	20

After the instruction is executed

D10	D11	D12	D13	D14
16	17	18	19	20

D100	D101	D102	D103	D104
1	2	3	4	5

Additional remark:

1. If S_1+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If S_2+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the operand **n** used in the instruciton is larger than 256 or less than 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
0308	D	SWAP	P	**S**	Exchange the high byte with the low byte

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Data source

Explanation:

1. When the 16-bit instruction is executed, the data in the low byte in **S** is exchanged with the data in the high byte in **S**.

2. When the 32-bit instruction is executed, the data in the low byte of the high word in **S** is exchanged with the data in the high byte of the high word in **S**, and the data in the low byte of the low word in **S** is exchanged with the data in the high byte of the low word in **S**.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example 1:

When X0.0 is ON, the data in the low byte in D0 is exchanged with the data in the high byte in D0.

Example 2:

When X0.0 is ON, the data in the low byte in D11 is exchanged with the data in the high byte in D11, and the data in the low byte in D10 is exchanged with the data in the high byte in D10.

API	Instruction code			Operand	Function
0309		SMOV	P	**S** · **m_1** · **m_2** · **D** · **n**	Transferring the digits

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●				○	○		
m_1	●	●			●	●		●	●				○	○		
m_2	●	●			●	●		●	●				○	○		
D		●			●	●		●								
n	●	●			●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
m_1		●			●	●							
m_2		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Data source

m_1 : Start digit which will be transferred from the source device

m_2 : Number of digits which will be transferred

D : Data destination

n : Start digit where the source data is stored in the destination device

Explanation:

1. The instruction can be used to allocate and combine the data. When the instruction is executed, the **m_2** digits of the number which start from the **m_1**th digit of the number in **S** are transferred to the **m_2** digits of the number which starts from the **n**th digit of the number in **D**.

2. The operand **m_1** should be within the range between 1 and 4. The operand **m_2** should be within the range between 1 and **m_1**. The operand **n** should be within the range between **m_2** and 4. (Four bits are regarded as a unit.)

3. When SM605 is OFF, the data involved in the instruction is binary-coded decimal numbers.

Suppose the number in **S** is K1234, and the number in **D** is K5678. After the instruction is executed, the number in **S** is 1234, and the number in **D** is 5128.

4. When SM605 is ON, the data involved in the instruction is binary numbers.

Suppose the number in **S** is 16#1234, and the number in **D** is 16#5678. After the instruction is executed, the number in **S** is 16#1234, and the number in **D** is 16#5128.

Example 1:

1. When SM605 is OFF, the data involved in the instruction is binary-coded decimal numbers. When X0.0 is ON, the two digits of the decimal number which start from the fourth digit of the decimal number (the digit in the thousands place of the decimal number) in D10 are transferred to the two digits of the decimal number which start from the third digit of the decimal number (the digit in the hundreds place of the decimal number) in D20. After the instruction is executed, the digits in the thousands place of the decimal number (10^3) and the ones place of the decimal number (10^0) in D20 are unchanged.

2. When the binary-code decimal number exceeds the range between 0 and 9,999, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200D.

Suppose the number in D10 is 1234, and the number in D20 is 5678. After the instruction is executed, the number in D10 is unchanged, and the number in D20 is 5128.

6

Example 2:

When SM605 is ON, the data involved in the instruction is binary numbers. When the instruction SMOV is executed, the binary numbers in D10 and D20 are not trasnformed into the binary-coded decimal numbers, and the digit which is transferred is composed of four bits.

Suppose the number in D10 is 16#1234, and the number in D20 is 16#5678. After the instruction is executed, the number in D10 is unchanged, and the number in D20 is 16#5128.

Example 3:

1. The instruction can be used to combine the values of the DIP switches which are connected to the input terminals whose numbers are not consecutive.

2. The two digits of the value of the DIP switch at the right are transferred to the the two digits of the number which start from the second digit of the number in D2, and the one digit of the value of the DIP switch at the left is transferred to the the first digit of the number in D1.

3. The instruction SMOV can be used to transfer the first digit of the number in D1 to the third digit of the number in D2. In other words, the two DIP switches can be combined into one DIP switch by means of the instruction SMOV.

Additional remark:

1. Suppose the data involved in the instruction is binary-coded decimal numbers. If the number in **S** is not within the range between 0 and 9999, or if the number in **D** is not within the range between 0 and 9999, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200D.

2. If $\mathbf{m_1}$ is less than 1, or if $\mathbf{m_1}$ is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If $\mathbf{m_2}$ is less than 1, or if $\mathbf{m_2}$ is larger than $\mathbf{m_1}$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. If **n** is less than $\mathbf{m_2}$, or if **n** is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
0310		MOVB	P	**S · n · D**	Transferring several bits

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●	●	●	●	●	●	●		●						
n	●	●			●	●		●	●		○	○	○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												
n		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Data source

n : Data length

D : Data destination

Explanation:

1. When the instruciton is executed, **n** pieces of data in devices starting from the device specified by **S** are transferred to the devices starting from the device specified by **D**.

2. When **S** is T, C or HC, only the state of the device is transferred, and the current value of the device is not transferred.

3. The operand **n** should be within the range between 1 and 256. When **n** is less than 1, or when **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

Example:

When X0.0 is ON, the data in D0.8~D0.13 is transferred to D1.2~D1.7.

Additional remark:

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

6.5 Jump Instructions

6.5.1 List of Jump Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0400	CJ	–	✓	Conditional jump
0401	JMP	–	–	Unconditional jump
0402	GOEND	–	–	Jumping to the end of the program

6.5.2 Explanation of Jump Instructions

API	Instruction code			Operand	Function
0400		CJ	P	**S**	Conditional jump

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S																

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Jump destination

Explanation:

1. When some part of the program in the PLC does not need to be executed, users can use CJ or CJP to shorten the scan time. Besides, when a dual output is used, users also can use CJ or CJP.

2. If the program specified by the label is prior to the instruction CJ, the watchdog timer error will occur, and the PLC will stop running. Please use the instruction carefully.

3. The instruction CJ can specify the same label repeatedly.

4. When the instruction is executed, the actions of the devices are as follows.

   - The state of Y, the state of M, and the state of S remain the same as those before the execution of the jump.
   - The timer keeps counting and when the set time is reached, the output T-coil will be driven.
   - For more information on the instructions MC and MCR, please refer to the example 2 below.
   - The general applied instructions are not executed.

Example 1:

1. When X0.0 is ON, the execution of the program jumps from NETWORK1 to LABEL1 (NETWORK3) and skips the NETWORK2.

2. When X0.0 is OFF, the execution of the program starts from NETWORK1 to NETWORK3 in sequence, and the instruction CJ is not executed.

Example 2:

1. The instruction CJ between the instruction MC and the instruction MCR can be used in the five conditions below.

   (a) The execution of the program jumps from the part of the program outside one MC/MCR loop to the part of the program outside another MC/MCR loop.

   (b) The execution of the program jumps from the part of the program outside the MC/MCR loop to the part of the program inside the MC/MCR loop.

   (c) The execution of the program jumps from the part of the program inside the MC/MCR loop to the part of the program inside the MC/MCR loop.

   (d) The execution of the program jumps from the part of the program inside the MC/MCR loop to the part of the program outside the MC/MCR loop.

   (e) The execution of the program jumps from the part of the program inside one the MC/MCR loop to the part of the program inside another the MC/MCR loop.

2. When the instruction MC is executed, the previous state of the switch contact is put onto the top of the stack inside the PLC. The stack is controlled by the PLC, and cannot be changed by users. When the instruction MCR is executed, the previous state of the switch contact is popped from the top of the stack. Under the conditions listed in (b), (d), and (e) above, the number of times the items are pushed onto the stack may be different from the number of times the items are popped from the stack. When this situation occurs, at most 32 items can be pushed onto the stack, and the items can be popped form the stack until the stack is empty. Therefore, when CJ or CJP is used with MC and MCR, users have to be careful of the pushing of the item onto the stack and the popping of the item from the stack.

NETWORK 1
X0.0
MC
En
N0
N
NETWORK 2
X0.2
CJ
En
LABEL 1
S
NETWORK 3
X0.3
CJ
En
LABEL 2
S
NETWORK 4
X0.1
MC
En
N1
N

NETWORK 5 LABEL 2:
SM400
Y0.1
NETWORK 6
MCR
N1
N
NETWORK 7 LABEL 1:
SM400
Y0.0
NETWORK 8
MCR
N0
N

Example 3:

The states of the devices are listed below.

Device	State of the contact before the execution of CJ M0=OFF	State of the contact during the execution of CJ M0=ON	State of the output coil during the execution of CJ M0=ON
Y, M, and S	M1, M2, and M3 are OFF.	M1, M2, and M3 are switched from OFF to ON.	Y0.1*1, M20, and S1 are OFF.
	M1, M2, and M3 are ON.	M1, M2, and M3 are switched from ON to OFF.	Y0.1*1, M20, and S1 are ON.
Timer	M4 is OFF.	M4 is switched from OFF to ON.	The timer is not enabled.
	M4 is ON.	M4 is switched from ON to OFF	The timer keeps counting and when the set time is reached, the output T-coil will be driven.
Accumulative timer	M6 is OFF.	M6 is switched from OFF to ON.	ST1 is not enabled.
	M6 is ON.	M6 is switched from ON to OFF.	The accumulative timer keeps counting and when the set time is reached, the output T-coil will be driven.
Counter	M7 and M10 are OFF.	M10 is ON/OFF.	The counter is not enabled.
	M7 is OFF. M10 is ON/OFF.	M10 is ON/OFF.	C0 stops counting. When M0 is switched OFF, C0 keeps counting.
Applied instruction	M11 is OFF.	M11 is switched from OFF to ON	The applied instruction is not executed.
	M11 is ON.	M11 is switched from ON to OFF	The applied instruction which is skipped is not executed.

*1:Y0.1 is a dual output. When M0 is OFF, Y0.1 is controlled by M1. When M0 is ON, Y0.1 is controlled by M12.

NETWORK 1
M0
CJ
En
LABEL1
S
NETWORK 2
M1
Y0.1
NETWORK 3
M2
M20
NETWORK 4
M3
S1
NETWORK 5
M4
TMR
En
T0
S1
10
S2
NETWORK 6
M5
T1
R
NETWORK 7
M6
TMR
En
ST1
S1
1000
S2
NETWORK 8
M7
C0
R

Additional remark:

Please refer to ISPSoft User Manual for more information about the use of the label (Pointer).

API	Instruction code			Operand	Function
0401		JMP		**S**	Unconditional jump

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S																

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S : Jump destination

Explanation:

1. The execution of the program jumps to the part of the program specified by the label (pointer) without any condition.
2. If the program specified by the label is prior to the instruction JMP, the watchdog timer error will occur, and the PLC will stop running. Please use the instruction carefully.
3. Please refer to the instruction CJ for more information on the states of the devices while executing the instruction.
4. Please refer to ISPSoft User Manual for more information about the use of the label (Pointer).

API	Instruction code			Operand	Function
0402		GOEND		–	Jumping to END

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

Explanation:

1. When the condition is met, the execution of the program jumps to END in the program.

2. Function blocks and interrupt tasks do not support the instruction. Besides, the instruction cannot be etween the instruction FOR and the instruction NEXT.

3. When the instruction GOEND is executed, the instructions skipped are not executed, the data in all devices is unchanged, and the states of all devices are also unchanged.

6.6 Program Execution Instructions

6.6.1 List of Program Execution Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0500	DI	–	–	Disabling the interrupt
0501	EI	–	–	Enabling the interrupt
0503	EIX	–	–	Disabling the specific interrupt
0504	DIX	–	–	Enabling the specific interrupt

6.6.2 Explanation of Program Execution Instructions

API	Instruction code			Operand	Function
0500		DI		-	Disabling the interrupt

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

Explanation:

1. Please refer to API0501 the instruction EI for more information.

API	Instruction code			Operand	Function
0501		EI		-	Enabling the interrupt

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

Explanation:

1. The use of the instruction EI indicates that the interrupt task is allowed to be used in the program. (Please refer to next page for more information on task.)

2. The interrupt task is allowed to be used between the instruction EI and the instruction DI in the program. When there is no part of the program in which the interrupt is disabled, users can choose not to use the instruction DI.

3. During the execution of one interrupt task, a new interrupt generated will not be executed, but will be memorized. Not until the execution of the present interrupt task is complete will the next interrupt task be executed. For example, during the execution of I0 (#1), 2 new I0 (#2, #3) are generated, only #2 I0 will be memorized.

4. When several interrupts occur, the interrupt task which should be executed first has higher priority. When several interrupts occur simultaneously, the interrupt task whose pointer number is smaller is executed first.

5. When the interrupt task occurring between DI and EI, it cannot be executed, and the interrupt request will be ignored. It is suggested not to use the instruction DI to disable interrupts while PLC operates.

6. When the immediate I/O signal is required in the execution of the interrupt task, users can use the instruction REF or the device DX/DY in the program to refresh the state of the I/O.

7. Every interrupt number is with a temporary maskable function. Please refer to the following page for the list of interrupt numbers.

Example:

- Set up the timed interrupt task I601 to 500ms in HWCONFIG of ISPSoft.
- When the PLC runs and once the program Cyclic_0 scans the instruction EI, the interrupt task I601 is enabled and the interrupt task is executed. When the execution of the interrupt task is complete, the main program is executed.
- When M0 is ON, the I601 timer interrupt task will be disabled.

- When M1 is ON, the I601 timer interrupt task will be enabled.
- When M2 is ON, the SR623 is 0 and the I601 timed interrupt task will be disabled.
- When M3 is ON, the SR623 is 1 and the I601 timed interrupt task will be enabled.

The program Cyclic_0:

The interrupt task:

Timer interrupts in a diagram:

Additional remark:

There are 7 types of interrupt tasks:

1. External interrupts (I000~I115)

   I000 represents the input X0.0 is falling edge triggered. I100 represents the input X0.1 is rising edge triggered. I101 represents the input X0.1 is falling edge triggered. And the rest can be done in the same manner.

2. Hardware high-speed comparator interrupts (I200~I253)

   This type of interrupts can be further divided into 6 groups. Each group is corresponding to a hardware high-speed counter. (refer to the instruction DCNT for more information). Each group is assigned with 4 interrupt numbers (refer to the instruction DHSCS for more information). For example, the interrupt numbers for the first group is I200~I203, and for the second group is I210~I213 and so on.

3. Software high-speed comparator interrupts (I260~I267)

   There are 8 interrupts for software high-speed comparators and these 8 interrupts are shared with 8 high-speed counters.

4. Communication interrupts

   The communication interrupt can be used as the instruction RS, that is, the receiving of the specific character triggers the interrupt, or can be used as the general interrupt. Please refer to API1812 COMRS for more information.

   COM1: I300

   COM2: I302

   Card 1: I304

   Card 2: I306

5. Extension module interrupts (I400~I431)

Each module is assigned with one interrupt. Users can set up 1 interrupt service for each extension module.

6. High-speed output interrupts (I500~I519)

When the pulse outputting is complete, the interrupt request will be sent; the interrupts (I500~I505) for the complete execution of the positioning instruction should work with special devices (SM) to activate the interrupt service. For example, when the instruction DDRVI completes the execution of the first axis, the interrupt request I500 will be sent; users can set the SM471 to ON to activate the interrupt service. The interrupts (I510~I519) for the complete execution of the position planning table instruction should work with the instruction TPO. When the pulse outputting is complete, the interrupt request will be sent.

7. Timer interrupts (I601~I604)

The timer interrupts should be set in HWCONFIG.

The timer interrupts I601~I603: The default value is 10 milliseconds (unit: 1ms) (1~2000 milliseconds).

The timer interrupts I604: The default value is 1 milliseconds (unit: 0.1ms) (0.1~200 milliseconds).

The complete interrupt numbers, descriptions and the maskable interrupts (SR) are listed below.

Interrupt number	Description	Maskable interrupts SR	Bit No.
I000	External interrupt: the input X0.0 is falling edge triggered.	SR623	0
I001	External interrupt: the input X0.1 is falling edge triggered.		1
I002	External interrupt: the input X0.2 is falling edge triggered.		2
I003	External interrupt: the input X0.3 is falling edge triggered.		3
I004	External interrupt: the input X0.4 is falling edge triggered.		4
I005	External interrupt: the input X0.5 is falling edge triggered.		5
I006	External interrupt: the input X0.6 is falling edge triggered.		6
I007	External interrupt: the input X0.7 is falling edge triggered.		7
I008	External interrupt: the input X0.8 is falling edge triggered.		8
I009	External interrupt: the input X0.9 is falling edge triggered.		9
I010	External interrupt: the input X0.10 is falling edge triggered.		10
I011	External interrupt: the input X0.11 is falling edge triggered.		11
I012	External interrupt: the input X0.12 is falling edge triggered.		12
I013	External interrupt: the input X0.13 is falling edge triggered.		13

Interrupt number	Description	Maskable interrupts SR	Bit No.
I014	External interrupt: the input X0.14 is falling edge triggered.		14
I015	External interrupt: the input X0.15 is falling edge triggered.		15
I100	External interrupt: the input X0.0 is rising-edge triggered.	SR624	0
I101	External interrupt: the input X0.1 is rising-edge triggered.		1
I102	External interrupt: the input X0.2 is rising-edge triggered.		2
I103	External interrupt: the input X0.3 is rising-edge triggered.		3
I104	External interrupt: the input X0.4 is rising-edge triggered.		4
I105	External interrupt: the input X0.5 is rising-edge triggered.		5
I106	External interrupt: the input X0.6 is rising-edge triggered.		6
I107	External interrupt: the input X0.7 is rising-edge triggered.		7
I108	External interrupt: the input X0.8 is rising-edge triggered.		8
I109	External interrupt: the input X0.9 is rising-edge triggered.		9
I110	External interrupt: the input X0.10 is rising-edge triggered.		10
I111	External interrupt: the input X0.11 is rising-edge triggered.		11
I112	External interrupt: the input X0.12 is rising-edge triggered.		12
I113	External interrupt: the input X0.13 is rising-edge triggered.		13
I114	External interrupt: the input X0.14 is rising-edge triggered.		14
I115	External interrupt: the input X0.15 is rising-edge triggered.		15
I200	High-speed comparator interrupt 1 of the hardware high-speed counter 1	SR625	0
I201	High-speed comparator interrupt 2 of the hardware high-speed counter 1		1
I202	High-speed comparator interrupt 3 of the hardware high-speed counter 1		2
I203	High-speed comparator interrupt 4 of the hardware high-speed counter 1		3
I210	High-speed comparator interrupt 1 of the hardware high-speed counter 2		4
I211	High-speed comparator interrupt 2 of the hardware high-speed counter 2		5
I212	High-speed comparator interrupt 3 of the hardware		6

Interrupt number	Description	Maskable interrupts SR	Bit No.
	high-speed counter 2		
I213	High-speed comparator interrupt 4 of the hardware high-speed counter 2		7
I220	High-speed comparator interrupt 1 of the hardware high-speed counter 3		8
I221	High-speed comparator interrupt 2 of the hardware high-speed counter 3		9
I222	High-speed comparator interrupt 3 of the hardware high-speed counter 3		10
I223	High-speed comparator interrupt 4 of the hardware high-speed counter 3		11
I230	High-speed comparator interrupt 1 of the hardware high-speed counter 4		12
I231	High-speed comparator interrupt 2 of the hardware high-speed counter 4		13
I232	High-speed comparator interrupt 3 of the hardware high-speed counter 4		14
I233	High-speed comparator interrupt 4 of the hardware high-speed counter 4		15
I240	High-speed comparator interrupt 1 of the hardware high-speed counter 5	SR626	0
I241	High-speed comparator interrupt 2 of the hardware high-speed counter 5		1
I242	High-speed comparator interrupt 3 of the hardware high-speed counter 5		2
I243	High-speed comparator interrupt 4 of the hardware high-speed counter 5		3
I250	High-speed comparator interrupt 1 of the hardware high-speed counter 6		4
I251	High-speed comparator interrupt 2 of the hardware high-speed counter 6		5
I252	High-speed comparator interrupt 3 of the hardware high-speed counter 6		6

Interrupt number	Description	Maskable interrupts SR	Bit No.
I253	High-speed comparator interrupt 4 of the hardware high-speed counter 6		7
I260	High-speed comparator interrupt 1 of the software high-speed counter	SR627	0
I261	High-speed comparator interrupt 2 of the software high-speed counter		1
I262	High-speed comparator interrupt 3 of the software high-speed counter		2
I263	High-speed comparator interrupt 4 of the software high-speed counter		3
I264	High-speed comparator interrupt 5 of the software high-speed counter		4
I265	High-speed comparator interrupt 6 of the software high-speed counter		5
I266	High-speed comparator interrupt 7 of the software high-speed counter		6
I267	High-speed comparator interrupt 8 of the software high-speed counter		7
I300	Receiving some specific word to trigger communication interruption in COM1	SR628	0
I301	Reserved		1
I302	Receiving some specific word to trigger communication interruption in COM1		2
I303	Reserved		3
I304	Receiving some specific word to trigger communication interruption in function card 1		4
I305	Reserved		5
I306	Receiving some specific word to trigger communication interruption in function card 2		6
I307	Reserved		7
I500	High-speed output interrupt: the positioning instruction completes the execution of the 1st axis	SR629	0

Interrupt number	Description	Maskable interrupts SR	Bit No.
I501	High-speed output interrupt: the positioning instruction completes the execution of the 2nd axis		1
I502	High-speed output interrupt: the positioning instruction completes the execution of the 3rd axis		2
I503	High-speed output interrupt: the positioning instruction completes the execution of the 4th axis		3
I504	High-speed output interrupt: the positioning instruction completes the execution of the 5th axis		4
I505	High-speed output interrupt: the positioning instruction completes the execution of the 6th axis		5
I510	High-speed output interrupt 1: the position planning table instruction completes the execution	SR630	0
I511	High-speed output interrupt 2: the position planning table instruction completes the execution		1
I512	High-speed output interrupt 3: the position planning table instruction completes the execution		2
I513	High-speed output interrupt 4: the position planning table instruction completes the execution		3
I514	High-speed output interrupt 5: the position planning table instruction completes the execution		4
I515	High-speed output interrupt 6: the position planning table instruction completes the execution		5
I516	High-speed output interrupt 7: the position planning table instruction completes the execution		6
I517	High-speed output interrupt 8: the position planning table instruction completes the execution		7
I518	High-speed output interrupt 9: the position planning table instruction completes the execution		8
I519	High-speed output interrupt 10: the position planning table instruction completes the execution		9
I601	Timer interrupts 1 (unit 1ms)	SR632	0
I602	Timer interrupts 1 (unit 1ms)		1
I603	Timer interrupts 1 (unit 1ms)		2

Interrupt number	Description	Maskable interrupts SR	Bit No.
I604	Timer interrupts 1 (unit 0.1ms)		3

Note: When several interrupts occur simultaneously, the interrupt task whose pointer number is smaller will be executed first. PLC will complete the execution of the on-going interrupt, and then execute other occurring interrupts according to their pointer number. For example, during the execution of the interrupt I400, if I500 and I300 occur simultaneously, after the execution of I400 is done, PLC will execute the I300 (smaller pointer number).

API	Instruction code			Operand	Function
0503		EIX		**S**	Enabling the specific interrupt

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S													○			

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S : Data source

Explanation:

1. Only decimal numbers can be inputted in S and the number inputted should be an interrupt number. If the number inputted is not an interrupt number, the instruction will not be executed and no warning will be shown. Users should input EIX500 when they wish to enable the interrupt I500. Please refer to the interrupt number list in the explanation of the instruction EI.

2. The default for interrupt tasks in the AS series is enabled. If the instruction DIX is used to disable the interrupts, users need to use the instruction EIX to enable the interrupts.

3. Users can use the instruction to enalbe the interrupt tasks in SR623~SR634.

4. If the instruction is not executed, an interrupt task to be performed or not is up to the contents of SR623~SR634.

5. Please refer to the examples in API0501 the instruction EI for more information.

API	Instruction code			Operand	Function
0504		DIX		**S**	Disabling the specific interrupt

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S													○			

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S : Data source

Explanation:

1. Only decimal numbers can be inputted in S and the number inputted should be an interrupt number. If the number inputted is not an interrupt number, the instruction will not be executed and no warning will be shown. Users should input DIX500 when they wish to disable the interrupt I500. Please refer to the interrupt number list in the explanation of the instruction EI.

2. The default for interrupt tasks in the AS series is enabled. Users can use the instruction DIX to disable the interrupts.

3. Users can use the instruction to disable the interrupt tasks in SR623~SR634.

4. If the instruction is not executed, an interrupt task to be performed or not is up to the contents of SR623~SR634.

5. Please refer to the examples in API0501 the instruction EI for more information.

6.7 I/O Refreshing Instructions

6.7.1 I/O List of I/O Refreshing Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0600	REF	–	✓	Refreshing the I/O

6.7.2 Explanation of I/O Refreshing Instructions

API	Instruction code			Operand	Function
0600		REF	P	**D · n**	Refreshing the I/O

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D	○	○														
n								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

D : I/O point whose state is refreshed

n : Number of I/O points whose states are refreshed

6

Explanation:

1. The I/O states are not refreshed until the instruction END is executed. When the scanning of the program starts, the states of the external inputs are read and stored in the memory. After the instruction END is executed, the states of the outputs in the memory is sent to the output terminals. Therefore, when users need the latest I/O data in the operation process, they can use this instruction or use the device DX/DY to execute input/output.

2. The operand **n** should be within the range between 1 and 16.

Example 1:

1. When X0.0 is ON, the PLC reads the states of the inputs X0.0~X0.15 immediately. The input signals are refreshed without any delay.

Example 2:

When X0.0 is ON, the output signals from Y0.0~Y0.7 are sent to the output terminals. The output signals are refreshed immediately without the need to wait for the execution of the instruction END.

Additional remark:

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is larger than 16, or if **n** is less than 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

6.8 Convenience Instructions

6.8.1 The List of Convenience Instructions

API	Instruction code		Pulse instruction	Function
	16-Bit	32-Bit		
0700	ALT	–	✓	Alternating between ON and OFF
0701	TTMR	–	–	Teaching timer
0702	STMR	–	–	Special timer
0703	RAMP	DRAMP	–	Cyclic ramp signal
0704	MTR	–	–	Matrix input
0705	ABSD	DABSD	–	Absolute drum sequencer
0706	INCD	–	–	Incremental drum sequencer
0708	–	DPIDE	–	PID algorithm
0709	XCMP	–	–	Setups for comparing the inputs of multiple work stations
0710	YOUT	–	–	Comparing the outputs of multiple work stations

6.8.2 Explanation of Convenience Instructions

API	Instruction code			Operand	Function
0700		ALT	P	**D**	Alternating between ON and OFF

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D	●	●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												

Pulse instruction	16-Bit instruction	32-Bit instruction
AS	AS	-

Symbol:

D : Destination device

Explanation:

1. When the instruction ALT is executed, the state of the device specified by **D** alternate between ON and OFF.

2. Generally, the pulse instruction ALTP is used.

Example 1:

When X0.0 is switched from OFF to ON for the first time, Y0.0 is ON. When X0.0 is switched from OFF to ON for the second time, Y0.0 is OFF.

Example 2:

In the beginning, M0 is OFF. Therefore, Y0.0 is ON, and Y0.1 is OFF. When X0.0 is switched from OFF to ON for the first time, M0 is ON. Therefore, Y0.1 is ON, and Y0.0 is OFF. When X0.0 is switched from OFF to ON for the second time, M0 is OFF. Therefore, Y0.0 is ON, and Y0.1 is OFF.

Example 3:

When X0.0 is ON, T0 generates a pulse every two seconds. The output Y0.0 alternates between ON and OFF according to the pulses generated by T0.

API	Instruction code			Operand	Function
0701		TTMR		**D · n**	Teaching timer

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
D		●						●			○					
n	●	●						●	●		○		○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							
n		●			●	●							

Pulse instruction	16-Bit instruction	32-Bit instruction
-	AS	-

Symbol:

D : Device in which the time is stored

n : Multiplier

Explanation:

1. A second is taken as the timing unit. The time for which the button switch has been turned ON is multiplied by **n**, and the product is stored in **D**. **D+1** is for system use only. When the instruction is executed, the value in **D+1** cannot be altered. Otherwise, the time will be counted incorrectly.

2. When the conditional contact is ON, **D** is reset to 0.

3. Setting the multiplier: When **n** is 0, **D** takes a second as the timing unit. When **n** is 1, the time for which the button switch has been turned ON is multiplied by 10, and **D** takes 100 milliseconds as the timing unit. When **n** is 2, the time for which the button switch has been turned ON is multiplied by 100, and **D** takes 10 milliseconds as the timing unit.

n	D
K0 (unit: 1 second)	1×T
K1 (unit: 100 milliseconds)	10×T
K2 (unit: 10 milliseonds)	100×T

4. When the on-line editing is used, please reset the conditional contact to initialize the instruction.

5. The operand **n** should be within the range between 0 and 2.

Example 1:

1. The time for which the button switch X0.0 has been turned ON is multiplied by n, and the product is stored in D0. Users can use the button switch to adjust the settings.

2. When X0.0 is switched OFF, the value in D0 is unchanged.

Additional remark:

1. If **D**+1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 0, or if **n** is larger than 2, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

API	Instruction code			Operand	Function
0702		STMR		**S · m · D**	Special timer

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					○											
m	●	●						●	●		○		○	○		
D		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S											●		
m		●			●	●							
D	●												

Pulse instruction	16-Bit instruction	32-Bit instruction
-	AS	-

Symbol:

S : Timer number (T0~T511)

m : Setting value of the timer

D : Output device

Explanation:

1. The instruction STMR is used to generate the off-delay relay, the one-shot circuit, and the flashing circuit.

2. The timer specified by the instruction TMR takes 100 milliseconds as the timing unit.

3. The timer specified by the instruction STMR cannot be used repeatedly.

4. **D** occupies four consecutive devices.

5. Before the instruction is executed, please reset **D**~**D**+3.

6. When the conditional contact is not enabled and the value of the device meets one of the two conditions mentioned below, **D**, **D**+1, and **D**+3 are ON for **m** seconds before they are switched OFF. When the conditional contact is not enabled and the value of the device does not meet either of the two conditions mentioned below, **D**~**D**+3 keep OFF.

   - The value of the timer is less than or equal to **m**, **D** is ON, and **D**+1 is OFF.

   - The value of the timer is less than **m**, **D** +2 is OFF, and **D**, **D**+1, and **D**+3 are ON.

7. When the on-line editing is used, please reset the conditional contact to initialize the instruction.

8. The operand **m** should be within the range between 1 and 32767.

Example:

1. When X0.0 is ON, the instruction STMR specifies the timer T0, and the setting value of T0 is five seconds.

2. Y0.0 is the off-delay contact. When X0.0 is switched from OFF to ON, Y0.0 is ON. Five minutes after X0.0 is switched from ON to OFF, Y0.0 is OFF.

3. When X0.0 is switched from ON to OFF, Y0.0 is ON for five seconds.

4. When X0.0 is switched from OFF to ON, Y0.2 is ON for five seconds.

5. Five seconds after X0.0 is switched from OFF to ON, Y0.3 is ON. Five seconds after X0.0 is switched from ON to OFF, Y0.3 is OFF.

6. When the conditional contact X0.0 is followed by the b contact of Y0.3, the flashing currents pass through Y0.1 and Y0.2. When X10 is switched OFF, Y0.0, Y0.1, and Y0.3 are switched OFF, and T10 is reset to 0.

Additional remark:

1. If **D**+3 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **m** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [4] of BOOL.

API	Instruction code			Operand	Function
0703	D	RAMP		S_1 · S_2 · D · n	Cyclic Ramp signal

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●					●	●	●		○	○	○	○		
S_2	●	●					●	●	●		○	○	○	○		
D		●					●	●								
n	●	●					●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-Bit instruction	32-Bit instruction
-	AS	AS

Symbol:

S_1 : Initial value of the ramp signal

S_2 : Final value of the ramp signal

D : Duration of the ramp signal

n : Number of scan cycles

Explanation:

1. The instruction is used to get the slope. The slope is linear, and has an absolute relationship with the scan time. Thus it is suggested to set a fixed scan time or write this instruction to the timer interrupt task.

2. The initial value of the ramp signal and the final value of the ramp signal are written into S_1 and S_2 respectively in advance. When X0.0 is ON, D increases from the setting value in S_1 to the setting value in S_2. The number of scan cycles is stored in D+1. When the value in D is equal to that in S_2, or when the value in D+1 is equal to **n**, reached to the scan cycles, SM687 is ON.

3. When the conditional contact is not enabled, the value in D, and D+1 are both 0, and SM687 is OFF.

4. When the on-line editing is used, please reset the conditional contact to initialize the instruction.

5. Please refer to ISPSoft User Manual for more information related to the fixing of the scan time.

6. The operand **n** should be within the range between 1 and 32767. When the operand n is out of the range, this instruction will not be executed.

7. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

8. Use with the flag SM686, the setting value in D is reset to 0. Refer to the examples below for more details.

Example:

When the instruction is used with the analog signal output, the action of cushioning the start/stop can be executed.

1. Suppose the instruction is being executed. When X0.0 is switched OFF, the execution of the instruction stops. When X0.0 is ON again, SM687 is OFF, D12 is reset to the setting value in D10, D13 is reset to 0, and the calculation is restarted.

2. SM686 is OFF, and when D12 is reaching the setting value in D11, SM687 is ON as a scan cycle. And when D12 is reset to the setting value in D10, D13 is reset to 0.

3. When SM686 is ON, and D12 is reaching the setting value in D11, the value in D12 will not be reset to 0, and SM687 is ON. Till the conditional contact is closed, the value in D12 will be reset to 0 and SM687 will be OFF. When SM686 is ON/OFF, the value in D12 changes as follows.

Additional remark:

1. If **D**+1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. When the 16-Bit instruction is executed, if users declare the operand **D** in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

4. When the 32-Bit instruction is executed, if users declare the operand **D** in ISPSoft, the data type will be ARRAY [2] of DWORD/DINT.

API	Instruction code			Operand	Function
0704		MTR		**S**、**D_1**、**D_2**、**n**	Matrix input

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	○															
D_1		○														
D_2		○	○	○				○								
n	●	●						●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												
D_1	●												
D_2	●												
n		●			●	●							

Pulse instruction	16-Bit instruction	32-Bit instruction
-	AS	-

Symbol:

S : Initial input device in the matrix scan

D_1 : Initial output device in the matrix scan

D_2 : Initial corresponding device in the matrix scan

n : Number of rows which are scanned

Explanation:

1. **S** specifies the initial input device in the matrix scan. The eight devices starting from the device specified by **S** are the input devices in the matrix scan.

2. **D_1** specifies the transistor output device Y as the initial device in the matrix scan. When the conditional contact is OFF, the states of the **n** devices starting from **D_1** are OFF.

3. One row of inputs is refreshed every scan cycle. There are 16 inputs in a row, and the scan starts from the first row to the **n^{th}** row.

4. The eight input devices starting from the device specified by **S** are connected to the **n** output devices starting from the device specified by **D_1** to form the **n** rows of switches. The states of the **n** rows of switches are read in the matrix scan, and stored in the devices starting from the device specified by **D_2**.

5. When the instruction is used, users can connect at most 8 rows of input switches in parallel to get 64 inputs (8×8=64).

6. The interval between the time when the instruction is executed and the next time when it is executed should be longer than the time it takes for the states of the I/O points on the module to be refreshed. Otherwise, the correct states of the inputs cannot be read.

7. Generally, the conditional contact used in the instruction is the normally-open contact SM400.

8. The operand **n** should be within the range between 2 and 8.

Example 1:

1. When M0 is ON, the instruction MTR is executed. The states of the two rows of switches are read in order, and stored in the internal relays M10~M17 and M20~M27 respectively.

2. The diagram below is the external wiring diagram of the 2-by-8 matrix input circuit which is composed of X0.0~X0.7 and Y0.0~Y0.7. The corresponding internal relays of the 16 switches are M10~M17 and M20~M27.

3. The eight input devices starting from X0.0 are connected to the two output devices starting from Y0.0 to form the two rows of switches. The states of the two rows of switches are read in the matrix scan, and stored in the devices starting from M10 specified by $\mathbf{D_2}$. That is, the states of the first row of switches are stored in M10~M17, and the states of the second row of switches are stored in M20~M27.

Additional remark:

1. When this instruction is executed, a too long or a too short scan cycle time will cause the state of the switches not be read correctly. Use the following tips to solve the issues.

   - When the scan cycle is too short, the I/O may not be able to respond in time and the correct states of the inputs cannot be read. Users can set a fixed scan time to solve this issue.

   - When the scan cycle is too long, the switch may become slow to react. Users can write this instruction to the timer interrupt task to set a fixed to execute this instruction.

2. If **S**+7, $\mathbf{D_1}$+**n**-1, or $\mathbf{D_2}$+(**n***8)-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If **n** is less than 2, or if **n** is larger than 8, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. If users declare the operand **S** in ISPSoft, the data type will be ARRAY [8] of BOOL.

API	Instruction code			Operand	Function
0705	D	ABSD		**S_1 · S_2 · D · n**	Absolute drum sequencer

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●							
S_2	●	●			●	●	●	●	●							
D		●	●	●				●								
n	●	●						●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D	●												
n		●	●		●	●	●						

Pulse instruction	16-Bit instruction	32-Bit instruction
-	AS	AS

Symbol:

S_1 : Initial device in the comparison

S_2 : Comparison value

D : Comparison result

n : Number of comparison groups

Explanation:

1. The instruction ABSD is used to generate multiple pulses corresponding to the current values of the counter.

2. Only the instruction DABSD can use the 32-Bit counter, but not the device E.

3. When the instruction ABSD is used, **n** should be within the range between 1 and 256.

Example 1:

1. Before the instruction ABSD is executed, the instruction MOV is used to write the setting values in D100~D107. The values in the even devices are minimum values, and the values in the odd devices are maximum values.

2. When X0.0 is ON, the current value of the counter C10 is compared with the maximum values and the minimum values in D100~D107, and the comparison results are stored in M10~M13.

3. When X0.0 is OFF, the original states of M10~M13 are unchanged.

4. When the current value of C10 is within the range between the minimum value and the maximum value, M10~M13 are ON. Otherwise, M10~M13 are OFF.

Minimum value	Maximum value	Current value of C10	Output
D100=40	D101=100	40≦C10≦100	M10=ON
D102=120	D103=210	120≦C10≦210	M11=ON
D104=140	D105=170	140≦C10≦170	M12=ON
D106=150	D107=390	150≦C10≦390	M13=ON

5. Suppose the minimum value is larger than the maximum value. When the current value of C10 is less than the maximum value (C10 < 60), or when the current value of C10 is larger than the minimum value (C10 > 140), M12 is ON. Otherwise, M12 is OFF.

Minimum value	Maximum value	Current value of C10	Output
D100=40	D101=100	40≦C10≦100	M10=ON
D102=120	D103=210	120≦C10≦210	M11=ON
D104=140	D105=60	60≦C10≦140	M12=OFF
D106=150	D107=390	150≦C10≦390	M13=ON

Additional remark:

1. If **S**+2*__n__-1 used in the instruction ABSD exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S**+4*__n__-1 used in the instruction DABSD exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If **D**+__n__-1 used in the instruction ABSD exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If **D**+2*__n__-1 used in the instruction DABSD exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

5. If **n** used in the instruction ABSD is less than 1 or larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
0706		INCD		S_1 · S_2 · **n** · **D**	Incremental drum sequencer

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●							
S_2	●	●			●	●		●	●							
D		●	●	●				●								
n	●	●						●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D	●												
n		●			●	●							

Pulse instruction	16-Bit instruction	32-Bit instruction
-	AS	-

Symbol:

S_1 : Initial device in the comparison

S_2 : Counter number

D : Comparison result

n : Number of comparison groups

Explanation:

1. The instruction INCD is used to generate multiple pulses for a pair of counters.

2. The current value of S_2 is compared with the setting value in S_1. When the current value matches the setting value, the current value of S_2 is rest to 0, and the current comparison group number is stored in S_2+1.

3. After the comparison between the current values of S_2 and the **n** groups of values is complete, SM688 is ON for a scan cycle.

4. When the conditional contact is not enabled, the value in S_2 is 0, the value in S_2+1 is 0, **D**~**D**+**n**-1 are OFF, and SM688 is OFF.

5. When the on-line editing is used, please reset the conditional contact to initialize the instruction.

6. The operand **n** should be within the range between 1 and 256.

Example:

1. Before the instruction INCD is executed, the instruction MOV is used to write the setting values in D100~D104. The values in D100~D104 are 15, 30, 10, 40, and 25 respectively.

2. The current values of C10 is compared with the setting values in D100~D104. When the current value matches the setting value, C10 is rest to 0, and counts again.

3. The current comparison group number is stored in C11.

4. When the value in C11 changes by one, M10~M14 act correspondingly. Please refer to the timing diagram below.

5. When the comparison between the current values of C10 and the values in D100~D104 is complete, SM688 is ON for a scan cycle.

6. When X0.0 is switched from ON to OFF, C10 and C11 are reset to 0, and M10~M14 are switched OFF. When X0.0 is ON again, the execution of the instruction starts from the beginning.

Additional remark:

1. If $\mathbf{S_2}$+1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If $\mathbf{S_1}$+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If **n** is less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

5. If users declare the operand $\mathbf{S_2}$ in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

API	Instruction code			Operand	Function
0708	D	PIDE		**As shown on the list below**	PID algorithm

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
PID_RUN	●	●	●	●				●								
SV								●								○
PV								●								○
PID_MODE								●					○	○		
PID_MAN	●	●	●	●				●								
MOUT_AUTO	●	●	●	●				●								
CYCLE								●								
KC_Kp								●								
Ti_Ki								●								
Td_Kd								●								
Tf								●								
PID_EQ	●	●	●	●				●								
PID_DE	●	●	●	●				●								
PID_DIR	●	●	●	●				●								
ERR_DBW								●	●							○
MV_MAX								●	●							○
MV_MIN								●	●							○
MOUT								●								
BIAS								●	●							○
I_MV								●								
MV								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING

Pulse instruction	16-Bit instruction	32-Bit instruction
-	-	AS

Symbol:

EN	:	Enable/Disable the instruction
PID_RUN	:	Enabling the PID algorithm
SV	:	Target value (SV)
PV	:	Process value (PV)
PID_MODE	:	PID control mode
PID_MAN	:	PID Auto/Manual mode
MOUT_AUTO	:	Manual/Auto output value
CYCLE	:	Sampling time (CYCLE)
Kc_Kp	:	Proportional gain
Ti_Ki	:	Integral coefficient (sec. or 1/sec)
Td_Kd	:	Derivative coefficient (sec)
Tf	:	Derivate-action time constant (sec)
PID_EQ	:	PID formula types
PID_DE	:	The calculation of the PID derivative error
PID_DIR	:	PID forward/reverse direction (PID_DIR)
ERR_DBW	:	Range within which the error value is counted as 0
MV_MAX	:	Maximum output value (MV_MAX)
MV_MIN	:	Minimum output value (MV_MIN)
MOUT	:	Manual output value (MOUT)
BIAS	:	Feedforward output value
I_MV	:	Accumulated integral value
MV	:	Output value (MV)

Explanation:

1. The instruction is used to implement the PID algorithm. After the sampling time is reached, the PID algorithm is implemented. PID stands for Proportional, Integral, Derivative. The PID control is widely applied to mechanical equipment, pneumatic equipment, and electronic equipment.
2. The setting of the parameters is as follows.

Operand	Data type	Function	Setting range	Description
PID_RUN	BOOL	Enabling the PID algorithm	True: The PID algorithm is implemented. False: The output value (MV) is reset to 0, and the PID algorithm is not implemented.	
SV	REAL	SV	Range of single-precision floating-point numbers	Target value
PV	REAL	PV	Range of single-precision floating-point numbers	Process value
PID_MODE	DWORD/DINT	PID control mode	0: Automatic control When PID_MAN is switched from True to False, the output value (MV) then is involved in the automatic algorithm. 1: The parameters are tuned automatically for the temperature control. When the tuning of the parameters is complete, the device is automatically set to 0, and is filled in with appropriate parameters Kc_Kp, Ti_K_i, Td_K_d and Tf. Note: while using automatic control, users can not input the setting vaule manually.	

Operand	Data type	Function	Setting range	Description
PID_MAN	BOOL	PID A/M mode	True: Manual The MV is output according to the MOUT, but it is still within the range between the MV_MIN and the MV_MAX. When PID_MODE is set to 1, the setting is ineffective. False: Automatic The MV is output according to the PID algorithm, and the output value is within the range between MV_MIN and MV_MAX.	
MOUT_AUTO	BOOL	MOUT automatic change mode	True: Automatic The MOUT varies with the MV. False: Normal The MOUT deos not vary with the MV.	
CYCLE	DWORD/DINT	Sampling time (T_S)	1~40,000 (unit: ms)	When the instruction is scanned, the PID algorithm is implmented according to the sampling time, and the MV is refreshed. (The PLC need the instruction to execute; it will not run the sampling time automaticlly) If T_S is less than 1, it will be counted as 1. If T_S is larger than 40,000, it will be counted as 40,000. When the instruction PID is used in the interval interrupt task, the

Operand	Data type	Function	Setting range	Description
				sampling time is the same as the interval between the timed interrupt tasks. The setting of the sampling cycle is invalid here.
Kc_Kp	REAL	Calculated proportional coefficient (Kc or Kp, according to the settings in PID_EQ)	Range of positive single-precision floating-point numbers	Calculated proportional coefficient (Kc or Kp) If the P coefficient is less than 0, the Kc_Kp will be 0. Independently, if Kc_Kp is 0, it will not be controlled by P.
Ti_Ki	REAL	Integral coefficient (Ti or Ki, according to the settings in PID_EQ)	Range of positive single-precision floating-point numbers (unit: Ti = sec; Ki = 1/sec)	If the calculated coefficient I is less than 0, Ti_Ki will be 0. If Ti_Ki is 0, it will not be controlled by I.
Td_Kd	REAL	Derivative coefficient (Td or Kd, according to the settings in PID_EQ)	Range of positive single-precision floating-point numbers (unit: sec)	If the calculated coefficient D is less than 0, Td_Kd will be 0. If Ti_Ki is 0, it will not be controlled by D.

Operand	Data type	Function	Setting range	Description
Tf	REAL	Derivate-action time constant	Range of positive single-precision floating-point numbers (unit: sec)	If the derivate-action time constant is less than 0, Tf will be 0 and it will not be controlled by the derivate-action time constant. (Derivative Smoothing)
PID_EQ	BOOL	PID formula types	TRUE: Dependent Formula FALSE: Independent Formula	
PID_DE	BOOL	The calculation of the PID derivative error	TRUE: Using the variations in the PV to calculate the control value of the derivative (Derivative of the PV). FALSE: Using the variations in the error (E) to calculate the control value of the derivative (Derivative of the error).	
PID_DIR	BOOL	PID forward/reverse direction	True: Reverse action (E=SV-PV) False: Forward action (E=PV-SV)	
ERR_DBW	REAL	Range within which the error value is counted as 0.	Range of single-precision floating-point numbers	The error value (E) is the difference between the SV and the PV. When the setting value is 0, the function is not enabled; otherwise the CPU module will check whether the present error is less than the absolute value of ERR_DBW, and check whether the present error meets the cross status condition. If the present error is less than the absolute value

Operand	Data type	Function	Setting range	Description
				of ERR_DBW, and meets the cross status condition, the present error will be counted as 0, and the PID algorithm will be implemented, otherwise the present error will be brought into the PID algorithm according to the normal processing.
MV_MAX	REAL	Maximum output value	Range of single-precision floating-point numbers	Suppose **MV_MAX** is set to 1,000. When the MV is larger than 1,000, 1,000 is output. The value in **MV_MAX** should be larger than that in **MV_MIN**. Otherwise, the maximum MV and the minimum MV will be reversed.
MV_MIN	REAL	Minimum output value	Range of single-precision floating-point numbers	Suppose **MV_MIN** is set to -1,000. When the MV is less than -1,000, -1,000 is output.
MOUT	REAL	MV	Range of single-precision floating-point numbers	When set to PID Manual, the MV value will be outputted as the manually set MOUNT value, between MV_MAX and MV_MIN.

Operand	Data type	Function		Setting range	Description
BIAS	REAL	Feedforward output value		Range of single-precision floating-point numbers	Feedforward output value, used for the PID feedforward.
I_MV (occupies 15 consecutive DWordDevice)	REAL	I_MV	Accumulated integral value	Range of single-precision floating-point numbers	Accumulated integral value temporarily stored is usually for reference. Users can still clear or modify it according to their needs. When the MV is greater than the MV_MAX, or when the MV is less than MV_MIN, the accumulated integral value in I_MV is unchanged.
		I_MV+1	The previous error value is temporarily stored in it.		
		I_MV+2 ~ I_MV+5	For system use only		
		I_MV+6	The previous PV is temporarily stored in it.		
		I_MV+7 ~ I_MV+14	For system use only		
MV	REAL	MV	The MV is withing the range between the MV_MIN and the MV_MAX.		

The diagram of switching to PID_MAN / MOUT_AUTO:

1. When switching the control mode **(PID_MAN=0)** from automatic to manual, users can set the flag MOUT_AUTO to 1 and the output value of MOUT will go along with the output value of MV. After switching to the manual mode **(PID_MAN=1)**, users can set the MOUT_AUTO to 0.

2. When **PID_RUN** is changing from TRUE to FALSE, the PLC will reset the value in MV to 0. When the value in MV is to be retained, users can set the EN as FALSE to dismiss the instruction and to keep the output value in MV.

Example:

1. Before the instruction DPID is executed, the setting of the parameters should be complete.

2. When X0.0 is ON, the instruction is executed. When M1 is ON, the DPID algorithm is implemented. When M1 is OFF, the MV is 0, and the MV is stored in D200. When X0.0 is switched OFF, the instruction is not executed, and the previous data is unchanged.

Additional remark:

1. The instruction can be used several times, but the registers specified by **I_MV**~**I_MV**+14 cannot be the same.

2. **I_MV** occupies 30 registers. **I_MV** used in the instruction PID in the above example occupies D126~D155.

3. The instruction DPID only can be used in the cyclic task and the interval interrupt task. When the instruction DPID is used in the interval interrupt task, the sampling time (Cycle) is the same as the interval between the timed interrupt tasks.

4. When the instruction is scanned, the DPID algorithm is implmented according to the sampling time (Cycle), and the MV is refreshed. When the instruction is used in the interrupt task, the sampling time (Cycle) is the same as the interval between the timed interrupt tasks. The PID algorithm is implemented according to the interval between the timed interrupt tasks.

5. Before the DPID algorithm is implemented, the process value used in the instruction PID has to be a stable value. When users need the input value in the module to implement the DPID algorithm, they have to notice the time it takes for the analog input to be converted into the digital input.

6. When the PV (process value) is in the range of **ERR_DBW**, at the beginning, the present error will be brought into the PID algorithm according to the normal processing and then the CPU module will check whether the present error meets the cross status condition: PV (process value) goes beyond the SV (target value). Once the condition is met, the present error will be counted as 0 to implement the PID algorithm. And after the PV (process value) is out of the **ERR_DBW** range, the present error will be brought into the PID algorithm again. If PID_DE is true, that means using the variations in the PV to calculate the control value of the derivative and after the cross status condition is met, the PLC will treat Δ **PV** as 0 to implement the PID algorithm. (Δ **PV**= current **PV** – previous **PV**). As the example shown below, the present error will be brought into the PID algorithm according to the normal processing in the section A and the present error or Δ **PV** will be counted as 0 to implement the PID algorithm in the section B.

The PID algorithm:

1. When **PID_MODE** is set to 0, the PID control mode is the automatic control mode.

- **Independent Formula & Derivative of E** (**PID_EQ**=False & **PID_DE**=False)

$$MV = K_P E + Ki\int_0^t Edt + K_d * \frac{dE}{dt} + BIAS \quad E = SV - PV \text{ or } E = PV - SV$$

- **Independent Formula & Derivative of PV** (**PID_EQ=False & PID_DE=Ture**)

$$MV = K_P E + Ki\int_0^t Edt - K_d * \frac{dPV}{dt} + BIAS \quad E = SV - PV$$

Or

$$MV = K_P E + Ki\int_0^t Edt + K_d * \frac{dPV}{dt} + BIAS \quad E = PV - SV$$

- **Dependent Formula & Derivative of E** (**PID_EQ**=True & **PID_DE**=False)

$$MV = K_c\left[E + \frac{1}{T_i}\int_0^t Edt + T_d * \frac{dE}{dt}\right] + BIAS \quad E = SV - PV \text{ or } E = PV - SV$$

- **Dependent Formula & Derivative of PV** (**PID_EQ**=True & **PID_DE**=True)

$$MV = K_c\left[E + \frac{1}{T_i}\int_0^t Edt - T_d * \frac{dE}{dt}\right] + BIAS \quad E = SV - PV$$

Or

$$MV = K_c\left[E + \frac{1}{T_i}\int_0^t Edt + T_d * \frac{dE}{dt}\right] + BIAS \quad E = PV - SV$$

2. When **PID_MODE** is set to 1, the PID control mode is the automatic tuning mode. After the tuning of the parameter is complete, **PID_MODE** is set to 0. The PID control mode becomes the automatic control mode.

PID Block Diagram:

PID Block Diagram (Independent)

PID Block Diagram (Dependent)

Suggestion:

1. Owing to the fact that the instruction DPID can be used in a lot of controlled environments, users have to choose the control function appropriately. For example, to prevent the improper control from occurring, **PID_MODE** can not be used in the motor controlled environment when it is set to 1.

2. When users tune the parameters **Kc_Kp**, **Ti_Ki**, and **Td_Kd** (PID_MODE is set to 0), they have to tune the KP first (according to the experience), and then set the **Ti_Ki** and the **Td_Kd** to 0. When users can handle the control, they can increase the **Ti_Ki** and the **Td_Kd**. When the **Kc_Kp** is 1, it means that the proportional gain is 100%. That is, the error value is increased by a factor of one. When the proportional gain is less than 100%, the error value is decreased. When the proportional gain is larger than 100%, the error value is increased.

3. To prevent the parameters which have been tuned automatically from disappearing after a power cut, users have to store the parameters in the latched data registers when is **PID_MODE** set to 1. The parameters which have been tuned automatically are not necessarily suitable for every controlled environment. Therefore, users can modify the parameters which have been tuned automatically. However, it is suggested that users only modify the **Ti_Ki** and the **Td_Kd.**

4. The instruction should be used with many parameters. To prevent the improper control from occurring, please do not set the parameters randomly.

Example 1: The steps of tuning the parameters used with the instruction PID

Suppose that the transfer function of the plant is the first-order function $G(s) = \frac{b}{s+a}$, the SV is 1, the sampling time Ts is 10 milliseconds. It is suggested that the steps of tuning the parameters are as follows.

Step 1: First, set the K_I and the K_D to 0. Next, set the K_P to 5, 10, 20 and 40 successively, and record the target values and the process values. The results are shown in the diagram below.

Step 2: When the K_P is 40, there is overreaction. Thus, the K_P is not chosen. When the K_P is 20, the reaction curve of the PV is close to the SV, and there is no overreaction. However, due to the fast start-up, the transient output value (MV) is big. The K_P is not chosen, either. When the K_P is 10, the reaction curve of the PV approaches the SV smoothly. Therefore, the K_P is chosen. When the K_P is 5, the reaction is slow. Thus, the K_P is not chosen.

Step 3: After the K_P is set to 10, increase the K_I. For example, the K_I is set to 1, 2, 4, and 8 successively. The K_I should not be larger than the K_P. Then, increase the K_D. For example, the K_D is set to 0.01, 0.05, 0.1, and 0.2 successively. The K_D should not be larger than ten percent of the K_P. Finally, the relation between the PV and the SV is present in the following diagram.

Note: The example is only for reference. Users have to tune the parameters properly according to the practical condition of the control system.

Example 2: Using the automatic tuning function to control the temperature

Purpose: Using the automatic tuning function to calcaulte the most appropriate parameters for the PID temperature control

Explanation:

Due to the fact that users may not be familiar with the characteristics of the temperature environment which is controlled for the first time, they can use the automatic tuning function to make an initial adjustment (**PID_MODE** is set to 1). After the tuning of the parameter is complete, **PID_MODE** is set to 0. The controlled environment in this sample is an oven. The program example is as below.

The experimental result of the automatic tuning function is shown below.

The experimental result of using the parameters which have been tuned to control the temperature is shown below.

As the diagam above shows, after the parameters are tuned automatically, users can get a good temperature control result. It only takes about twenty minutes to control the temperature. When the target temperature changes from 80°C to 100°C, the result is as below.

As the diagam above shows, when the target temperature changes from 80°C to 100°C, the parameters tuned previously still can be used to control the temperature. Besides, it does not take much time to control the temperature.

API	Instruction code			Operand	Function
0709		XCMP		S_1 · S_2 · S_3 · S_4 · D	Setups for comparing the inputs of multiple work stations

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2							○									
S_3								○								
S_4								○								
D								○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2												●	
S_3		●			●	●							
S_4			●				●						
D			●				●						

Pulse instruction	16-Bit instruction	32-Bit instruction
-	AS	-

Symbol:

S_1 : Trigger input point

S_2 : High-speed counter number

S_3 : Setting of the numbers for work station and objects

S_4 : Reference value for comparison and the observational error

D : Initial corresponding device for the comparison result in the stack area

Explanation:

1. This instruction is only applicable for AS series with firmware version 1.04 or higher.

2. The operand **S_1** is for the setting of the trigger input points; high-speed inputs are X0.0~X0.15 and for inputs other than X0.0~X0.15, they are general type. Once the instruction is executed, the external interrupts for the inputs (X0.0~X0.15) will be enabled. Therefore it is not suggested to use the inputs with interrupt tasks; otherwise, when the instruction is executed, the interrupts will be disabled, and only after the instruction is done, the interrupt tasks will be resumed. For the general type, they will be affected by the scan time though they are suitable for the environments where the inputs are not as stable.

3. The operand **S_2** works with 32-bit counters (HC0~HC255). When the inputs are high-speed trigger input type, it is suggested to implement the hardware high-speed counter and users can use the instruction DCNT to enable the counter. When it is required to use high-speed outputting, users can use the instruction DMOV to copy the output current position, for example copying the axis of SR460 to HC0, (DMOV SR460 HC0).

4. The operand **S_3** occupies a consecutive three 16-bit devices. **S_3**+0 is n the setting of the work station number and **S_3**+1 is m the maximum object number. **S_3**+2 is the result of the object being filtered. The range for n and m is 1~32. When the value is out of range, the value will be seen as the maximum (32) or the minimum (1). The range for **S_3**+2 (the number of filter) is 0~32767. Value less than 0 will be seen as 0 and when the value is 0, the filtering function is disabled. It is suggested to declare an array of 3 words or 3 consecutive word type variables.

5. It is suggested to set the maximum number for **S_3**+1 (m). If m<n, users should take notice of the objects and make sure they are sufficient on the production line.

6. The operand **S_4** occupies a consecutive of 3xn 32-bit devices (6xn 16-bit devices). If the space taken exceeds the range of device D, the instruction will not be executed. The n is the work station number set in the operand **S_3**. The functions for each device and the corresponding number for **S_4** are listed below. It is suggested to declare an array of 3n double words or 3 consecutive double word type variables.

Function	Work station 1	Work station 2	・・・	Work station n
Reference value for comparison (32-bit)	**S_4**+0	**S_4**+2	・・・	**S_4**+(n-1)x2
Observational error when entering (32-bit)	**S_4**+2xn	**S_4**+2xn+2	・・・	**S_4**+(2xn-1)x2
Observational error when leaving (32-bit)	**S_4**+4xn	**S_4**+4xn+2	・・・	**S_4**+(4xn-1)x2

When the reference value is set to 0 for a specific work station, the specific work station will stop working. Users can use this technique to manage the work station.

7. The operand **D** is initial corresponding device for the comparison result in the stack area. The operand **D** occupies a consecutive of 2xn 16-bit devices and 2xmxn 32-bit devices (or 4xmxn 16-bit devices). If the space taken exceeds the range of device D, the instruction will not be executed. The functions for each device and the corresponding number for **D** are listed below.

Function	Work station 1	Work station 2	・・・	Work station n
Value of the head index (16-bit)	**D**+0	**D**+1	・・・	**D**+(n-1)
Value of the tail index (16-bit)	**D**+n	**D**+(n+1)	・・・	**D**+(2xn-1)
Compared counter result 1 of the object when entering (32-bit)	**D**+2xn	**D**+2xn+2	・・・	**D**+2xn+2(n-1)
Compared counter result 1 of the	**D**+4xn	**D**+4xn+2	・・・	**D**+4xn+2(n-1)

Function	Work station 1	Work station 2	・・・	Work station n
object when leaving (32-bit)				
:	:	:	:	:
Compared counter result m of the object when entering (32-bit)	**D**+4xmxn-2xn		・・・	**D**+4xmxn-2
Compared counter result m of the object when leaving (32-bit)	**D**+4xmxn		・・・	**D**+4xmxn+2 (n-1)

It tends to occupy more space in the stack area. If the space taken exceeds the range of device D, the PLC will only execute what is valid in the storage and no warning will be shown. It is suggested to declare an array of 2xn+4xmxn words.

8. There is no limit on the number of times the instruction can be executed but only one execution can be done at a time.

9. This instruction is suggested to use with the instruction API0710 YOUT and use the same initial corresponding device for the comparison result in the stack area (the operand **D**).

10. The timing diagram of executing the high-speed counter and filter (read from right to left).

*1. PLC reads the current counter value.

*2. Drop the counter value: the number of filters read is less than the number of filters set.

*3. Record the counter value: the signal is high (on time) and records the counter value to the comparing stack area for entering.

*4. Record the counter value: the signal is low (off time) and records the counter value to the comparing stack area for leaving.

11. When the signal is rising-/falling-edge triggered and complete processing filter, the PLC reads the high-speed counter value and adds one in the value of the head index. The PLC will record the entering and leaving counter results of each work station. The compared counter result is the current counter value + reference value + observational error. Whether it is rising-triggered or falling-edge triggered, the value of the head index will be accumulated. The maximum head index value is mx2 (the maximum number of object).

12. The value of the head index will be cyclically accumulated, when the signal is rising-/falling-edge triggered and complete with processing the number of filters. (the default for trigger input is OFF) The maximum value of the head index is mx2 (the maximum number of object). For example, the number of object is set to 10, the value of the head index (default: 0) will be added to 1, 2, 3 to 20 and then 1, 2, 3 to 20 repeatedly. When the value of the head index is set to 0, it means there is no object entered after the instruction is executed. After adding one to the value of the head index, the PLC will check the value in the tail index. If the value (after adding one) in the value of the head index equals to the value of the tail index, the addition will be cancelled and the counter result will be recorded.

13. When the instruction is executed and the state of the initial input is **OFF**, the **rising-edge trigger** will correspond to the **odd numbers** of the head index value, and the ***falling-edge trigger*** will correspond to the ***even numbers*** of the head index value.

14. When the instruction is executed and the state of the initial input is **ON**, the **falling-edge trigger** will correspond to the **odd numbers** of the head index value, and the ***rising-edge trigger*** will correspond to the ***even numbers*** of the head index value.

15. When the instruction is executed, values in the accumulated area and the index areas will not be cleared. If the data is in the latched area and needed to be enabled again, users need to use the instruction ZRST to clear the values in the head and tail indexes.

Example:

Please refer to the example in the API0710 YOUT for more information.

API	Instruction code			Operand	Function
0710		YOUT		S_1 · S_2 · S_3 · D	Comparing the outputs of multiple work stations

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								○								
S_3								○								
D		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2		●			●	●							
S_3			●				●						
D	●												

Pulse instruction	16-Bit instruction	32-Bit instruction
-	AS	-

Symbol:

S_1 : High-speed counter number

S_2 : Setting of the numbers for work station and objects

S_3 : Initial corresponding device for the comparison result in the stack area

D : Initial corresponding device for the output work station

Explanation:

1. This instruction is only applicable for AS series with firmware version 1.04 or higher.

2. The operand **S_1** is for the setting of the high-speed counter. The high-speed counter should be set the same as the high-speed counter set from the API0709 XCMP.

3. The operand **S_2** occupies a consecutive two 16-bit devices. **S_2**+0 is n the setting of the work station number and **S_2**+1 is m the maximum object number. The range for n and m is 1~32. When the value is out of range, the value will be seen as the maximum (32) or the minimum (1). The settings here should be the same as the API0709 XCMP.

4. The operand **S_3** is initial corresponding device for the comparison result in the stack area. The operand **S_3** occupies a consecutive of 2xn 16-bit devices and 2xmxn 32-bit devices (or 4xmxn 16-bit devices). For the functions of each device and the corresponding number for **D**, please refer to the API0709 XCMP. It is suggested to use the same variable as the API0709 XCMP does.

5. There is no limit on the number of times the instruction can be executed but only one execution can be done at a time.

6. This instruction is suggested to use with the API0709 XCMP and use the same initial corresponding device for the comparison result in the stack area (the operand **S_3**).

7. The operand **D** is only for the outputs of Y as well as M devices and BOOL data type. It occupies a consecutive of work stations xn. When it is used as the output point or the M device, the instruction will run to refresh the output states.

8. The corresponding values for the odd numbered head index (for example 1, 3, 5,…) are called compared counter result of the object when entering. The corresponding values for the even numbered head index (for example 2, 4, 6,…) are called compared counter result of the object when leaving.

9. When the compared counter result of entering and leaving in the stack area are 0, the actions in this area will not be executed and the state of the corresponding output work station will be OFF. Adding 2 to the value in the tail index but the value (after adding 2) will not exceed the value of the head index.

10. When the instruction YOUT is executed, each work station will check the compared value of entering and leaving in the tail index. When the counter value is larger or the same as the compared value of entering, the corresponding output point will be ON and will add 1 to the value of the tail index. When the counter value is larger or the same as the compared value of leaving, the corresponding output will be OFF and will add 1 to the value of the tail index but the value (after adding 1) will not exceed the value of the head index.

Example: the example of work station x3 and the number of objects can be up to 4.

Step 1: using the input point X0.4 as the object detection interrupts, HC202 is the high-speed counter for the encoder and the output point Y0.0 as the initial output point for the work station to output.

Step 2: using the register to edit, set up the reference value, the observational error when entering and leaving.

Device D	D500	D502	D504
Reference value for comparison (32-bit)	K2000	K3000	K4000
Device D	D506	D508	D510

Observational error when entering (32-bit)	K100	K120	K130
Device D	D512	D514	D516
Observational error when leaving (32-bit)	K50	K-20	K20
Device D	D2000	D2001	D2002
Value of the head index (16-bit)	K0	K0	K0
Device D	D2003	D2004	D2005
Value of the tail index (16-bit)	K0	K0	K0

Step 3: Set up the initial values and write the programs.

Set up 3 work stations for D0, 4 objects for D1 and 50 filters for D2. After the contact M0 is activated, settings of the object detection, compared value setup, the compared counter result of the object entering as well as leaving, and the output controls for each work station can be done. For instance the system detects 2 objects entered and 4 triggers to read the compared counter results, 3000, 3500, 4500, and 5000 in HC202 (HC202=K5060). The compared value and the head/tail index in the stack area are as below.

Device D	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K1	K1	K1
Device D number	D2006	D2008	D2010
Compared counter result 1 of the object when entering (32-bit)	K5100	K6120	K7130
Device D number	D2012	D2014	D2016
Compared counter result 1 of the object when leaving (32-bit)	K5550	K6480	K7520
Device D number	D2018	D2020	D2022
Compared counter result 2 of the object when entering (32-bit)	K6600	K7620	K8630
Device D number	D2024	D2026	D2028
Compared counter result 2 of the object	K7050	K7980	K9020

when leaving (32-bit)			
Device D number	D2030	D2032	D2034
Compared counter result 3 of the object when entering (32-bit)	K0	K0	K0
Device D number	D2036	D2038	D2040
Compared counter result 3 of the object when leaving (32-bit)	K0	K0	K0

When the high-speed counter HC202 reaching 5200, the state of the output point Y is as below:

Output point Y number	Y0.0	Y0.1	Y0.2
16-bit value	ON	OFF	OFF
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K2	K1	K1

When the high-speed counter HC202 reaching 6200, the state of the output point Y is as below:

Output point Y number	Y0.0	Y0.1	Y0.2
16-bit value	OFF	ON	OFF
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K3	K2	K1

When the high-speed counter HC202 reaching 6800, the state of the output point Y is as below:

Output point Y number	Y0.0	Y0.1	Y0.2
16-bit value	ON	OFF	OFF
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K4	K3	K1

When the high-speed counter HC202 reaching 7300, the state of the output point Y is as below:

Output point Y number	Y0.0	Y0.1	Y0.2
16-bit value	OFF	OFF	ON
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K4	K3	K2

When the high-speed counter HC202 reaching 7700, the state of the output point Y is as below:

Output point Y number	Y0.0	Y0.1	Y0.2
16-bit value	OFF	ON	OFF
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K4	K4	K3

When the high-speed counter HC202 reaching 8000, the state of the output point Y is as below:

Output point Y number	Y0.0	Y0.1	Y0.2
Output state	OFF	OFF	OFF
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K4	K4	K3

When the high-speed counter HC202 reaching 8700, the state of the output point Y is as below:

Output point Y number	Y0.0	Y0.1	Y0.2
Output state	OFF	OFF	ON
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K4	K4	K4

6.9 Logic Instructions

6.9.1 List of Logic Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0800	WAND	DAND	✓	Logical AND operation
0801	MAND	–	✓	Matrix AND operation
0802	WOR	DOR	✓	Logical OR operation
0803	MOR	–	✓	Matrix OR operation
0804	WXOR	DXOR	✓	Logical exclusive OR operation
0805	MXOR	–	✓	Matrix exclusive OR operation
0808	WINV	DINV	✓	Logical reversed INV operation
0809	LD&	DLD&	–	S_1&S_2
0810	LD|	DLD|	–	S_1|S_2
0811	LD^	DLD^	–	S_1^S_2
0812	AND&	DAND&	–	S_1&S_2
0813	AND|	DAND|	–	S_1|S_2
0814	AND^	DAND^	–	S_1^S_2
0815	OR&	DOR&	–	S_1&S_2
0816	OR|	DOR|	–	S_1|S_2
0817	OR^	DOR^	–	S_1^S_2

6.9.2 Explanation of Logic Instructions

API	Instruction code			Operand	Function
0800	D	WAND	P	S_1 · S_2 · **D**	Logical AND operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Data source 1

S_2 : Data source 2

D : Operation result

Explanation:

1. The logical operator AND takes the binary representations in S_1 and S_2, and performs the logical AND operation on each pair of corresponding bits. The operation result is stored in **D**.

2. Only the instruction DAND can use the 32-bit counter.

3. The result in each position is 1 if the first bit is 1 and the second bit is 1. Otherwise, the result is 0.

Example 1:

When X0.0 is ON, the logical operator AND takes the data in the 16-bit device Y0 and the 16-bit device Y2, and performs the logical AND operation on each pair of corresponding bits. The operation result is stored in Y4.

Example 2:

When X0.0 is ON, the logical operator AND takes the data in the 32-bit device (Y11, Y10) and the 32-bit device (Y21, Y20), and performs the logical AND operation on each pair of corresponding bits. The operation result is stored in (Y41, Y40).

API	Instruction code			Operand	Function
0801		MAND	P	S_1 · S_2 · **D** · **n**	Matrix AND operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●							
S_2	●	●			●	●		●	●							
D		●			●	●		●								
n	●	●			●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Matrix source 1

S_2 : Matrix source 2

D : Operation result

n : Length of the array

Explanation:

1. The operator AND takes the **n** rows of binary representations in S_1 and the **n** rows of binary representations in S_2, and performs the matrix AND operation on each pair of corresponding bits. The operation result is stored in **D**.

2. The result in each position is 1 if the first bit is 1 and the second bit is 1. Otherwise, the result is 0.

3. The operand **n** should be within the range between 1 and 256.

Example:

When X0.0 is ON, the operator AND takes the data in the 16-bit devices Y0~Y2 and the data in 16-bit devices Y10~Y12, and performs the matrix AND operation on each pair of corresponding bits. The operation result is stored in the 16-bit devices Y20~Y22.

Additional remark:

1. If **S_1+n-1**, **S_2+n-1**, or **D+n-1** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. Explanation of matrix instructions:

   - A matrix is composed of more than one 16-bit register. The number of registers in a matrix is the length of the array **n**. There are 16×**n** bits in a matrix, and the matrix operation is performed on one bit at a time.

   - The matrix instruction takes the 16×**n** bits in a matrix as a string of bits, rather than takes them as values. The matrix operation is performed on one specified bit.

   - The matrix instruction mainly processes the one-to-many status or the many-to-many status, such as the moving, the copying, the comparing, and the searching. It is a handy and important applied instruction.

   - When the matrix instruction is executed, users need a 16-bit register to specify a certain bit among the 16**n** bits in the matrix for the operation. The 16-bit register is called the pointer, and is specified by users. The value in the register is within the range between 0 and 16**n**-1, and corresponds to the bit within the range between b0 and b16**n**-1.

- The shift of the specified data, or the rotation of the specified data can be involved in the matrix operation. Besides, the bit number decreases from the left to the right, as illustrated below.

- The width of the matrix (C) is 16 bits.

- Pr represents the pointer. When the value in Pr is 15, b15 is specified.

Example: The following matrix is composed of the three 16-bit devices Y0, Y1, and Y2. The data in Y0 is 16#AAAA, the data in Y1 is 16#5555, and the data in Y2 is 16#AAFF.

C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	Y0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	Y1
1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	Y2

Example: The following matrix is composed of the three 16-bit devices X 0, X 1, and X 2. The data in X 0 is 16#37, the data in X 1 is 16#68, and the data in X 2 is 16#45.

C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	
0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	X0
0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	X1
0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	X2

API	Instruction code			Operand	Function
0802	D	WOR	P	S_1 · S_2 · **D**	Logical OR operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Data source 1

S_2 : Data source 2

D : Operation result

Explanation:

1. The logical operator OR takes the binary representations in S_1 and S_2, and performs the logical inclusive OR operation on each pair of corresponding bits. The operation result is stored in **D**.

2. Only the instruction DOR can use the 32-bit counter but not the device E.

3. The result in each position is 1 if the first bit is 1, the second bit is 1, or both bits are 1. Otherwise, the result is 0.

Example 1:

When X0.0 is ON, the logical operator OR takes the data in the 16-bit device Y0 and the 16-bit device Y2, and performs the logical inclusive OR operation on each pair of corresponding bits. The operation result is stored in Y4.

Example 2:

When X0.1 is ON, the logical operator OR takes the data in the 32-bit device (Y11, Y10) and the 32-bit device (Y21, Y20), and performs the logical inclusive OR operation on each pair of corresponding bits. The operation result is stored in (Y41, Y40).

API	Instruction code			Operand	Function
0803		MOR	P	S_1 · S_2 · **D** · **n**	Matrix OR operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●							
S_2	●	●			●	●		●	●							
D		●			●	●		●								
n	●	●			●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Matrix source 1

S_2 : Matrix source 2

D : Operation result

n : Length of the array

Explanation:

1. The operator OR takes the **n** rows of binary representations in **S_1** and the **n** rows of binary representations in **S_2**, and performs the matrix OR operation on each pair of corresponding bits. The operation result is stored in **D**.

2. The result in each position is 1 if the first bit is 1, the second bit is 1, or both bits are 1. Otherwise, the result is 0.

3. The operand **n** should be within the range between 1 and 256.

Example:

When X0.0 is ON, the operator OR takes the data in the 16-bit devices Y0~Y2 and the data in 16-bit devices Y10~Y12, and performs the matrix OR operation on each pair of corresponding bits. The operation result is stored in the 16-bit devices Y20~Y22.

Additional remark:

1. If **S_1+n-1**, **S_2+n-1**, or **D+n-1** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
0804	D	WXOR	P	S_1 · S_2 · **D**	Logical exclusive OR operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Data source 1

S_2 : Data source 2

D : Operation result

Explanation:

1. The logical operator XOR takes the binary representations in **S_1** and **S_2**, and performs the logical exclusive OR operation on each pair of corresponding bits. The operation result is stored in **D**.

2. Only the instruction DXOR can use the 32-bit counter, but not the device E.

3. The result in each position is 1 if the two bits are different, and 0 if they are the same.

Example 1:

When X0.0 is ON, the logical operator XOR takes the data in the 16-bit device Y0 and the 16-bit device Y2, and performs the exclusive OR operation on each pair of corresponding bits. The operation result is stored in Y4.

Example 2:

When X0.1 is ON, the logical operator XOR takes the data in the 32-bit device (Y11, Y10) and the 32-bit device (Y21, Y20), and performs the logical exclusive OR operation on each pair of corresponding bits. The operation result is stored in (Y41, Y40).

API	Instruction code			Operand	Function
0805		MXOR	P	**S_1 · S_2 · D · n**	Matrix exclusive OR operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●							
S_2	●	●			●	●		●	●							
D		●			●	●		●								
n	●	●			●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Matrix source 1

S_2 : Matrix source 2

D : Operation result

n : Length of the array

Explanation:

1. The operator XOR takes the **n** rows of binary representations in **S_1** and the **n** rows of binary representations in of **S_2**, and performs the matrix exclusive OR operation on each pair of corresponding bits. The operation result is stored in **D**.

2. The result in each position is 1 if the two bits are different, and 0 if they are the same.

3. The operand **n** should be within the range between 1 and 256.

Example:

When X0.0 is ON, the operator XOR takes the data in the 16-bit devices Y0~Y2 and the data in 16-bit devices Y10~Y12, and performs the matrix exclusive OR operation on each pair of corresponding bits. The operation result is stored in the 16-bit devices Y20~Y22.

Additional remark:

1. If **S_1+n-1**, **S_2+n-1**, or **D+n-1** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
0808	D	WINV	P	**S**・ **D**	Logical reversed INV operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○	○	○	○		
D	●	●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Data source

D : Operation result

6

Explanation:

1. Use the data in **S** to execute the INV instruction. The operation result is stored in **D**.

2. Only the instruction DINV can use the 32-bit counter but not the device E.

3. When an INV instruction is executed, reverse processing is performed. If the state is 0 before the INV instruction is executed, the state will change to 1 as a result of the INV instruction.

Example 1:

When X0.0 is ON, the operator WINV takes data in the 16-bit device Y0 and performs the INV operation on the corresponding bits. The operation result is stored in the 16-bit device Y4.

Example 2:

When X0.0 is ON, the operator DINV takes data in the 32-bit devices Y11~Y10 and performs the INV operation on each pair of corresponding bits. The operation result is stored in the 32-bit device Y41~Y40.

API	Instruction code			Operand	Function
0809~ 0811	D	LD #		S_1 , S_2	Contact type of logical operation LD #

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol:

Taking LD& and DLD& for example

S_1 : Data source 1

S_2 : Data source 2

Explanation:

1. The instruction is used to compare the data in S_1 with that in S_2. When the comparison result is not 0, the condition of the instruction is met. When the comparison result is 0, the condition of the instruction is not met.

2. Only the instruction DLD # can use the 32-bit counter but not the device E.

3. The instruction LD # can be connected to the mother line directly.

API No.	16-bit instruction	32-bit instruction	Comparison operation result	
			ON	OFF
0809	LD&	**D**LD&	S_1&$S_2 \neq 0$	S_1&$S_2 = 0$
0810	LD|	**D**LD|	S_1|$S_2 \neq 0$	S_1|$S_2 = 0$
0811	LD^	**D**LD^	S_1^$S_2 \neq 0$	S_1^$S_2 = 0$

4. &: Logical AND operation

5. |: Logical OR operation

6. ^: Logical exclusive OR operation

Example:

1. The logical operator AND takes the data in C0 and C1, and performs the logical AND operation on each pair of corresponding bits. When the operation result is not 0, Y1.0 is ON.

2. The logical operator OR takes the data in D200 and D300, and performs the logical OR operation on each pair of corresponding bits. When the operation result is not 0 and X1.0 is ON, Y1.1 is ON.

3. The logical operator XOR takes the data in C201 and C200, and performs the logical exclusive OR operation on each pair of corresponding bits. When the operation result is not 0, or when X1.1 is ON, Y1.2 is ON.

Additional remark:

If $\mathbf{S_1}$ or $\mathbf{S_2}$ is illegal, the condition of the instruction is not met, SM0 is ON, and the error in SR0 is 16#2003.

API	Instruction code			Operand	Function
0812~ 0814	D	AND #		S_1 · S_2	Contact type of logical operation AND #

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol:

Taking AND& and DAND& for example

S_1 : Data source 1

S_2 : Data source 2

Explanation:

1. The instruction is used to compare the data in S_1 with that in S_2. When the comparison result is not 0, the condition of the instruction is met. When the comparison result is 0, the condition of the instruction is not met.

2. Only the instruction DAND # can use the 32-bit counter, but not the device E.

3. The instruction AND # and the contact are connected is series.

API No.	16-bit instruction	32-bit instruction	Comparison operation result	
			ON	OFF
0812	AND&	**D**AND&	S_1&$S_2 \neq 0$	S_1&$S_2 = 0$
0813	AND|	**D**AND|	S_1|$S \neq 0$	S_1|$S_2 = 0$
0814	AND^	**D**AND^	S_1^$S_2 \neq 0$	S_1^$S = 0$

4. &: Logical AND operation

5. |: Logical OR operation

6. ^: Logical exclusive OR operation

Example:

1. When X0.0 is ON, the logical operator AND takes the data in C0 and C10, and performs the logical AND operation on each pair of corresponding bits. When the operation result is not 0, Y1.0 is ON.

2. When X0.1 is OFF, the logical operator OR takes the data in D10 and D0, and performs the logical OR operation on each pair of corresponding bits. When the operation result is not 0, Y1.1 keeps ON.

3. When X0.2 is ON, the logical operator XOR takes the data in the 32-bit register (D200, D201) and the data in the 32-bit register (D100, D101), and performs the logical exclusive OR operation on each pair of corresponding bits. When the operation result is not 0, or when X0.3 is ON, Y1.2 is ON.

Additional remark:

If $\mathbf{S_1}$ or $\mathbf{S_2}$ is illegal, the condition of the instruction is not met, SM0 is ON, and the error in SR0 is 16#2003.

API	Instruction code			Operand	Function
0815~ 0817	D	OR #		S_1 , S_2	Contact type of logical operation OR #

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol:

Taking OR& and DOR& for example

S_1 : Data source 1

S_2 : Data source 2

Explanation:

1. The instruction is used to compare the data in S_1 with that in S_2. When the comparison result is not 0, the condition of the instruction is met. When the comparison result is 0, the condition of the instruction is not met.

2. Only the instruction DOR # can use the 32-bit counter.

3. The instruction OR # and the contact are connected in parallel.

API No.	16-bit instruction	32-bit instruction	Comparison operation result	
			ON	OFF
0815	OR&	**D**OR&	S_1&$S_2 \neq 0$	S_1&$S_2 = 0$
0816	OR|	**D**OR|	S_1|$S_2 \neq 0$	S_1|$S_2 = 0$
0817	OR^	**D**OR^	S_1^$S_2 \neq 0$	S_1^$S_2 = 0$

4. &: Logical AND operation

5. |: Logical OR operation

6. ^: Logical exclusive OR operation

Example:

1. When X0.1 is ON, Y0.0 is ON. Besides, when the logical operator AND performs the logical AND operation on each pair of corresponding bits in C0 and C10 and the operation result is not 0, Y0.0 is ON.

2. When X0.2 and X0.3 are ON, Y0.1 is ON. When the logical operator OR performs the logical OR operation on each pair of corresponding bits in the 32-bit register (D10, D11) and the 32-bit register (D20, D21) and the operation result is not 0, Y0.1 is ON. Besides, when the logical operator XOR performs the logical exclusive OR operation on each pair of corresponding bits in the 32-bit counter HC0 and the 32-bit register (D200, D201) and the operation result is not 0, Y0.1 is ON.

Additional remark:

If S_1 or S_2 is illegal, the condition of the instruction is not met, SM0 is ON, and the error in SR0 is 16#2003.

6.10 Rotation Instructions

6.10.1 List of Rotation Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0900	ROR	DROR	✓	Rotating to the right
0901	RCR	DRCR	✓	Rotating to the right with the carry flag
0902	ROL	DROL	✓	Rotating to the left
0903	RCL	DRCL	✓	Rotating to the left with the carry flag
0904	MBR	–	✓	Rotating the matrix bits

6.10.2 Explanation of Rotation Instructions

API	Instruction code			Operand	Function
0900	D	ROR	P	**D · n**	Rotating to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●	●	●			○	○				
n	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

D : Device which is rotated

n : Number of bits forming a group

Explanation:

1. The values of the bits in the device specified by **D** are divided into groups (**n** bits as a group), and these groups are rotated to the right.

2. Only the instruction DROR can use the 32-bit counter, but not the device E.

3. The operand **n** used in the 16-bit instruction should be within the range between 1 and 16. The operand **n** used in the 32-bit instruction should be within the range between 1 and 32. When n is less than 0, the instruction will not be executed; when **n** exceeds the range, it will be seen as the maximum value (32) of the range, and the instruction will be executed.

4. Generally, the pulse instructions RORP and DRORP are used.

Example:

When X0.0 is switched from OFF to ON, the values of the bits in D10 are divided into groups (four bits as a group), and these groups are rotated to the right. (The value of the bit marked ※ is transmitted to the carry flag SM602.)

Additional remark:

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0901	D	RCR	P	**D · n**	Rotating to the right with the carry flag

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●	●	●			○	○				
n	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

D : Device which is rotated

n : Number of bits forming a group

Explanation:

1. The values of the bits in the device specified by **D** are divided into groups (**n** bits as a group), and these groups are rotated to the right with the carry flag SM602.

2. Only the instruction DRCR can use the 32-bit counter, but not the device E.

3. The operand **n** used in the 16-bit instruction should be within the range between 1 and 16. The operand **n** used in the 32-bit instruction should be within the range between 1 and 32. When n is less than 0, the instruction will not be executed; when **n** exceeds the range, it will be seen as the maximum value (32) of the range, and the instruction will be executed.

4. Generally, the pulse instructions RCRP and DRCRP are used.

Example:

When X0.0 is switched from OFF to ON, the values of the bits in D10 are divided into groups (four bits as a group), and these groups are rotated to the right with the carry flag SM602. (The value of the bit marked ※ is transmitted to the carry flag SM602.)

Additional remark:

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0902	D	ROL	P	**D · n**	Rotating to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●	●	●			○	○				
n	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

D : Device which is rotated

n : Number of bits forming a group

Explanation:

1. The values of the bits in the device specified by **D** are divided into groups (**n** bits as a group), and these groups are rotated to the left.

2. Only the instruction DROL can use the 32-bit counter, but not the device E.

3. The operand **n** used in the 16-bit instruction should be within the range between 1 and 16. The operand **n** used in the 32-bit instruction should be within the range between 1 and 32. When n is less than 0, the instruction will not be executed; when **n** exceeds the range, it will be seen as the maximum value (32) of the range, and the instruction will be executed.

4. Generally, the pulse instructions ROLP and DROLP are used.

Example:

When X0.0 is switched from OFF to ON, the values of the bits in D10 are divided into groups (four bits as a group), and these groups are rotated to the left. (The value of the bit marked ※ is transmitted to the carry flag SM602.)

Additional remark:

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0903	D	RCL	P	**D · n**	Rotating to the left with the carry flag

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●	●	●			○	○				
n	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

D : Device which is rotated

n : Number of bits forming a group

Explanation:

1. The values of the bits in the device specified by **D** are divided into groups (**n** bits as a group), and these groups are rotated to the left with the carry flag SM602.
2. Only the instruction DRCL can use the 32-bit counter, but not the device E.
3. The operand **n** used in the 16-bit instruction should be within the range between 1 and 16. The operand **n** used in the 32-bit instruction should be within the range between 1 and 32. When n is less than 0, the instruction will not be executed; when **n** exceeds the range, it will be seen as the maximum value (32) of the range, and the instruction will be executed.
4. Generally, the pulse instructions RCLP and DRCLP are used.

Example:

When X0.0 is switched from OFF to ON, the values of the bits in D10 are divided into groups (four bits as a group), and these groups are rotated to the left with the carry flag SM602. (The value of the bit marked ※ is transmitted to the carry flag SM602.)

Additional remark:

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0904		MBR	P	**S**、**D**、**n**	Rotating the matrix bits

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●							
D		●			●	●		●								
n	●	●			●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Matrix source

D : Operation result

n : Length of the array

Explanation:

1. The values of the **n** rows of bits in **S** are rotated to the right or to the left. When SM616 is OFF, the values of the bits are rotated to the left. When SM616 is ON, the values of the bits are rotated to the right. The vacancy resulting from the rotation is filled by the value of the bit rotated last, and the operation result is stored in **D**. The value of the bit rotated last not only fills the vacancy, but also is transmitted to the carry flag SM614.

2. The operand **n** used in the 16-bit instruction should be within the range between 1 and 16. The operand **n** used in the 32-bit instruction should be within the range between 1 and 32. When n is less than 0, the instruction will not be executed; when **n** exceeds the range, it will be seen as the maximum value (32) of the range, and the instruction will be executed.

3. Generally, the pulse instruction MBRP is used.

Example 1:

When X0.0 is ON, SM616 is OFF. The values of the bits in the 16-bit registers D0~D2 are rotated to the left, and the operation result is stored in the 16-bit registers D20~D22. The value of the bit marked ※ is transmitted to the carry flag SM614.

Example 2:

When X0.0 is ON, SM616 is ON. The values of the bits in the 16-bit registers D0~D2 are rotated to the right, and the operation result is stored in the 16-bit registers D20~D22. The value of the bit marked ※ is transmitted to the carry flag SM614.

Additional remark:

1. If **S**+**n**-1 or **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. The flags:

   SM614: It is the carry flag for the matrix rotation/shift/output.

   SM616: It is the direction flag for the matrix rotation/shift.

6.11 Timer and Counter Instructions

6.11.1 List of Timer and Counter Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1000	RST	DRST	–	Resetting the contact to OFF or clearing the value in the register.
1001	TMR	–	–	16-bit timer (Unit: 100ms)
1002	TMRH	–	–	16-bit timer (Unit: 1ms)
1003	CNT	–	–	16-bit counter
1004	–	DCNT	–	32-bit counter (Including the use of high-speed counters)
1005	–	DHSCS	–	Setting high-speed comparison
1006	–	DHSCR	–	Resetting high-speed comparison
1007	–	DHSZ	–	High-speed input zone comparison
1008	–	DSPD	–	Speed detection
1009	PWD	–	–	Pulse Width Detection
1010	–	DCAP	–	Capturing the high-speed count value in the external input interrupt
1011	TMRM	–	–	16-bit timer (Unit: 10ms)

6.11.2 Explanation of Timer and Counter Instructions

API	Instruction code			Operand	Function
1000	D	RST		**D**	Resetting the contact or clearing the register

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		○	○	○	○	○	○	○		○	○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●	●	●			●	●		●		●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol:

D : Device which is reset

Explanation:

1. DRST only supports the clearing of the values in the 32-bit HC device and two consecutive 16-bit D devices. For other devices, use RST instruction to clear their values.

2. When RST instruction is driven, the specified devices will act as follows.

Device	State
Bit	The coil and contact are set to OFF.
T · C	The current timer value and counter value are set to 0. And the coil and contact are set to OFF.
Word	The 16-bit content value will be cleared to 0.
DWord · HC · Real	The 32-bit content value including the floating point number will be cleared to 0.

3. If RST instruction is not executed, the state of the device specified by the instruction will keep unchanged.

4. The instruction supports the direct output.

Example:

When X0.0 is ON, Y0.5 is set to OFF.

The 32-bit D1 or D0 will be cleared to zero as X0.0 is ON.

API	Instruction code			Operand	Function
1001		TMR		S_1 , S_2	16-bit timer (100ms)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1					○											
S_2								○				○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1											●		
S_2		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Timer number

S_2 : Setting value of the timer

Explanation:

Refer to the explanation of API1002 TMRH instruction for details.

API	Instruction code			Operand	Function
1002		TMRH		S_1 · S_2	16-bit timer (1ms)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					○											
S_2								○				○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1											●		
S_2		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Timer number

S_2 : Setting value of the timer

Explanation:

1. The T timer specified by TMR instruction takes 100ms as the timing unit. And the T timer specified by TMRH instruction takes 1ms as the timing unit.

2. The range of the setting values of the timers in TMR and TMRH instructions is 0~32767.

3. If the same T timer is used repeatedly in the program, including being used in the different instructions TMR and TMRH, the setting value is the one that the value of the timer matches first.

4. The T timer will reset to zero automatically when the conditional contact changes from ON to OFF.

5. As long as users add the letter S in front of the device T, the timer used in the instruction TMR becomes an accumulative timer. When the conditional contact is OFF, the value of the accumulative timer is not cleared. When the conditional contact is ON, the timer counts from the current value. The ST accumulative timer should be used with the instruction RST together to clear the value of the timer.

6. If the same T timer is used repeatedly in the program, it is OFF when one of the conditional contacts is OFF.

7. If the same T timer is used repeatedly as T and ST in the program, T is OFF when one of the conditional contacts is OFF.

8. When the instruction TMR is executed, the specified timer coil is ON and the timer begins to count. As the value of the timer matches the setting value, the contact is ON.

9. The timers T0~T411 are defined as general timers. T412~T511 are subroutine timers by default. If the ranges of the two types of timers need be changed, use the hardware configuration software HWCONFIG for the modification.

10. The general timers will compare the timing values when TMR instructions are scanned. They are applied to the condition every time the TMR instruction status is scanned.

    For the subroutine timers, the system counts the time and compares the timing values after the END instruction is executed. And so the subroutine timers are applied to the situation in which TMR instruction is executed not in every scan but the long-lasting timing and comparing are needed.

Example 1:

When X0.0 is ON, the setting value 50 is loaded to the timer T0. When the value of T0 matches 50, the contact of T0 is ON.

Example 2:

When X0.0 is ON, the setting value 50 is loaded to the timer T0. When the value of T0 is 25 and X0.0 is switched from OFF to ON, T0 counts up from 25 to 50, and the contact of T0 is ON.

Example 3:

When X0.0 is ON, the setting value 1000 is loaded to the timer T5. When the value of T5 is 500 and X0.0 is switched from OFF to ON, T5 counts up from 500 to 1000, and the contact of T5 is ON.

Example 4:

When X0.0 is ON, the setting value 1000 is loaded to the timer T5. When the value of T5 is 500 and X0.0 is switched from OFF to ON, T5 counts up from 500 to 1000, and the contact of T5 is ON.

Additional remark:

When the operand S1 is declared via ISPSoft, the data type TIMER should be selected for the general T timer. If the accumulative ST timer is used, users need specify the ST device.

API	Instruction code			Operand	Function
1003		CNT		S_1 · S_2	16-bit counter

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1						○										
S_2								○				○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Counter number

S_2 : Setting value of the counter

Explanation:

1. When the instruction CNT is switched from OFF to ON, it indicates that the specified counter coil changes from OFF to ON and then the value of the counter is increased by 1. When the value of the counter matches the setting value, the contact of the counter will be ON.

2. When the value of the counter matches the setting value, the state of the contact and value of the counter both remain unchanged if any counting pulse is input. Use RST to reset the counter for one more counting.

Example:

When SM408 is ON for the first time, the setting value 10 is loaded to the counter C0 and the counter begins counting. After SM408 is swiched from OFF to ON ten times, the value of C0 is 10 and the contact of C0 is ON. After C0 is ON, SM408 continues to switch from OFF to ON. But the value of C0 will not be accumulated after it reaches the setting value of C0.

Additional remark:

When the operand S1 is declared via ISPSoft, the data type COUNTER should be selected.

API	Instruction code			Operand	Function
1004		DCNT		S_1 , S_2	32-bit counter

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1							○									
S_2								○					○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													
S_2			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
-	-	AS

Symbol:

S_1 : Counter value

S_2 : Setting value of the counter

Explanation:

6

1. DCNT is the instruction to enable the 32-bit counter within the range between HC0 and HC255.

2. When the operand S1 is declared via ISPSoft, the data type CNT can not be selected and users should specify an HC device number.

3. For the count-up/count-down counters HC0~HC63, when the conditional contact of the instruction DCNT is switched from OFF to ON, the counters will count up by increasing the values by 1 when SM621~SM684 are OFF or count down by decreasing the values by 1 when SM621~SM684 are ON.

4. For the count-up counters HC64~HC199, they will count up by increasing the values by 1 when the conditional contact of the instruction DCNT is switched from OFF to ON.

5. When the instruction DCNT is OFF, the counter will stop counting. But the original count value will not be cleared. RST can be used to clear the count value and reset the contact to OFF.

6. Refer to the following pages for details on the high-speed counter HC200~HC255.

Example:

NETWORK1→

When PLC runs, the value of the counter HC0 is cleared and the counter counts up as SM621 is OFF. At the moment, SM408 is ON for the first time. So the setting value 10 will be loaded to the counter HC0 and the counter will begin counting.

NETWORK2→

After SM408 is swiched from OFF to ON ten times, the value of the counter HC0 matches the setting value 10 and the contact of HC0 is ON. After HC0 is ON, the value of the counter will keep increasing because SM408 continues to change from OFF to ON though the value of HC0 has reached the setting value.

NETWORK3→

When HC0 continues to count up and the value reaches the setting value 20, the counter will count down as SM621 is ON in the program. After SM408 is switched from OFF to ON ten times and the value of HC0 decreases from 10 to 9, the contact of HC0 will be OFF.

After the contact of HC0 is OFF, the value of HC0 will still continue to decrease since SM408 continues to change from OFF to ON.

Additional remark:

1. For the setting mode of SM621~SM684, refer to the explanation of the 32-bit counter HC in Chapter 2.

Explanation of the high-speed counter:

AS300 high-speed counters can be classified into hardware counter (Up to 200KHz input at most and for differential input points, up to 4MHz input) and software counter (up to 10KHz at most).

Hardware counter:

Input / HC No.	X0.															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HC200	P#1												R#1			
HC201	P	D#1											R			
HC202	A#1	B#1											R			
HC203	--#2	--											--			
HC204			P											R		
HC205			P	D										R		
HC206			A	B										R		
HC207			--	--										--		
HC208					P										R	
HC209					P	D									R	
HC210					A	B									R	
HC211					--	--									--	
HC212							P									R
HC213							P	D								R
HC214							A	B								R
HC215							--	--								--
HC216									P							
HC217									P	D						
HC218									A	B						
HC219									--	--						
HC220											P					
HC221											P	D				
HC222											A	B				
HC223											--	--				

Note 1: P: single-phase pulse input, D: Direction signal input, A and B: two phase two input, R: Reset signal input.

Note 2: '--' means that the counting mode is reserved and unable to use now. The empty box indicates no function.

Note3: Refer to the SM/SR table for count up/down state selection and the selection of how many times frequency input.

Note 4: The function of R (input resetting) is disabled by default. Refer to SM/SR comparison table on how to use R.

Take HC200 for example. SM291 is switched to ON to start the R function and then the rising edge of X0.12 triggers the clearing of the value of HC200.

Software counter:

Input / HC No.	X0.															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HC232													P			
HC233													P	D		
HC234													A	B		
HC235													UP#5	DN#5		
HC236															P	
HC237															P	D
HC238															A	B
HC239															UP	DN
HC240		P														
HC241	UP	DN														
HC242				P												
HC243			UP	DN												
HC244						P										
HC245					UP	DN										
HC246								P								
HC247							UP	DN								
HC248										P						
HC249									UP	DN						
HC250												P				
HC251											UP	DN				
HC252														P		
HC253																P

Note 5: UP: Single phase count-up input (Same as CW), DN: Single phase count-down input (Same as CCW)

The high-speed counters within the range of HC200~HC255, which has not been listed in the table are the reserved devices inside PLC. They are not recommended to the program designer for use.

The high-speed counter, SM/SR and function are corresponded as shown in the following table.

	Count-up/count-down function			**Starting the Reset function**	**Reversing the direction**	**Counting mode**
HC No.	SM No.	Attribute	Explanation	SM No.	SM No.	SR No.
HC200	SM300	R/W	Show/ set	SM291	SM281 （Applicable for HC201）	SR190
HC201	SM301	R	Show			
HC202	SM302	R	Show			
HC204	SM304	R/W	Show/ set	SM292	SM282 （Applicable for HC205）	SR191
HC205	SM305	R	Show			
HC206	SM306	R	Show			
HC208	SM308	R/W	Show/ set	SM293	SM283 （Applicable for HC209）	SR192
HC209	SM309	R	Show			
HC210	SM310	R	Show			
HC212	SM312	R/W	Show/ set	SM294	SM284 （Applicable for HC213）	SR193
HC213	SM313	R	Show			
HC214	SM314	R	Show			
HC216	SM316	R/W	Show/ set	--	SM285 （Applicable for HC217）	SR194
HC217	SM317	R	Show			
HC218	SM318	R	Show			
HC220	SM320	R/W	Show/ set	--	SM286 （Applicable for HC221）	SR195
HC221	SM321	R	Show			
HC222	SM322	R	Show			
HC232	SM332	R/W	Show/ set	--	SM287 （Applicable for HC233）	SR196
HC233	SM333	R	Show			
HC234	SM334	R	Show			
HC235	SM335	R	Show			
HC236	SM336	R/W	Show/ set	--	SM288	SR197

	Count-up/count-down function			Starting the Reset function	Reversing the direction	Counting mode
HC No.	SM No.	Attribute	Explanation	SM No.	SM No.	SR No.
HC237	SM337	R	Show		（Applicable for HC237）	
HC238	SM338	R	Show			
HC239	SM339	R	Show			
HC240	SM340	R/W	Show/ set	--	--	Supports one time frequency and rising edge –triggered counting only
HC241	SM341	R	Show			
HC242	SM342	R/W	Show/ set			
HC243	SM343	R	Show			
HC244	SM344	R/W	Show/ set			
HC245	SM345	R	Show			
HC246	SM346	R/W	Show/ set	--	--	Supports one time frequency and rising edge –triggered counting only
HC247	SM347	R	Show	--	--	Supports one time frequency and rising edge –triggered counting only
HC248	SM348	R/W	Show/ set			
HC249	SM349	R	Show			
HC250	SM350	R/W	Show/ set			
HC251	SM351	R	Show			
HC252	SM352	R/W	Show/ set			
HC253	SM353	R	Show			

Note 1: All SM special flags in the above table are OFF by default.

Note 2: When SM under “Count-up/count-down function” is OFF, it indicates that the corresponding counter counts up or displays it is counting up. If SM is ON, it indicates that the corresponding counter counts down or displays it is counting down.

Note 3: The “R” under Attribute means “Read only” and “W” means “Set”.

Note 4: The SR special registers under “Counting mode” are of 1 time frequency input by default. The input value 2 indicates double frequency and 4 is the 4 times frequency. 4 times frequency is only applicable for the counter of A/B 2-phase input. If the value is not 1, 2 and 4 in SR, PLC will run with the 1 time frequency.

Note 5: All single-phase counters above counts by using 1 time frequency and the rising-edge counting mode changing the input point from OFF to ON.

Note 6: Reversing the direction is applicable for the counters of “P” (Pulse input) and “D” (Direction). When SM is ON, the counting direction (up/down) will be reversed. For example, when the preset direction input is OFF, the counter counts up. When SM is set to ON, the counter will change to count down.

API	Instruction code			Operand	Function
1005	D	HSCS		**S_1 · S_2 · D**	Setting high-speed comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								●					○	○		
D		○	○	○				○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2			●				●						
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	-	AS

Symbol:

S_1 : Counter number

S_2 : Comparative value

D : Comparison result

Explanation:

1. The instruction should be used with the high-speed counter of number HC200 and above together. If the high-speed counter specified by **S_1** changes by increasing or decreasing by 1 in the value, DHSCS instruction will make the comparison immediately. When the current value of the high-speed counter is equal to the comparative value specified by **S_2**, the device specified by **D** will change to ON. After that, the device specified by **D** remains ON even if the comparison result is that the current value and the comparative value become not equal..

2. If the device specified by **D** is Y0.0~Y0.15, the comparison result that the value of **S_2** is equal to the current value of the high-speed counter will be output to the output terminals Y0.0~Y0.15. And other Y devices will be affected by the scan cycle. But all devices are updated immediately and not affected by the scan cycle.

3. The **D** operand can also specify an interrupt device I within the range: I2□□, □□=00~67.

4. The high-speed counters can be classified into the software counter and hardware counter. The available high-speed comparators and interrupt device numbers are listed in the following table.

Type	Range of counter numbers	High-speed comparator number	High-speed interrupt device number
Hardware counter	HC200 ~ HC203	Comparator: HCC00~HCC03	I200~I203
	HC204 ~ HC207	Comparator: HCC04~HCC07	I210~I213
	HC208 ~ HC211	Comparator: HCC08~HCC11	I220~I223
	HC212 ~ HC215	Comparator: HCC12~HCC15	I230~I233
	HC216 ~ HC219	Comparator: HCC16~HCC19	I240~I243
	HC220 ~ HC223	Comparator: HCC20~HC223	I250~I253
Software counter	HC232 ~ HC253	-	I260~I267

5. Explanation of the hardware comparators for DHSCS, DHSCR and DHSZ instructions:

- Every one group of hardware counters share 4 high-speed comparators. One DHSCS or DHSCR instruction occupies 1 high-speed comparator. One DHSZ instruction uses 2 high-speed comparators.
- During the program editing, every group of hardware counters can use 4 high-speed comparators at most for DHSCS, DHSCR or DHSZ instructions. Otherwise, the syntax error will occur.

6. Explanation of the software comparators for DHSCS and DHSCR instructions:

- There are 8 software comparators for comparison of the Set or Reset function. Every one DHSCS or DHSCR instruction uses one high-speed comparator.
- The software comparators compare the interrupt by assigning a corresponding software comparator according to the interrupt numbers. It is noted that the same interrupt number can not be used repeatedly.
- For DHSCS or DHSCR instructions, the Set or Reset comparators for them can not exceed 8 pieces in the program. Otherwise, the syntax error will occur.

7. Explanation of the software comparators for DHSZ instruction:

- There are 8 software comparators for the zone comparison. One DHSZ instruction uses one comparator.
- DHSZ instruction can use maximum 8 software comparators. Otherwise, the syntax error will occur if the used software comparators are more than 8 pieces.

Example 1:

When M0 is ON, DHSCS instruction is executed.

When the current value of HC200 changes from 99 to 100 or from 101 to 100, Y0.10 is ON, which will output to the external output terminal Y0.10 in real time and it will remain ON.

Example 2:

The Y output of DHSCS instruction is different from the general Y output.

1. When M0 is ON, DHSCS instruction is executed. When the current value of HC200 changes from 99 to 100 or from 101 to 100, Y0.10 will output its state to the external output terminal immediately, which is irrelevant to the program scan time.

2. When the current value of HC200 changes from 99 to100, the contact of HC200 is ON immediately. But when the execution of SET Y0.11 is reached, Y0.11 will still be affected by the scan time and will output its state after END is passed .

Example 3:

The interrupt in hardware high-speed comparsion:

When the current value of HC200 changes from 99 to100 or 101 to100, the program jumps to the interrup pointer for execution of the interrupt program and Y0.10 is ON.

Main program:

I200 interrupt program:

API	Instruction code			Operand	Function
1006	D	HSCR		S_1 · S_2 · **D**	Resetting high-speed input comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								●					○	○		
D		○	○	○			○	○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2			●				●						
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	-	AS

Symbol:

S_1 : Counter number

S_2 : Comparative value

D : Comparison result

Explanation:

1. The instruction should be used with the high-speed counter of number HC200 and above together. If the high-speed counter specified by S_1 changes by increasing or decreasing the value by 1, DHSCR instruction will make the comparison immediately. When the current value of the high-speed counter is equal to the comparative value specified by S_2, the device specified by D will change to OFF. After that, the device specified by **D** remains OFF even if the comparison result is that the current value and the comparative value become not equal.

2. If the device specified by **D** is Y0.0~Y0.15, the comparison result that the comparative value of S_2 is equal to the current value of the counter will be output to the external output terminals Y0.0~Y0.15. And other Y devices will be affected by the scan cycle. But all devices are updated immediately and not affected by the scan cycle.

3. D operand can also specify the HC device to reset, which is only limited to the condition in which the high-speed counter number is the same as that of S_1.

4. Refer to DHSCS instruction for more information.

Example 1:

1. When M0 is ON and HC200 changes its current value from 99 to 100 or from 101 to 100, Y0.10 will be reset to OFF.

2. When HC200 changes its current value from 199 to 200, the contact of HC200 is ON and Y0.11 is ON. But the

program scan time output will be delayed.

Example 2:

For HC200 specified as the hardware high-speed counter of the same number, the contact of HC200 will be reset to OFF when HC200 changes its current value from 999 to 1000 or from 1001 to 1000.

API	Instruction code			Operand	Function
1007	D	HSZ		S_1 · S_2 · S_3 · D	High-speed input zone comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								●					○	○		
S_3								●					○	○		
D		○	○	○				○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2			●				●						
S_3			●				●						
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	-	AS

Symbol:

S_1 : Counter number

S_2 : Lower bound of the comparison zone

S_3 : Upper bound of the comparison zone

D : Comparison result (3 consecutive devices)

Explanation:

1. The instruction should be used with the high-speed counter of number HC200 and above together. The low bound of S_2 must be less than the upper bound of S_3. If the zone limit values are not set properly, PLC will make the adjustment of them automatically.

2. If S_1 specifies a software counter and the specified counter changes by increasing or decreasing by 1 in the value, DHSZ instruction will make the comparison right away. The comparison condition and output state are shown below.

Comparison condition	D+0 state	D+1 state	D+2 state
The count value of S_1 < the lower bound specified by S_2	ON	OFF	OFF
The lower bound specified by S_2 <= the count value of S_1 < the upper bound specified by S_3	OFF	ON	OFF
The count value of S_1 >= the upper bound specified by S_3	OFF	OFF	ON

Note: The lower bound specified by **S_2** must be less than the upper bound specified by **S_3**. If the zone bound is set incorrectly, PLC will make the adjustment automatically.

3. If **S_1** specifies a hardware counter and the value of the specified counter reaches the lower bound specified by **S_2** or the upper bound specified by **S_3**, DHSZ instruction will make the comparison immediately according to the count direction (up/down). And the comparison condition and output state are shown in the following table

Count direction	Comparison condition	**D**+0 state	**D**+1 state	**D**+2 state
Count up	The count value of **S_1** == the lower bound specified by **S_2**	OFF	ON	OFF
	The count value of **S_1** == the upper bound specified by **S_3**	OFF	OFF	ON
Count down	The count value of **S_1** == the lower bound specified by **S_2**	ON	OFF	OFF
	The count value of **S_1** == the upper bound specified by **S_3**	OFF	ON	OFF

4. If the device specified by **D** is Y0.0~Y0.15, the comparison result will be output to the external output terminals Y0.0~Y0.15. And other Y devices will be affected by the scan cycle. But all devices are updated immediately and not affected by the scan cycle.

5. Refer to DHSCS instruction for other relevant information on the high-speed zone comparison.

Example:

1. When D is specified as Y0.10, Y0.11~Y0.12 will be occupied automatically.

2. When DHSZ is executed, the instruction compares the current value in HC200 with the upper/lower bound (1500/2000) of the comparison zone, and one of Y0.10~Y0.12 will be ON according to the comparison result.

3. When the current value in HC200 <1500, Y0.10 is ON. When 1500<= the current value in HC200<2000, Y0.11 is ON.

   When the current value in HC200>=2000, Y0.12 is ON.

API	Instruction code			Operand	Function
1008	D	SPD		S_1、S_2、D	Speed detection

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								○	○				○	○		
D								○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2			●				●						
D			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
-	-	AS

Symbol:

S_1 : Counter value

S_2 : Setting value of the cycle time

D : Detected speed value

Explanation:

1. When executing this speed detection instruciton, S_1 has to be used with the instruction DCNT to enable the high speed counter with the counter number over HC200 (including HC200).

2. S_2 is the setting value of the cycle time with the unit of millisecond (ms). Setting range is between 10~1000. When the value is out of range, the system will execute it as the minimum value or the maximum value and the PLC will not send error messages.

3. When the set value in S_2 is reached, this instruction will store the number of pulses in the D apporinted device. That is why the PLC will not be affected by the PLC scanning.

4. This instruction has no limitation on the editing times, but it only allows 8 sets of speed detection instructions to run simultaneously. The 9th set of the speed detection instruction or later ones will be ignored and no error messages will be sent. When executing this instruction, the set parameters of the operand will be recorded. Thus during the execution of this instruciton, editing on the parameters is not allowed.

Example:

Where there is an input pulse signal in X0.0, DSPD instruction can be used for the speed detection. When M0 is ON, DSPD will have the number of pulses counted by HC200 shown in D0 every 500ms.

In the following example, the value in D0 is 7500 and the actual pulse input frequency of X0.0 is 15kHz (7500/500ms).

API	Instruction code			Operand	Function
1009		PWD		S_1 · S_2 · D	Pulse Width Detection

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2								○	○				○	○		
D_1								○								
D_2		○	○	○												

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2		●				●							
D_1			●				●						
D_2	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Number of the input point

S_2 : Unit of measurement

D_1 : Pulse width detection time (32-bit value)

D_2 : Updated flag

Explanation:

1. S_1 supports the following 8 inputs, X0.0/X0.1/X0.2/X0.3/X0.4/X0.6/X0.8/X0.10. But S_1 cannot share the same inputs with the high speed counter.

2. S_2 is the unit of measurement. The instruction will not be executed if the setting value of S_2 is not the one among the codes in the following table.

S_2 code	Unit of measurement	Range of detection	Applicable frequency range	Remark
0	1us		1Hz ~ 10kHz	
1	1ms		0.02Hz ~ 100Hz	
2	10ns		10Hz ~ 1MHz	Not supported by X0.1 and X0.3.
4	1us		1Hz ~ 10kHz	
5	1ms		0.02Hz ~ 100Hz	
Other values	PWD will not be executed.			

3. **D_1** is used for storing the pulse width detection time (32-bit value) and the detection range is 0~100,000,000. If the vaule is over the maximum value, it will be seen as the maximum value. If the vaule is 0, that means during the execution of this instruction, there is no input switched from ON to OFF.

4. **D_2** is the updated flag. Whenever the detection of the **S_1** input is completed and the instruciton is scanned, the updated flag will be switched to ON for one scan cycle time. Users can check if the detection value has been updated to the most recent value using the updated flag. When the instruciton is executed for the first time, the updated flag will be reset to OFF as well.

6

5. When the value in **S_2** is among 0, 1 and 2, refer to the timing diagram below for the procedures performed such as storing detection values and updating flags during the execution of the instruction. Time to start the timer is when the **S_1** input is switched from OFF to ON as it is shown in the position ① of the following diagram. Time to store the detection time is when the **S_1** input is switched from ON to OFF as shown in the position ② of the following diagram.

Figure 1 Detection mode when the value in **S_2** is 0, 1 or 2

6. When the value in **S_2** is 4, or 5, refer to the timing diagram below for the procedures performed such as storing detection values and updating flags during the execution of the instruction. Time to start the timer is when the **S_1** input

is switched from OFF to ON as it is shown in the position ① of the following diagram. Time to store the detection time is when the **S_1** input is switched from OFF to ON as shown in the position ② of the following diagram.

Figure 2 Detection mode when the value in **S_2** is 4 or 5

7. This instruction has no limitation on the editing times, but it only allows 8 sets of pulse width detection instructions to run simultaneously. The 9th set of the pulse width detection instruction or later ones will be invalid and no error messages will be sent. When executing this instruction, the set parameters of the operand will be recorded. Thus during the execution of this instruciton, editing on the parameters is not allowed.

8. Before executing this instruction, please check the input hardware response time and the pulse time set in HWCONFIG. For example, when the value in the **S_2** is set to 0 or 2, that means the unit of measurement is microsecond (μs). And the **S_1** input value should be set to 0 to disable the Input Point Filter Time in HWCONFIG.

Example:

There is a pulse signal of 10kHz in the input X0.0. When M0 is ON, the instruction PWD can detect the input signal of the X0.0 and have the pulse width shown in D10/D11 (32-bit data), the time unit is set to 0 and the width detected from D10 is 50μs.

API	Instruction code			Operand	Function
1010	D	CAP		S_1 , S_2 , D	Capturing the high-speed count value in the external input interrupt

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2							○									
D								○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2												●	
D			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
-	-	AS

Symbol:

S_1 : Number of an external interrupt input point

S_2 : High-speed counter number

D : Register for storing the captured value

Explanation:

6

1. Only the 16 input points X0.0~X0.15 of PLC can be used in S_1. They are used with the external interrupt service program to start the function. It is to be noted that S_1 should not share the same input point with the high-speed counter.

2. The high-speed counter HC device is selected in S_2. The HC device has to be used with the DCNT instruction to start the counting function.

3. The captured value of the high-speed counter is stored in (32-bit) **D** when the interrupt takes place. Time to store data is when the interrupt occurs and not affected by the PLC program scan.

4. The procedure of the instruction operation is shown below: (Suppose that the input interrupt is triggered by the falling edge.)

   ① → When the execution of the instruction starts, the value in **D** will not change and users can enter the default value.

   ② → When the interrupt in S_1 occurs, the value of the counter specified by S_2 is captured immediately and stored in D.

5. The instruction can start DCAP instructions for four different input points at most. If one input point is set as the external interrupt triggered by the rising edge and falling edge, the time to capture the value is when the input is triggered by the rising edge and by falling edge respectively and the count value is stored in the device specified by **D**. When two instructions specify the same interrupt input point, the one which is started first will use the interrupt input point first.

6. The HC device number is set in **S_2**. The high-speed counters within the range of HC200~HC255 are recommended. For details on the counters, refer to the explanation of API1004 DCNT instruction.

Example:

External interrupt is triggered by the rising edge in X0.7.

External interrupt is triggered by the falling edge in X0.7.

Note:

1. When M0 is ON, the DCAP instruction is enabled. When an external interrupt occurs in X0.7, the value in HC200 is captured and stored to (32-bit) D0.

2. When the external input interrupt is triggered by the rising edge once, E0 is modified to 0 through setting D100, the count value in D0 is stored to D10 through modifying E0 and the value in D100 is 0+2.

3. When the external input interrupt is triggered by the falling edge once, E0 is modified to 2 through setting D100, the count value (10+E0=12) in D0 is stored to D12 through modifying E0 and the value in D100 is 0+2. When the value in D100 is 20, D100 is cleared to 0.

4. If the external interrupt is triggered by the rising edge and falling edge five times respectively, the value capturing will be carried out 10 times and the captured values will be stored in D10, D12…D28.

   The 1st captured value= D10

   The 2nd captured value= D12

   .

   .

   The 10th captured value=D28

   The 11th captured value=D10

API	Instruction code			Operand	Function
1011		TMRM		S_1 · S_2	16-bit timer (10ms)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					○											
S_2								○				○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1											●		
S_2		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Timer number

S_2 : Setting value of the timer

Explanation:

The instruction takes 10ms as the unit.

Refer to the explanation of API1002 TMRH instruction for details.

6.12 Shift Instructions

6.12.1 The List of Shift Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1100	SFTR	–	✓	Shifting the states of the devices to the right
1101	SFTL	–	✓	Shifting the states of the devices to the left
1102	WSFR	–	✓	Shifting the data in the word devices to the right
1103	WSFL	–	✓	Shifting the data in the word devices to the left
1104	SFWR	–	✓	Shifting the data and writing it into the word device
1105	SFRD	–	✓	Shifting the data and reading it from the word device
1106	SFPO	–	✓	Reading the latest data from the data list
1107	SFDEL	–	✓	Deleting the data from the data list
1108	SFINS	–	✓	Inserting the data into the data list
1109	MBS	–	✓	Shifting the matrix bits
1110	SFR	–	✓	Shifting the values of the bits in the 16-bit registers by **n** bits to the right
1111	SFL	–	✓	Shifting the values of the bits in the 16-bit registers by **n** bits to the left
1112	BSFR	–	✓	Shifting the states of the **n** bit devices by one bit to the right
1113	BSFL	–	✓	Shifting the states of the **n** bit devices by one bit to the left
1114	NSFR	–	✓	Shifting **n** registers to the right
1115	NSFL	–	✓	Shifting **n** registers to the left

6.12.2 Explanation of Shift Instructions

API	Instruction code			Operand	Function
1100		SFTR	P	**S** · **D** · **n_1** · **n_2**	Shifting the states of the devices to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●	●	●				●								
D		●	●	●				●								
n_1								●	●		○	○	○	○		
n_2								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												
D	●												
n_1		●			●	●							
n_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Initial device in which the value is shifted

D : Initial device in which the value is shifted

n_1 : Length of the data which is shifted

n_2 : Number of bits forming a group

Explanation:

1. The states of the **n_1** bit devices starting from **D** are divided into groups (**n_2** bits as a group), and these groups are shifted to the right. The states of the **n_2** bit devices starting from **S** are shifted to the devices starting from **D** to fill the vacancy.

2. Generally, the pulse instruction SFTRP is used.

3. The operand **n_1** should be within the range between 1 and 1024. The operand **n_2** should be within the range between 1 and **n_1**.

Example 1:

1. When X0.0 is switched from OFF to ON, the states of the sixteen bit devices starting from M0 to M15 are divided into groups (four bits as a group), and these groups are shifted to the right.

2. The shift of the states of the bit devices to the right during a scan is illustrated as follows.

❶ M3~M0 → Being carried

❷ M7~M4 → M3~M0

❸ M11~M8 → M7~M4

❹ M15~M12 → M11~M8

❺ X0.3~X0.0 → M15~M12

Example 2:

1. When X0.0 is switched from OFF to ON, the states of the sixteen bit devices starting from M0 to M15 are divided into groups (five bits as a group), and these groups are shifted to the right.

2. The shift of the states of the bit devices to the right during a scan is illustrated as follows.

❶ M0 → Being carried

❷ M5 → M0

❸ M10~M6 → M5~M1

❹ M15~M11 → M10~M6

❺ X0.4~X0.0 → M15~M11

Additional remark:

1. If **S**+**n**$_2$-1 or **D**+**n**$_1$-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n**$_1$ is less than 1, or if **n**$_1$ is larger than 1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If **n**$_2$ is less than 1, or if **n**$_2$ is larger than **n**$_1$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1101		SFTL	P	**S**、**D**、**n_1**、**n_2**	Shifting the states of the devices to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●	●	●				●								
D		●	●	●				●								
n_1								●	●		○	○	○	○		
n_2								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												
D	●												
n_1		●			●	●							
n_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Initial device in which the value is shifted

D : Initial device in which the value is shifted

n_1 : Length of the data which is shifted

n_2 : Number of bits forming a group

Explanation:

1. The states of the **n_1** bit devices starting from **D** are divided into groups (**n_2** bits as a group), and these groups are shifted to the left. The states of the **n_2** bit devices starting from **S** are shifted to the devices starting from **D** to fill the vacancy.

2. Generally, the pulse instruction SFTLP is used.

3. The operand **n_1** should be within the range between 1 and 1024. The operand **n_2** should be within the range between 1 and **n_1**.

Example 1:

1. When X0.0 is switched from OFF to ON, the states of the sixteen bit devices starting from M0 to M15 are divided into groups (four bits as a group), and these groups are shifted to the left.

2. The shift of the states of the bit devices to the left during a scan is illustrated as follows.

❶ M15~M12 → Being carried

❷ M11~M8 → M15~M12

❸ M7~M4 → M11~M8

❹ M3~M0 → M7~M4

❺ X0.3~X0.0 → M3~M0

6

Example 2:

1. When X0.0 is switched from OFF to ON, the states of the sixteen bit devices starting from M0 to M15 are divided into groups (five bits as a group), and these groups are shifted to the left.

2. The shift of the states of the bit devices to the left during a scan is illustrated as follows.

❶ M15 → Being carried

❷ M10 → M15

❸ M9~M5 → M14~M10

❹ M4~M0 → M9~M5

❺ X0.4~X0.0 → M4~M0

Additional remark:

1. If **S**+$\mathbf{n_2}$-1 or **D**+$\mathbf{n_1}$-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If $\mathbf{n_1}$ is less than 1, or if $\mathbf{n_1}$ is larger than 1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If $\mathbf{n_2}$ is less than 1, or if $\mathbf{n_2}$ is larger than $\mathbf{n_1}$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1102		WSFR	P	**S** · **D** · **n_1** · **n_2**	Shifting the data in the word devices to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●							
D		●			●	●		●								
n_1								●	●		○	○	○	○		
n_2								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n_1		●			●	●							
n_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Initial device in which the value is shifted

D : Initial device in which the value is shifted

n_1 : Length of the data which is shifted

n_2 : Number of bits forming a group

Explanation:

1. The data in the **n_1** word devices starting from **D** is divided into groups (**n_2** words as a group), and these groups are shifted to the right. The data in the **n_2** word devices starting from **S** are shifted to the devices starting from **D** to fill the vacancy.

2. Generally, the pulse instruction WSFRP is used.

3. The operand **n_1** should be within the range between 1 and 512. The operand **n_2** should be within the range between 1 and **n_1**.

Example 1:

1. When X0.0 is switched from OFF to ON, the data in the sixteen word devices starting from D20 to D35 is divided into groups (four words as a group), and these groups are shifted to the right.

2. The shift of the data in the word devices to the right during a scan is illustrated as follows.

❶ D23~D20 → Being carried

❷ D27~D24 → D23~D20

❸ D31~D28 → D27~D24

❹ D35~D32 → D31~D28

❺ D13~D10 → D35~D32

Example 2:

1. When X0.0 is switched from OFF to ON, the data in the sixteen word devices starting from D20 to D35 is divided into groups (five words as a group), and these groups are shifted to the right.

2. The shift of the data in the word devices to the right during a scan is illustrated as follows.

❶ D20 → Being carried

❷ D25 → D20

❸ D30~D26 → D25~D21

❹ D35~D31 → D30~D26

❺ D14~D10 → D35~D31

Additional remark:

1. If **S**+**n**$_2$-1 or **D**+**n**$_1$-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n**$_1$ is less than 1, or if **n**$_1$ is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If **n**$_2$ is less than 1, or if **n**$_2$ is larger than **n**$_1$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1103		WSFL	P	**S** · **D** · **n_1** · **n_2**	Shifting the data in the word devices to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●							
D		●			●	●		●								
n_1								●	●		○	○	○	○		
n_2								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n_1		●			●	●							
n_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Initial device in which the value is shifted

D : Initial device in which the value is shifted

n_1 : Length of the data which is shifted

n_2 : Number of bits forming a group

Explanation:

1. The data in the **n_1** word devices starting from **D** is divided into groups (**n_2** words as a group), and these groups are shifted to the left. The data in the **n_2** word devices starting from **S** are shifted to the devices starting from **D** to fill the vacancy.

2. Generally, the pulse instruction WSFLP is used.

3. The operand **n_1** should be within the range between 1 and 512. The operand **n_2** should be within the range between 1 and **n_1**.

Example 1:

1. When X0.0 is switched from OFF to ON, the data in the sixteen word devices starting from D20 to D35 is divided into groups (four words as a group), and these groups are shifted to the left.

2. The shift of the data in the word devices to the left during a scan is illustrated as follows.

❶ D35~D32 → Being carried

❷ D31~D28 → D35~D32

❸ D27~D24 → D31~D28

❹ D23~D20 → D27~D24

❺ D13~D10 → D23~D20

6

Example 2:

1. When X0.0 is switched from OFF to ON, the data in the sixteen word devices starting from D20 to D35 is divided into groups (five words as a group), and these groups are shifted to the left.

2. The shift of the data in the word devices to the left during a scan is illustrated as follows.

❶ D35 → Being carried

❷ D30 → D35

❸ D29~D25 → D34~D30

❹ D24~D20 → D29~D25

❺ D14~D10 → D24~D20

Additional remark:

1. If **S**+$\mathbf{n_2}$-1 or **D**+$\mathbf{n_1}$-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If $\mathbf{n_1}$ is less than 1, or if $\mathbf{n_1}$ is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If $\mathbf{n_2}$ is less than 1, or if $\mathbf{n_2}$ is larger than $\mathbf{n_1}$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1104		SFWR	P	**S · D · n**	Shifting the data and writing it into the word device

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●								
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Device in which the data is shifted

D : Initial device

n : Data length

Explanation:

6

1. The data in the **n** word devices starting from the device specified by **D** is defined as a first in, first out data type, and the device specified by **D** is taken as a pointer. When the instruction is executed, the value of the pointer increases by one, and the data in the device specified by **S** is written into the device specified by the pointer. When the value of the pointer is larger than or equal to **n**-1, the instruction does not process the writing of the data, and the carry flag SM602 is ON.

2. Generally, the pulse instruction SFWRP is used.

3. The operand **n** should be within the range between 2 and 512.

Example:

1. The value of the pointer D0 is cleared to 0 first. When X0.0 is switched from OFF to ON, the data in D20 is written into D1, and the value in D0 becomes 1. When X0.0 is switched from OFF to ON again, the data in D20 is written to D2, and the value in D0 becomes 2.

2. The data in the word device is shifted and written in the following way.

   - The data in D20 is written into D1.

   - The value in D0 becomes 1.

Additional remark:

1. If the value in **D** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **D**+**n**-1 exceeds the device range, the instruction is not executed. SM0 is ON, and the error code in SR0 is 16#2003.

3. If **n** is less than 2, or if **n** is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. The instruction SFWR can be used with the instruction SFRD to write and read the data.

API	Instruction code			Operand	Function
1105		SFRD	P	**S · D · n**	Shifting the data and reading it from the word device

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S		●			●	●		●			○					
D		●			●	●		●								
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Initial device

D : Device in which the data is shifted

n : Data length

Explanation:

6

1. The data in the **n** word devices starting from the device specified by **S** is defined as a first in, first out data type, and the device specified by **S** is taken as a pointer. When the instruction is executed, the value in the device specified by **S** decreases by one, the data in the device specified by **S**+**1** is written into the device specified by **D**, the data in the devices specified by **S**+**n**-1~**S**+2 is shifted to the right, and the data in the device specified by **S**+**n**-1 is unchanged. When the value in the device specified by **S** is equal to 0, the instruction does not process the reading of the data, and the zero flag SM600 is ON.

2. Generally, the pulse instruction SFRDP is used.

3. The operand **n** should be within the range between 2 and 512.

Example:

1. When X0.0 is switched from OFF to ON, the data in D21 is written into D0, the data in D29~D22 is shifted to the right, the data in D29 is unchanged, and the value in D20 decreases by one.

2. The data in the word device is shifted and read in the following way.

   - The data in D21 is read and shifted to D0.

   - The data in D29~D22 is shifted to the right.

- The value in D20 decreases by one.

Additional remark:

1. If the value in **S** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If **n** is less than 2, or if **n** is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. The instruction SFWR can be used with the instruction SFRD to write and read the data.

API	Instruction code			Operand	Function
1106		SFPO	P	**S · D**	Reading the latest data from the data list

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S		●			●	●		●								
D		●			●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Initial device

D : Device in which the data is stored

Explanation:

1. The device specified by **S** is taken as a pointer. When the instruction is executed, the data in the device specified by the value of the pointer is written into the device specified by **D** and cleared to 0, and the value in the device specified by **S** decreases by one. When the value in the device specified by **S** is equal to 0, the instruction does not process the reading of the data, and the zero flag SM600 is ON.

2. Generally, the pulse instruction SFPOP is used.

Example:

When X0.0 is ON, the data in the device specified by the value in D0 is written into D10. After the data is shifted, the data in the device specified by the value in D0 is cleared to 0, and the value in D0 increases by one.

Additional remark:

1. If the value in **S** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S**+(The value in **S**) exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1107		SFDEL	P	**S**、**D**、**n**	Deleting the data from the data list

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S		●			●	●		●								
D		●			●	●		●			○	○				
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Initial device

D : Device in which the data is stored

n : Device in which the data is deleted

6

Explanation:

1. The value in the device specified by **S** indicates the length of the data, and the data is in the devices specified by **S**+1~**S**+(The value in **S**). When the instruction is executed, the data in the device specified by **S**+**n** is stored in **D** and deleted, the data in the devices specified by **S**+**n**+**1**~**S**+(The value in **S**) is shifted to the right, the data in the device specified by **S**+(The value in **S**) is cleared to 0, and the value in the device specified by **S** decreases by one. When the value in the device specified by **S** is equal to 0, the instruction does not process the deleting of the data, and the zero flag SM600 is ON.

2. Generally, the pulse instruction SFDELP is used.

3. The operand **n** should be within the range between 1 and 32767.

Example:

Suppose the value in D0 is 9, and **n** is 4. When X0.0 is ON, the data in D4 is stored in D20. After the data in D4 is deleted, the data in D5~D9 is shifted to the right, and the value in D0 decreases by one.

Additional remark:

1. If the value in **S** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S**+**n** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If **S**+(The value in **S**) exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If **n** is larger than the value in **S**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

5. If **n** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1108		SFINS	P	**S**・**D**・**n**	Inserting the data into the data list

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S		●			●	●		●								
D	●	●			●	●		●	●		○	○	○	○		
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Initial device

D : Data which is inserted

n : Device into which the data is inserted

Explanation:

6

1. The value in the device specified by **S** indicates the length of the data, and the data is in the devices specified by **S**+1~**S**+(The value in **S**). When the instruction is executed, the data in **D** is inserted into **S**+**n**, the original data in the devices specified by **S**+**n**~**S**+(The value in **S**) is shifted to the left, and the value in the device specified by **S** increases by one. When the value in the device specified by **S** is equal to 32767, the instruction does not process the writing of the data, the value in the device specified by **S** does not increase, and the carry flag SM602 is ON.

2. Generally, the pulse instruction SFINSP is used.

3. The operand **n** should be within the range between 1 and 32767.

Example:

Suppose the value in D0 is 8, and **n** is 4. When X0.0 is ON, the data in D200 is inserted into D4, the original data in D4~D8 is shifted to D5~D9, and the value in D0 increases by one.

Additional remark:

1. If the value in **S** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S**+**n** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003

3. If **S**+(The value in **S**)+1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If **n** is larger than the value in **S**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

5. If **n** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1109		MBS	P	**S**、**D**、**n**	Shifting the matrix bits

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●							
D		●			●	●		●								
n	●	●			●	●		●	●		○		○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Matrix source

D : Operation result

n : Length of the array

Explanation:

6

1. The values of the **n** rows of bits in **S** are shifted to the right or to the left. When SM616 is OFF, the values of the bits are shifted to the left. When SM616 is ON, the values of the bits are shifted to the right. The vacancy (from shifted to the left is b0 and from shift to the right is b16n-1) resulting from the shift is filled by the state of the borrow flag SM615, the value of the bit shifted last (from shifted to the left is b16n-1 and from the shift to the right is b0) is transmitted to the carry flag SM614, and the operation result is stored in **D**.

2. The operand **n** should be within the range between 1 and 256.

3. Generally, the pulse instruction MBSP is used.

Example 1:

When X0.0 is ON, SM616 is OFF. The values of the bits are shifted to the left. Suppose SM615 is OFF. After the values of the bits in the 16-bit registers D0~D2 are shifted to the left, the operation result is stored in the 16-bit registers D20~D22, and SM614 is ON.

Example 2:

When X0.0 is ON, SM616 is ON. The values of the bits are shifted to the right. Suppose SM615 is ON. After the values of the bits in the 16-bit registers D0~D2 are rotated to the right, the operation result is stored in the 16-bit registers D20~D22, and SM614 is OFF.

Additional remark:

1. If **S**+**n**-1 or **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. The flags:

SM614: It is the carry flag for the matrix rotation/shift/output.

SM615: It is the borrow flag for the matrix shift/output.

SM616: It is the direction flag for the matrix rotation/shift.

API	Instruction code			Operand	Function
1110		SFR	P	**D · n**	Shifting the values of the bits in the 16-bit registers by **n** bits to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●		●			○	○				
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

D : Device involved in the shift

n : Number of bits

Explanation:

1. The values of the bits in **D** are shifted by **n** bits to the right. The vacancies (b15~b15-**n**+1) resulting from the shift is filled by 0, and the value of b**n**-1 is transmitted to SM602.

2. The operand **n** should be within the range between 1 and 16.

3. Generally, the pulse instruction SFRP is used.

Example:

When X0.0 is ON, the values of b0~b15 in D0 are shifted by 6 bits to the right, and the value of b5 is transmitted to SM602. The values of b10~b15 are cleared to 0 after the shift.

The shift of the values of the bits to the right during a scan is illustrated as follows.

❶ b5~b0 → Being carried (The value of b5 is transmitted to SM602.)

❷ b15~b6 → b9~b0

❸ 0 → b15~b10

Additional remark:

If **n** is less than 0, or if **n** is larger than 16, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1111		SFL	P	**D**、**n**	Shifting the values of the bits in the 16-bit registers by **n** bits to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●		●			○	○				
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

D : Device involved in the shift

n : Number of bits

Explanation:

1. The values of the bits in **D** are shifted by **n** bits to the left. The vacancies (b0~b**n**-1) resulting from the shift is filled by 0, and the value of b16-**n** is transmitted to SM602.

2. The operand **n** should be within the range between 1 and 16.

3. Generally, the pulse instruction SFLP is used.

Example:

When X0.0 is ON, the values of b0~b15 in D0 are shifted by 6 bits to the right, and the value of b10 is transmitted to SM602. The values of b0~b5 are cleared to 0 after the shift.

The shift of the values of the bits to the left during a scan is illustrated as follows.

❶ b15~b10 → Being carried (The value of b10 is transmitted to SM602.

❷ b9~b0 → b15~b6

❸ 0 → b5~b0

Additional remark:

If **n** is less than 0, or if **n** is larger than 16, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1112		BSFR	P	**D**、**n**	Shifting the states of the **n** bit devices by one bit to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●	●	●				●								
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

D : Initial device involve in the shift

n : Data length

Explanation:

1. The states of the **n** bit devices starting from **D** are shifted by one bit to the right. The state of **D**+**n**-1 is cleared to 0, and the state of **D** is transmitted to the carry flag SM602.

2. Generally, the pulse instruction BSFRP is used.

3. The operand **n** should be within the range between 1 and 1024.

Example:

When X0.0 is ON, the states of M0~M5 are shifted by one bit to the right, the state of M5 is cleared to 0, and the state of M0 is transmitted to the carry flag SM602.

Additional remark:

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1113		BSFL	P	**D · n**	Shifting the states of the **n** bit devices by one bit to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●	●	●				●								
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

D : Initial device involve in the shift

n : Data length

Explanation:

1. The states of the **n** bit devices starting from **D** are shifted by one bit to the left. The state of **D** is cleared to 0, and the state of **D**+**n**-1 is transmitted to the carry flag SM602.

2. Generally, the pulse instruction BSFLP is used.

3. The operand **n** should be within the range between 1 and 1024.

Example:

When X0.0 is ON, the states of M0~M5 are shifted by one bit to the left, the state of M0 is cleared to 0, and the state of M5 is transmitted to the carry flag SM602.

Additional remark:

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1114		NSFR	P	**D · n**	Shifting **n** registers to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●		●								
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

D : Initial device involve in the shift

n : Data length

Explanation:

1. The data in the **n** registers starting from **D** is shifted to the right, and the data in **D**+**n**-1 is cleared to 0.

2. Generally, the pulse instruction NSFRP is used.

3. The operand **n** should be within the range between 1 and 512.

Example:

When X0.0 is ON, the data in D1~D6 is shifted to the right, and the data in D6 is cleared to 0.

D6	D5	D4	D3	D2	D1	D0
30	2235	9578	754	28	423	11

After the shift

D6	D5	D4	D3	D2	D1	D0
0	30	2235	9578	754	28	423

Being cleared to 0

Additional remark:

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1115		NSFL	P	**D · n**	Shifting **n** registers to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●		●								
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

D : Initial device involve in the shift

n : Data length

Explanation:

1. The data in the **n** registers starting from **D** is shifted to the left, and the data in **D** is cleared to 0.

2. Generally, the pulse instruction NSFLP is used.

3. The operand **n** should be within the range between 1 and 512.

Example:

When X0.0 is ON, the data in D0~D5 is shifted to the left, and the data in D0 is cleared to 0.

D6	D5	D4	D3	D2	D1	D0
30	2235	9578	754	28	423	11

After the shift

D6	D5	D4	D3	D2	D1	D0
2235	9578	754	28	423	11	0

Being cleared to 0

Additional remark:

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

6.13 Data Processing Instructions

6.13.1 List of Data Processing Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1200	SER	DSER	✓	Searching the data
1201	SUM	DSUM	✓	Number of bits whose states are ON
1202	DECO	–	✓	Decoder
1203	ENCO	–	✓	Encoder
1204	SEGD	–	✓	Seven-segment decoding
1205	SORT	DSORT	✓	Sorting the data
1206	ZRST	–	✓	Resetting the zone
1207	BON	DBON	✓	Checking the state of the bit
1208	MEAN	DMEAN	✓	Mean
1209	CCD	–	✓	Sum check
1210	ABS	DABS	✓	Absolute value
1211	MINV	–	✓	Inverting the matrix bits
1212	MBRD	–	✓	Reading the matrix bit
1213	MBWR	–	✓	Writing the matrix bit
1214	MBC	–	✓	Counting the bits with the value 0 or 1
1215	DIS	–	✓	Disuniting the 16-bit data
1216	UNI	–	✓	Uniting the 16-bit data
1217	WSUM	DWSUM	✓	Getting the sum
1221	LIMIT	DLIMIT	✓	Confining the value within the bounds
1222	BAND	DBAND	✓	Deadband control
1223	ZONE	DZONE	✓	Controlling the zone
1224	–	FMEAN	✓	The mean of the floating points
1225	–	FSUM	✓	The sum of the floating points

6.13.2 Explanation of Data Processing Instructions

API	Instruction code			Operand	Function
1200	D	SER	P	**S_1 · S_2 · D · n**	Searching the data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●							
S_2	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●								
n	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Initial device involved in the comparison

S_2 : Compared data

D : Initial device in which the comparison result is stored

n : Data length

Explanation:

1. **n** singed decimal values in the registers starting from the register specified by **S_1** are compared with the singed decimal value in the register specified by **S_2**, and the comparison results are stored in the registers **D**~**D**+4.

Device	Description
D	Number of equal values
D+1	Data number of the first equal value
D+2	Data number of the last equal value
D+3	Data number of the minimum value
D+4	Data number of the maximum value

2. The operand **n** should be within the range between 1 and 256.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example:

1. When X0.0 is ON, the values in D10~D19 are compared with the value in D0, and the comparison results are stored in D50~D54. When the equal value does not exist, the values in D50~D52 are 0.

2. The data number of the minimum value is stored in D53, and the data number of the maximum value is stored in D54. If there is more than one minimum value or maximum value, the data number which is bigger is stored.

n

S_1	Value	Compared data	Data number	Result
D10	88	S_2 D0=100	0	
D11	100		1	Equal
D12	110		2	
D13	150		3	
D14	100		4	Equal
D15	300		5	
D16	100		6	Equal
D17	5		7	Minimum
D18	100		8	Equal
D19	500		9	Maximum

D	Value	Description
D50	4	Number of equal values
D51	1	Data number of the first equal value
D52	8	Data number of the last equal value
D53	7	Data number of the minimum value
D54	9	Data number of the maximum value

Additional remark:

1. If **S_1+n**-1 or **D**+4 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the operand **n** used in the 16-bit instruction is less than 1 or larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If the operand **n** used in the 32-bit instruction is less than 1 or larger than 128, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. If the operand **D** used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [5] of WORD/INT.

5. If the operand **D** used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [5] of DWORD/DINT.

API	Instruction code			Operand	Function
1201	D	SUM	P	**S · D**	Number of bits whose states are ON

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source device

D : Destination device

Explanation:

1. The number of bits whose values are 1 in **S** is stored in **D**.

2. When the values of the bits in the source device specified by S are 0, the zero flag SM600 is ON.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example:

When X0.0 is ON, the number of bits whose values are 1 in D0 is stored in D2.

Additional remark:

If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1202		DECO	P	**S · D · n**	Decoder

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●	●	●	●	●		●	●	○	○	○				
D		●	●	●	●	●		●				○				
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●	●			●	●							
D	●	●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Source device

D : Device in which the decoded values are stored

n : Number of bits whose values are decoded

Explanation:

1. The values of the lower **n** bits in the source device specified by **S** are decoded as the values of the lower 2^n bits in **D**.

2. The values of the consecutive **n** bits in the source device specified by **S** are decoded as the values of the lower 2^n bits in **D**.

3. When the source device specified by **S** is a timer or counter, the device will be seen as a word device.

4. When **D** is a bit device, **n** is within the range between 1 and 8. When **n** is 8, the values of the 8 bits is decoded as the values of the 256 bits. (Please note that the devices in which the decoded values are stored can not be used repeatedly.)

5. When **D** is a word device, **n** is within the range between 1 and 4. When **n** is 4, the values of the 4 bits is decoded as the values of the 16 bits.

6. Generally, the pulse instruction DECOP is used.

Example 1:

1. When Y0.0 is switched from OFF to ON, the instruction DECO decodes the values of the 3 bits in X0.0~X0.2 as the values of the 8 bits in M100~M107.

2. After the values of the 3 bits in X0.0~X0.2 are added up, the value 3 is gotten. The third bit in M10~M1007, that is, the bit in M103, is set to 1.

3. After the instruction DECO is executed and Y0.0 is switched OFF, the values of the 8 bits in M100~M107 are unchanged.

Example 2:

1. When X0.0 is switched from OFF to ON, the instruction DECO decodes the values of b2~b0 in D10 as the values of b7~b0 in D20, and the values of b15~b8 in D10 become 0.

2. The values of the lower 3 bits in D10 is decoded as the values of the lower 8 bits in D20. The values of the higher 8 bits are 0.

3. After the instruction DECO is executed and X0.0 is switched OFF, the data in D20 is unchanged.

Additional remark:

1. Suppose **D** is a bit device. If **n** is less than 1, or if **n** is larger than 8, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. Suppose **D** is a word device. If **n** is less than 1, or if **n** is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. Suppose **S** is a bit device. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. Suppose **D** is a bit device. If **D**+(2^**n**)-**1** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1203		ENCO	P	**S · D · n**	Encoder

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●	●	●	●	●		●	●	○	○	○				
D		●			●	●		●			○	○				
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●	●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Source device

D : Device in which the encoded values are stored

n : Number of bits whose values are encoded

Explanation:

1. When **S** is a word device, the values of the lower 2^n bits in the source device specified by **S** are encoded as the values of the lower **n** bits in **D**.

2. When **S** is a bit device, the higher bit with the value **S**+(n-1) from the lower 2^n bits is processed and the result is stored in **D**.

3. When the source device specified by **S** is a timer or counter, the device will be seen as a word device.

4. When **S** is a bit device, **n** is within the range between 1 and 8. When **n** is 8, the values of the 256 bits is encoded as the values of the 8 bits.

5. When **S** is a word device, **n** is within the range between 1 and 4. When **n** is 4, the values of the 16 bits is encoded as the values of the 4 bits.

6. Generally, the pulse instruction ENCOP is used.

Example 1:

1. When X0.0 is switched from OFF to ON, the instruction ENCO encodes the values of the 8 bits in M0~M7 as the values of the lower 3 bits in D0, and the values of b15~b3 in D0 become 0.

2. After the instruction ENCO is executed and X0.0 is switched OFF, the data in **D** is unchanged.

Example 2:

1. When X0.0 is switched from OFF to ON, the instruction ENCO encodes the values of b0~b7 in D10 as the values of b2~b0 in D20, and the values of b15~b3 in D20 become 0. (The values of b8~b18 in D10 are invalid data.)

2. After the instruction ENCO is executed and X0.0 is switched OFF, the data in **D** is unchanged.

Additional remark:

1. If there is no bit whose value is 1 in the source device specified by S, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. Suppose **S** is a bit device. If **n** is less than 1, or if **n** is larger than 8, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. Suppose **S** is a word device. If **n** is less than 1, or if **n** is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. Suppose **S** is a bit device. If **S**+(2^**n**)-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

5. Suppose **D** is a bit device. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1204		SEGD	P	**S · D**	Seven-segment decoding

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Source device

D : Device in which the seven-segment data is stored

Explanation:

The values of the lower 4 bits (b0~b3) in the source device specified by **S** are decoded as the seven-segment data stored in **D**.

Example:

When X0.0 is ON, the values of b0~b3 in D0 are decoded as the seven-segment data stored in Y0.0~Y0.15. If the data in the source device exceeds four bits, the values of the lower 4 bits are decoded.

The relation between the seven-segment data and the bit pattern of source data is presented in the following table.

Hex	Bit pattern	Assignment of segments	Segment state							Display
			B0(a)	B1(b)	B2(c)	B3(d)	B4(e)	B5(f)	B6(g)	
0	0000		ON	ON	ON	ON	ON	ON	OFF	
1	0001		OFF	ON	ON	OFF	OFF	OFF	OFF	
2	0010		ON	ON	OFF	ON	ON	OFF	ON	
3	0011		ON	ON	ON	ON	OFF	OFF	ON	
4	0100		OFF	ON	ON	OFF	OFF	ON	ON	
5	0101		ON	OFF	ON	ON	OFF	ON	ON	
6	0110		ON	OFF	ON	ON	ON	ON	ON	
7	0111		ON	ON	ON	OFF	OFF	ON	OFF	
8	1000		ON	ON	ON	ON	ON	ON	ON	
9	1001		ON	ON	ON	ON	OFF	ON	ON	
A	1010		ON	ON	ON	OFF	ON	ON	ON	
B	1011		OFF	OFF	ON	ON	ON	ON	ON	
C	1100		ON	OFF	OFF	ON	ON	ON	OFF	
D	1101		OFF	ON	ON	ON	ON	OFF	ON	
E	1110		ON	OFF	OFF	ON	ON	ON	ON	
F	1111		ON	OFF	OFF	OFF	ON	ON	ON	

API	Instruction code			Operand	Function
1205	D	SORT	P	**S** · **m_1** · **m_2** · **D** · **n**	Sorting the data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S								●								
m_1								●					○	○		
m_2								●					○	○		
D								●								
n								●					○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
m_1		●	●		●	●	●						
m_2		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Initial device in which the original data is stored

m_1 : Number of rows of data

m_2 : Number of columns of data

D : Initial device in which the sorted data is stored

n : Reference value involved in the sorting of the data

Explanation:

1. The data which is sorted is stored in the m1×m2 registers starting from the register specified by **D**. If **S** and **D** specify the same register, the sorted data is the same as the original data in the register specified by **S**.

2. The operand **m_1** should be within the range between 1 and 32. The operand **m_2** should be within the range between 1 and 6. The operand **n** should be within the range between 1 and **m_2**.

3. When SM604 is OFF, the data is sorted in ascending order. When SM604 is ON, the data is sorted in descending order.

4. It is suggested to use pluse type instruction instead of sorting repeatly.

5. Only the 32-bit instruction can use the 32-bit counter, but not the device E.

Example:

1. Suppose SM604 is OFF. When X0.0 is switched from OFF to ON, the data is sorted in ascending order.

2. The data which will be sorted is shown below.

m_2 columns of data

Column

m_1 rows of data

Column / Row	1	2	3	4	5
	Student number	Chinese	English	Math	Physics
1	(D0) 1	(D5) 90	(D10) 75	(D15) 66	(D20) 79
2	(D1) 2	(D6) 55	(D11) 65	(D16) 54	(D21) 63
3	(D2) 3	(D7) 80	(D12) 98	(D17) 89	(D22) 90
4	(D3) 4	(D8) 70	(D13) 60	(D18) 99	(D23) 50
5	(D4) 5	(D9) 95	(D14) 79	(D19) 75	(D24) 69

3. When the value in D100 is 3, the data is sorted as follows.

m_2 columns of data

Column

Row \ Column	1	2	3	4	5
	Student number	**Chinese**	**English**	**Math**	**Physics**
1	(D50) 4	(D55) 70	(D60) 60	(D65) 99	(D70) 50
2	(D51) 2	(D56) 55	(D61) 65	(D66) 54	(D71) 63
3	(D52) 1	(D57) 90	(D62) 75	(D67) 66	(D72) 79
4	(D53) 5	(D58) 95	(D63) 79	(D68) 75	(D73) 69
5	(D54) 3	(D59) 80	(D64) 98	(D69) 89	(D74) 90

m_1 rows of data

4. When the value in D100 is 5, the data is sorted as follows.

m_2 columns of data

Column

Row \ Column	1	2	3	4	5
	Student number	**Chinese**	**English**	**Math**	**Physics**
1	(D50) 4	(D55) 70	(D60) 60	(D65) 99	(D70) 50
2	(D51) 2	(D56) 55	(D61) 65	(D66) 54	(D71) 63
3	(D52) 5	(D57) 95	(D62) 79	(D67) 75	(D72) 69
4	(D53) 1	(D58) 90	(D63) 75	(D68) 66	(D73) 79
5	(D54) 3	(D59) 80	(D64) 98	(D69) 89	(D74) 90

m_1 rows of data

Additional remark:

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If $\mathbf{m_1}$, $\mathbf{m_2}$, or **n** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1206		ZRST	P	D_1 · D_2	Resetting the zone

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D_1		●	●	●	●	●	●	●		○	○	○				
D_2		●	●	●	●	●	●	●		○	○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D_1	●	●			●	●							
D_2	●	●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

D_1 : Initial device which is reset

D_2 : Final device which is reset

Explanation:

1. When the instruction is executed, the values in D_1~D_2 are cleared. The device type should be the same to perform this instruction.

2. When the device number of D_1 is larger than the device number of D_2, only D_2 is reset.

3. The instruction ZRST can use the 32-bit counter.

Example:

1. When X0.0 is ON, the auxiliary relays M300~M399 are reset to OFF.

2. When X1.0 is ON, the 16-bit counters C0~C127 are reset. (The values of C0~C127 are cleared to 0, and the contact and the coil are reset to OFF.)

3. When X2.0 is ON, the stepping relays S0~S127 are reset to OFF.

4. When X3.0 is ON, the output relays Y0.0~Y1.15 are reset to OFF.

5. When X4.0 is ON, the 32-bit counters HC0~HC63 are reset. (The values of HC0~HC63 are cleared to 0, and the contact and the coil are reset to OFF.)

6. When X5.0 is ON, the timers T0~T127 are reset. (The values of T0~T127 are cleared to 0. and the contact and the coil are reset to OFF.)

Additional remark:

1. If $\mathbf{D_1}$ and $\mathbf{D_2}$ are different types of devices, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2007.

2. If $\mathbf{D_1}$ and $\mathbf{D_2}$ contain different formats of data, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2007.

API	Instruction code			Operand	Function
1207	D	BON	P	**S · D · n**	Checking the state of the bit

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○	○	○		
D		●	●	●				●		○						
n	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D	●												
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source device

D : Device in which the check result is stored

n : Bit whose state is judged

Explanation:

1. The state of the **n**th bit in **S** is checked, and the result is stored in **D**.

2. The operand **n** used in the 16-bit instruction should be within the range between 0 and 15, and the operand **n** used in the 32-bit instruction should be within the range between 0 and 31.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example:

1. When X0.0 is ON, Y0.1 is ON if the value of the 15th bit in D0 is 1. When X0.0 is ON, Y0.1 is OFF if the value of the 15th bit in D0 is 0.

2. When X0.0 is switched OFF, the state of Y0.1 remains the same as that before X0.0's being OFF.

Additional remark:

If **n** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1208	D	MEAN	P	**S · D · n**	Mean

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●							
D		●			●	●	●	●			○	○				
n	●	●			●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Initial device

D : Device in which the mean is stored

n : Number of devices

Explanation:

1. After the values in the **n** devices starting from the device specified by **S** are added up, the mean of the sum is stored in **D**.

2. If a remainder appears in the calculation, it is left out.

3. The operand **n** used in the 16-bit instruction should be within the range between 1 and 256

4. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example:

When X0.0 is ON, the values in the three registers starting from D0 are added up. After the values are added up, the sum is divided by 3. The quotient is stored in D10, and the remainder is left out.

Additional remark:

1. If the operand **n** used in the 16-bit instruction is less than 1 or larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If the operand **n** used in the 32-bit instruction is less than 1 or larger than 128, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1209		CCD	P	**S**、**D**、**n**	Sum check

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●							
D		●			●	●		●								
n	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Initial device

D : Device in which the sum is stored

n : Number of pieces of data

6

Explanation:

1. In communication, the sum check is used to compare checksums on the same data on different occasions or on different representations of the data in order to verify the data integrity.

2. The 16-bit conversion mode: When SM606 is OFF, the working mode of the instruction is the 16-bit conversion mode. The **n** pieces of data in the registers starting from the register specified by **S** (eight bits as a group) are added up. The sum is stored in the register specified by **D**, and the values of the parity bits are stored in **D**+1.

3. The 8-bit conversion mode: When SM606 is ON, the working mode of the instruction is the 8-bit conversion mode. The **n** pieces of data in the registers starting from the register specified by **S** (Eight bits forms a group, and only low eight bits are valid.) are added up. The sum is stored in the register specified by **D**, and the values of the parity bits are stored in **D**+1.

4. The operand **n** should be within the range between 1 and 256.

Example 1:

1. When SM606 is OFF, the working mode of the instruction is the 16-bit conversion mode.

2. When X0.0 is ON, the six pieces of data in D0~D2 (eight bits as a group) are added up. The sum is stored in D100, and the values of the parity bits are stored in D101.

S	Data
D0 Low	100 = 0 1 1 0 0 1 0 0
D0 High	111 = 0 1 1 0 1 1 1 1
D1 Low	120 = 0 1 1 1 1 0 0 0
D1 High	202 = 1 1 0 0 1 0 1 0
D2 Low	123 = 0 1 1 1 1 0 1 1
D2 High	211 = 1 1 0 1 0 0 1 1
D100	867
D101	0 0 0 1 0 0 0 1

Sum

The parity bit is set to 1 if the number of ones is odd.
The parity bit is set to 0 if the number of ones is even.

D100	0	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1
D101	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Parity bits

Example 2:

1. When SM606 is ON, the working mode of the instruction is the 8-bit conversion mode.

2. When X0.0 is ON, the six pieces of data in D0~D5 (eight bits as a group) are added up. The sum is stored in D100, and the values of the parity bits are stored in D101.

S	Data
D0 Low	100 = 0 1 1 0 0 1 0 0
D1 Low	111 = 0 1 1 0 1 1 1 1
D2 Low	120 = 0 1 1 1 1 0 0 0
D3 Low	202 = 1 1 0 0 1 0 1 0
D4 Low	123 = 0 1 1 1 1 0 1 1
D5 Low	211 = 1 1 0 1 0 0 1 1
D100	867
D101	0 0 0 1 0 0 0 1

Sum

The parity bit is set to "1" if the number of ones is odd.
The parity bit is set to "0" if the number of ones is even.

D100	0	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1
D101	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Parity bits

6

Additional remark:

1. Suppose SM606 is ON. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. Suppose SM606 is OFF. If **S**+**n**/2-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If **n** is less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

API	Instruction code			Operand	Function
1210	D	ABS	P	**D**	Absolute value

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

D : Device involved in the getting of the absolute value

Explanation:

1. When the instruction ABS is executed, the absolute value of the value in the device specified by **D** is gotten.

2. Generally, the pulse instruction ABSP is used.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example:

Suppose the value in D0 before the execution of the instruction is -1234. When X0.0 is switched from OFF to ON, the absolute value of -1234 in D0 is gotten. That is, the value in D0 becomes 1234 after the instruction is executed.

API	Instruction code			Operand	Function
1211		MINV	P	**S · D · n**	Inverting the matrix bits

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●							
D		●			●	●		●								
n	●	●			●	●		●	●		○		○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Matrix source

D : Operation result

n : Length of the array

Explanation:

1. The bits in the **n** devices starting from the device specified by **S** are inverted, and the inversion result is stored in **D**.

2. The operand n should be within the range between 1 and 256.

Example:

When X0.0 is ON, the bits in the three 16-bit registers D0~D2 are inverted, and the inversion result is stored in the 16-bit registers D20~D22.

	b15															b0
D0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
D1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
D2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

After the instruction is executed

	b15															b0
D20	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
D21	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
D22	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Additional remark:

1. If **S**+**n**-1 or **D**+**n**-1 exceeds the device range, the instruction is not execute, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1212		MBRD	P	**S · n · D**	Reading the matrix bit

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●							
n	●	●			●	●		●	●		○		○	○		
D		●			●	●		●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Matrix source

n : Length of the array

D : Pointer

6

Explanation:

1. When the instruction is executed, the state of SM613 is checked. If SM613 is ON, the value of the pointer **D** is cleared to 0. The value of the bit specified by the value of the pointer **D** is read into SM614. After the value of the bit is read, the state of SM612 is checked. If SM612 is ON, the value of the pointer **D** increases by one.

2. When the value of the last bit is read, SM608 is ON, and the bit number is recorded in the pointer **D**.

3. The operand **n** should be within the range between 1 and 256.

4. The value of the pointer is specified by users. The values range from 0 to 16**n**–1, and correspond to the range from b0 to b16**n**–1. If the value of the pointer exceeds the range, SM611 is set to 1, and the instruction is not executed.

Example:

1. Suppose SM613 is OFF and SM612 is ON when X0.0 is switched from OFF to ON.

2. Suppose the current value in D20 is 45. When X0.0 is switched from OFF to ON three times, users can get the following execution results.

❶ The value in D20 is 46, SM614 is OFF, and SM608 is OFF.

❷ The value in D20 is 47, SM614 is ON, and SM608 is OFF.

❸ The value in D20 is 47, SM614 is OFF, and SM608 is ON.

Additional remark:

1. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. The flags:

   SM608: The matrix comparison comes to an end. When the last bits are compared, SM608 is ON.

   SM611: It is the matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.

   SM612: It is the matrix pointer increasing flag. The current value of the pointer increases by one.

   SM613: It is the matrix pointer clearing flag. The current value of the pointer is cleared to zero.

   SM614: It is the carry flag for the matrix rotation/shift/output.

API	Instruction code			Operand	Function
1213		MBWR	P	**S · n · D**	Writing the matrix bit

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S		●			●	●		●								
n	●	●			●	●		●	●		○		○	○		
D		●			●	●		●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Matrix source

n : Length of the array

D : Pointer

Explanation:

6

1. When the instruction is executed, the state of SM613 is checked. If SM613 is ON, the value of the pointer **D** is cleared to 0. The state of SM615 is written into the bit specified by the value of the pointer **D**. After the state of SM615 is written into the bit, the state of SM612 is checked. If SM612 is ON, the value in the pointer **D** increases by one.

2. When the state of SM615 is written into the last bit, SM608 is ON, and the bit number is recorded in the pointer **D**. If value of the pointer **D** exceeds the range, SM611 is ON.

3. The operand **n** should be within the range between 1 and 256.

4. The value of the pointer is specified by users. The values range from 0 to 16**n**–1, and correspond to the range from b0 to b16**n**–1. If the value of the pointer exceeds the range, SM611 is set to 1, and the instruction is not executed.

Example:

1. Suppose SM613 is OFF and SM612 is ON when X0.0 is switched from OFF to ON.

2. Suppose the current value in D20 is 45. When X0.0 is switched from OFF to ON one time, users can get the execution result shown below. When the value in D20 is 45, SM615 is OFF, and SM608 is OFF.

Additional remark:

1. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. The flags:

   SM608: The matrix comparison comes to an end. When the last bits are compared, SM608 is ON.

   SM611: It is the matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.

   SM612: It is the matrix pointer increasing flag. The current value of the pointer increases by one.

   SM613: It is the matrix pointer clearing flag. The current value of the pointer is cleared to zero.

   SM615: It is the borrow flag for the matrix shift/output.

API	Instruction code			Operand	Function
1214		MBC	P	**S · n · D**	Counting the bits with the value 0 or 1

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S		●			●	●		●								
n	●	●			●	●		●	●		○		○	○		
D		●			●	●		●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Matrix source

n : Length of the array

D : Operation result

Explanation:

6

1. The instruction is used to count the bits with the value 1 or 0 in the **n** devices starting from the device specified by **S**. The operation result is stored in **D**.

2. When SM617 is ON, the bits with the value 1 is counted. When SM617 is OFF, the bits with the value 0 is counted. When the operation result is 0, SM618 is ON.

3. The operand **n** should be within the range between 1 and 256.

Example:

Suppose SM617 is ON. When X0.0 is ON, the bits with the value 1 are counted, and the operation result is stored in D20.
Suppose SM617 is OFF. When X0.0 is ON, the bits with the value 0 are counted, and the operation result is stored in D20.

Additional remark:

1. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. The flags:

   SM617: The bits with the value 0 or 1 are counted.

   SM618: It is ON when the matrix counting result is 0.

API	Instruction code			Operand	Function
1215		DIS	P	**S · n · D**	Disuniting the 16-bit data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●		○	○				
n	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Data source

n : Number of devices

D : Operation result

6

Explanation:

1. The 16-bit value in the register specified by **S** is divided into four groups (four bits as a group), and these groups are stored in the low four bits in every register (The registers range from **D** to **D**+(**n**-1).).

2. The operand **n** should be within the range between 1 and 4.

Example:

Suppose the value in D0 is 16#1234. When M0 is enabled, the instruction DIS is executed. The value in D0 is divided into four groups (four bits as a group), and these groups are stored in the low four bits in every register (The registers range from D10 to D13.).

Additional remark:

1. If **D**~**D**+(**n-1**) exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1216		UNI	P	**S · n · D**	Uniting the 16-bit data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●							
n	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Data source

n : Data length

D : Operation result

6

Explanation:

1. The 16-bit values in the registers specified by **S**~**S**+(**n**-1) are divided into groups (four bits as a group), and every group which is composed of b0~b3 is stored in the register specified by **D** (b0~b15).

2. The operand **n** should be within the range between 1 and 4.

Example:

Suppose the values in D0~D3 are 16#1234, 16#5678, 16#8765, and 16#4321 respectively. When M0 is enabled, the instruction UNI is executed. The values in D0~D3 are divided into groups (four bits as a group), and every group which is composed of b0~b3 is stored in D10(b0~b15).

Additional remark:

1. If **S**~**S**+(**n**-1) exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is less than 1, or if **n** is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1217	D	WSUM	P	**S · n · D**	Getting the sum

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●							
n	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
n		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Data source

n : Data length

D : Operation result

Explanation:

1. The signed decimal values in **S**~**S**+**n**-1 are added up, and the sum is stored in the register specified by **D**.

2. The operand **n** used in the 16-bit instruction should be within the range between 1 and 256.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example:

When the instruction WSUM is executed, the values in D0~D2 are added up, and the sum is stored in D10.

Additional remark:

1. If **n** used in the 16-bit instruction is less than 1 or larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If **n** used in the 32-bit instruction is less than 1 or larger than 128, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If **S**+**n**-1 or **D** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the operand **D** used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be DWORD or ARRAY [2] of WORD.

5. If the operand **D** used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [2] of DWORD.

API	Instruction code			Operand	Function
1221	D	LIMIT	P	**S_1 · S_2 · S_3 · D**	Confining the value within the bounds

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
S_3	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Minimum output value

S_2 : Maximum output value

S_3 : Input value

D : Output value

Explanation:

1. The input value in **S_3** is compared with the minimum output value in **S_1** and the maximum output value in **S_2**, and the comparison result is stored in **D**.

   If the minimum output value in **S_1** is larger than the input value in **S_3**, the output value stored in **D** is equal to the minimum output value in **S_1**.

   If the maximum output value in **S_2** is less than the input value in **S_3**, the output value stored in **D** is equal to the maximum output value in **S_2**.

   If the input value in **S_3** is within the range between the minimum output value in **S_1** and the maximum output value in **S_2**, the output value stored in **D** is equal to the input value in **S_3**.

If the minimum output value in **S_1** is larger than the maximum output value in **S_2**, the instruction is not executed.

2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example:

- When X0.0 is ON, the state of X1 is converted into the binary value, and the conversion result is stored in D0. Besides, the value stored in D0 is compared with 500 and 5000, and the comparison result is stored in D1.

Minimum output value	Maximum output value	Output value in D0	Function	Output value in D1
500	5000	499	D0<500	500
		5001	D0>5000	5000
		600	500≦D0≦5000	600

Additional remark:

If the minimum output value in **S_1** is larger than the maximum output value in **S_2**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1222	D	BAND	P	**S_1 · S_2 · S_3 · D**	Deadband control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
S_3	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Minimum value of the deadband

S_2 : Maximum value of the deadband

S_3 : Input value

D : Output value

Explanation:

1. The minimum value of the deadband in **S_1** or the maximum value of the deadband in **S_2** is subtracted from the input value in **S_3**, and the difference is stored in **D**.

   If the minimum value of the deadband in **S_1** is larger than the input value in **S_3**, the minimum value of the deadband in **S_1** is subtracted from the input value in **S_3**, and the difference is stored in **D**.

   If the maximum value of the deadband in **S_2** is less than the input value in **S_3**, the maximum value of the deadband in **S_2** is subtracted from the input value in **S_3**, and the difference is stored in **D**.

If the input value in **S_3** is within the range between the minimum of the deadband in **S_1** and the maximum value of the deadband in **S_2**, the output value stored in **D** is 0.

If the minimum value of the deadband in **S_1** is larger than the maximum value of the deadband in **S_2**, the instruction is not executed.

2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

3. The figures:

4. The minimum value of the deadband in **S_1**, the maximum value of the deadband in **S_2**, the input value in **S_3**, and the output value in **D** should be within the range described below.

- If the instruction BAND is executed, the minimum value of the deadband in **S_1**, the maximum value of the deadband in **S_2**, the input value in **S_3**, and the output value in **D** is within the range between -32768 and 32767. Suppose the minimum value of the deadband in **S_1** is 10 and the maximum value of the deadband in **S_3** is -32768. The output value in **D** is calculated as follows.

  Output value in **D**=-32768-10=16#8000-16#000A=16#7FF6=32758

- If the instruction DBAND is executed, the minimum value of the deadband in **S_1**, the maximum value of the deadband in **S_2**, the input value in **S_3**, and the output value in **D** is within the range between -2147483648 and 2147483647. Suppose the minimum value of the deadband in (**S_1**+1, **S_1**) is 1000 and the maximum value of the deadband in (**S_3**+1, **S_3**) is -2147483648. The output value in (**D**+1, **D**) is calculated as follows.

  Output value in (**D**+1, **D)**

  =-2147483648-1000=16#80000000-16#000003E8=16#7FFFFC18

  =2147482648

Example 1:

When X0.0 is ON, -1000 or 1000 is subtracted from the binary-coded decimal value in X1, and the difference is stored in D1.

The execution results:

Minimum value of the deadband	Maximum value of the deadband	Input value in D0	Function	Output value in D1
-1000	1000	-1200	D0<-1000=>D1=D0- (-1000)	-200
		1200	D0>1000=>D1=D0-1000	200
		500	-1000≦D0≦1000=>D0=0	0

Example 2:

When X0.0 is ON, -10000 or 10000 is subtracted from the binary-coded decimal value in (X2, X1), and the difference is stored in (D11, D10).

The execution results:

Minimum value of the deadband	Maximum value of the deadband	Input value in (D1, D0)	Function	Output value in (D11, D10)
-10000	10000	-12000	(D1 · D0) <-10000 => (D11 · D10) = (D1 · D0) - (-10000)	-2000
		12000	(D1 · D0) >10000 => (D11 · D10) = (D1 · D0) -10000	2000
		5000	-10000≦ (D1 · D0) ≦10000 => (D1 · D0) =0	0

Additional remark:

If the minimum value of the deadband in S1 is larger than the maximum value of the deadband in S2, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1223	D	ZONE	P	S_1 · S_2 · S_3 · **D**	Controlling the zone

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1	●	●			●	●	●	●	●		○	○	○	○		
S_2	●	●			●	●	●	●	●		○	○	○	○		
S_3	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S_1 : Negative deviation

S_2 : Positive deviation

S_3 : Input value

D : Output value

Explanation:

1. The negative deviation in S_1 or the positive deviation in S_2 is added to the input value in S_3, and the sum is stored in **D**.

   If the input value in S_3 is less than 0, the negative deviation in S_1 is added to the input value in S_3, and the sum is stored in **D**.

   If the input value in S_3 is larger than 0, the positive deviation in S_2 is added to the input value in S_3, and the sum is stored in **D**.

   If the input value in S_3 is equal to 0, the output value stored in **D** is 0.

2. The figures:

3. Only the 32-bit instructions can use the 32-bit counter but not the device E.

4. The negative deviation in S_1, the positive deviation in S_2, the input value in S_3, and the output value in **D** should be within the range described below.

   - If the instruction ZONE is executed, the negative deviation in S_1, the positive deviation in S_2, the input value in S_3, and the output value in **D** is within the range between -32768 and 32767. Suppose the negative deviation in S_1 is -100 and the input value in S_3 is -32768. The output value in **D** is calculated as follows.

     Output value in **D**=(-32768)+(-100)=16#8000+16#FF9C=16#7F9C=32668

   - If the instruction DZONE is executed, the negative deviation in S_1, the positive deviation in S_2, the input value in S_3, and the output value in **D** is within the range between -2147483648 and 2147483647. Suppose the negative deviation in (S_1+1, S_1) is -1000 and the input value in (S_3+1, S_3) is -2147483648. The output value in (**D**+1, **D**) is calculated as follows.

     Output value in (**D**+1, **D)**

     =-2147483648+(-1000)=16#80000000+16#FFFFFC18=16#7FFFFC18=2147482648

Example 1:

When X0.0 is ON, -100 or 100 is added to the binary-coded decimal value in X1, and the sum is stored in D10.

The execution results:

Negative deviation	Positive deviation	Input value in D0	Function	Output value in D10
-100	100	-10	D0<0=>D10= (-10) + (-100)	-110
		0	D0=0=>D10=0	0
		50	D0>0=>D10=50+100	150

Example 2:

When X0.0 is ON, -10000 or 10000 is added to the binary-coded decimal value in (X2, X1), and the sum is stored in (D11, D10).

Negative deviation	Positive deviation	Input value in (D1, D0)	Function	Output value in (D11, D10)
-10000	10000	-10	(D1 · D0) <0 => (D11 · D10) = (-10) + (-10000)	-10010
		0	(D1 · D0) =0 => (D11 · D10) =0	0
		50	(D1 · D0) >0 => (D11 · D10) =50+10000	10050

API	Instruction code			Operand	Function
1224		FMEAN	P	**S · D · n**	The mean of the floating points

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S							●	●								
n							●	●					○	○		
D							●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
n			●				●						
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Initial device

D : Device in which the mean is stored

n : Number of devices

Explanation:

1. After the single precision floating points in the **n** devices starting from the device specified by **S** are added up, the mean of the sum is stored in **D**.

2. The operand **n** used in the 16-bit instruction should be within the range between 1 and 256,

3. The flags: SM600 (zero flag), SM601 (borrow flag), SM602 (carry flag):

   - When the operation result is zero, SM600 is ON. Otherwise, it is OFF.

   - If the value while adding or the absolute result of the operation is less than the floating point number can be shown, the D=16#FF800000 and the borrow flag SM601 is ON.

   - If the value while adding or the absolute result of the operation is larger than the floating point number can be shown, the D=16#7F800000 and the carry flag SM602 is ON.

Example:

When X0.0 is ON, add the values of the 3 single precision floating points in (D1, D0), (D3, D2), (D5, D4) and then divide the addition result by 3, after that store the result in (D11, D10).

[(D1, D0)+(D3, D2)+(D5, D4)]/3 ⟶ (D11, D10)

Additional remark:

1. If the operand **n** used is less than 1 or larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If **S**+2***n**-1exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in S exceeds the range of the floating point number can be shown, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1225		FSUM	P	**S · n · D**	The sum of the floating points

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S							●	●								
n							●	●					○	○		
D							●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
n			●				●						
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Data source

n : Data length

D : Operation result

Explanation:

1. After the single precision floating points in the **n** devices starting from the device specified by **S** are added up, the sum is stored in **D**.

2. The operand **n** used in the 16-bit instruction should be within the range between 1 and 256,

3. The flags: SM600 (zero flag), SM601 (borrow flag), SM602 (carry flag):

   - When the operation result is zero, SM600 is ON. Otherwise, it is OFF.

   - If the value while adding or the absolute result of the operation is less than the floating point number can be shown, the D=16#FF800000 and the borrow flag SM601 is ON.

   - If the value while adding or the absolute result of the operation is larger than the floating point number can be shown, the D=16#7F800000 and the carry flag SM602 is ON.

Example:

When the instruction FSUM is executed, the values of the 3 single precision floating points in (D1, D0), (D3, D2), (D5, D4) will be added up and the result will be stored in (D11, D10).

Additional remark:

1. If the operand **n** used is less than 1 or larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If **S**+2*__n__-1exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in S exceeds the range of the floating point number can be shown, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

6.14 Structure Creation Instructions

6.14.1 List of Structure Creation Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1300	FOR	–	–	Start of the nested loop
1301	NEXT	–	–	End of the nested loop
1302	BREAK	–	–	Terminating the FOR-NEXT loop

6.14.2 Explanation of Structure Creation Instructions

API	Instruction code			Operand	Function
1300		FOR		**S**	Start of the nested loop

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●			●	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S : Number of times the loop is executed repeatedly

Explanation:

Please refer to the instruction API1301 NEXT for more details.

API	Instruction code			Operand	Function
1301		NEXT		-	End of the nested loop

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

Explanation:

1. The program between FOR and NEXT is executed N times. After the program between FOR and NEXT is executed N times, the program follows NEXT is executed. The instruction FOR specifies the number of times the program between FOR and NEXT is executed.

2. N should be within the range between 1 and 32,767. If N is less than 1, it is count as 1.

3. If the program between FOR and NEXT is not executed, it can be skipped by the use of the instruction CJ.

4. The following conditions result in errors.

   - The instruction NEXT is prior to the instruction FOR.
   - The instruction FOR exists, but the instruction NEXT does not exist.
   - The instruction NEXT follows the instruction FEND or END.
   - The number of times the instruction FOR is used is different from the number of times the instruction NEXT is used.

5. FOR/NEXT supports the nested program structure. There are at most 32 levels of nested program structures. If the loop is executed many times, it takes more time for the program in the PLC to be scanned, and the watchdog timer error will occur. Users can use the instruction WDT to resolve the problem.

Example 1:

After program A is executed three times, the program follows the instruction NEXT is executed. Program B is executed four times every time program is executed. Therefore, program B is executed twelve times in total.

Example 2:

When X0.0 is OFF, the program between FOR and NEXT is executed. When X0.0 is ON, the instruction CJ is executed. The execution of the program jumps to LABEL 1:, i.e. network 6, and network 4~network 5 are not executed.

Example 3:

If the program between FOR and NEXT is not executed, it can be skipped by the use of the instruction CJ. When X0.1 in network 8 is ON, the instruction CJ is executed. The execution of the program jumps to LABEL 1:, i.e. network 12, and network 9~network 11 are not executed.

NETWORK 4
FOR
2
S
NETWORK 5
X0.0
INC
En
D2
D
NETWORK 6
FOR
4
S
NETWORK 7
X0.0
WDT
En
INC
En
D3
D
NETWORK 8
X0.1
CJ
En
LABEL1
S
NETWORK 9
FOR
5
S
NETWORK 10
X0.0
INC
En
D4
D

Additional remark:

Please refer to ISPSoft User Manual for more information related to the usage of the label.

API	Instruction code			Operand	Function
1302		BREAK		**D**	Terminating the FOR-NEXT loop

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
D		●			●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

D : Device in which the remaining number of times the loop can be executed is stored

Explanation:

1. The instruction BREAK is used to terminate the FOR/NEXT loop. The remaining number of times the FOR/NEXT loop can be repeated is stored in **D**, and the execution of the program jumps to the instruction NEXT and execute the next instruction.

2. When the instruction BREAK is executed, the remaining number of times the FOR/NEXT loop can be repeated is stored in **D**, including this time the instruction BREAK is executed.

3. When the instruction BREAK is executed for the first time to terminate the FOR/NEXT loop, the execution of the program cannot jump to the instruction NEXT to execute the next instruction. But if the instruction BREAK is executed for more than one time to terminate the FOR/NEXT loop, the execution of the program can jump to the instruction NEXT to execute the next instruction.

Example:

When the FOR/NEXT loop is executed, 1 is added to the value in D0. When the value in D0 is equal to 30, the FOR/NEXT loop is terminated, and the remaining number of times the FOR/NEXT loop can be repeated, i.e. 71, is stored in D10. The execution of the program jumps to LABEL 1:, i.e. network 6, and 1 is added to the value in D2.

Additional remark:

1. If the instruction BREAK is outside the FOR/NEXT loop, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2017.

2. Please refer to ISPSoft User Manual for more information related to the usage of the label.

6.15 Module Instructions

6.15.1 List of Module Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1400	FROM	DFROM	✓	Reading the data from the control register in the extension module
1401	TO	DTO	✓	Writing the data into the control register in the extension module

6.15.2 Explanation of Module Instructions

API	Instruction code			Operand	Function
1400	D	FROM	P	m_1 . m_2 . m_3 , D_1 , D_2 , n	Reading the data from the control register in the extension module

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
m_1					●	●	●	●	●		○	○	○	○		
m_2					●	●	●	●	●		○	○	○	○		
m_3					●	●	●	●	●		○	○	○	○		
D_1					●	●	●	●								
D_2					●	●	●	●								
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
m_1		●	●		●	●	●						
m_2		●	●		●	●	●						
m_3		●	●		●	●	●						
D_1		●	●		●	●	●						
D_2		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

m_1 : Numbers of the CPU module or the remote extension module

m_2 : The order numbers of the extension number

m_3 : Control register number

D_1 : Device in which the data is stored

D_2 : Device in which the error code is stored

n : Data length

Explanation:

1. Users can use this instruction to read the data from the control register in the extension module.

2. The operand m_1 should be within the range between 0 and 16. 0 represents the CPU module, and 1~16 represent the extension modules.

3. The operand $\mathbf{m_2}$ represents the number of the right-side extension modules that is connected to the CPU module or the remote modules. The first device is number 1, the second device is number 2 and so on. Any type of the connected modules will be counted and up to 32 devices can be connected.

4. The operand $\mathbf{m_3}$ specifies the control register number.

5. When the instruction FROM is executed, $\mathbf{D_2}$ is set to 0. When an error occurs, $\mathbf{D_2}$ is not set to 0. Please refer to the additional remark below for more information about the error codes. When the instruction is not executed, $\mathbf{D_2}$ will not be specified by any error code.

6. The operand $\mathbf{m_1}$ should be within the range between 1 and 8.

7. Only the 32-bit instructions can use the 32-bit counter, but not he device E.

Example:

When X0.0 is switched from OFF to ON, the instruction FROM is executed. Read the data stored in CR#2 from the right side of the first module and store the data in D100. Owing to the fact that no error occurs, the code stored in D110 is 16#0000.

Additional remark:

1. If the values in $\mathbf{m_1}$ and $\mathbf{m_2}$ exceed the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If $\mathbf{D_1}$~$\mathbf{D_1}$+**n**-1 exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in **n** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. Due to the fact that the use of the instruction FROM decreases the execution efficiency of the CPU module and that of the I/O module, users are suggested to use pulse type instruction to perform a single trigger as the example shown above.

5. When the instruction is executed, if there is any error responded from modules, the error code will be stored in **D**$_2$.

   The descriptions of the error codes are shown below:

Error code	Description
16#1400	Reading the data from the control register (CR) in the module but no such CR number.
16#1401	The value inputted in the module is invalid.
16#1402	The module is not responding. Communication timeout.

API	Instruction code			Operand	Function
1401	D	TO	P	m_1 . m_2 · m_3 · **S** · **D** · **n**	Writing the data into the control register in the extension module

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
m_1					●	●	●	●	●		○	○	○	○		
m_2					●	●	●	●	●		○	○	○	○		
m_3					●	●	●	●	●		○	○	○	○		
S					●	●	●	●	●				○	○		
D					●	●	●	●								
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
m_1		●	●		●	●	●						
m_2		●	●		●	●	●						
m_3		●	●		●	●	●						
S		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

6

Symbol:

m_1 : Numbers of the CPU module or the remote extension module

m_2 : The order numbers of the extension number

m_3 : Control register number

S : Device in which the data is stored

D : Device in which the error code is stored

n : Data length

Explanation:

1. Users can use this instruction to write the data to the control register in the extension module.

2. The operand m_1 should be within the range between 0 and 16. 0 represents the CPU module, and 1~16 represent

the extension modules.

3. The operand **m_2** represents the number of the right-side extension modules that is connected to the CPU module or the remote modules. The first device is number 1, the second device is number 2 and so on. Any type of the connected modules will be counted and up to 32 devices can be connected.

4. The operand **m_3** specifies the control register number.

5. When the instruction FROM is executed, **D_2** is set to 0. When an error occurs, **D_2** is not set to 0. Please refer to the additional remark below for more information about the error codes. When the instruction is not executed, **D_2** will not be specified by any error code.

6. The operand **m_1** should be within the range between 1 and 8.

7. Only the 32-bit instructions can use the 32-bit counter, but not he device E.

8. When **S** is a hexadecimal value, **n** hexadecimal values are transmitted to the I/O module. Suppose **S** is 16#0001 and **n** is 3. Three 16#0001s are transmitted to the I/O module.

Example:

When X0.0 is switched from OFF to ON, the instruction TO is executed. Write the data stored in D100 to CR#2 in the right side of the first module. Owing to the fact that no error occurs, the code stored in D110 is 16#0000.

Additional remark:

1. If the values in **m_1** and **m_2** exceed the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **D_1~D_1+n**-1 exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in **n** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. Due to the fact that the use of the instruction TO decreases the execution efficiency of the CPU module and that of the I/O module, users are suggested to use pulse type instruction to perform a single trigger as the example shown above.

5. When the instruction is executed, if there is any error responded from modules, the error code will be stored in $\mathbf{D}_2$. The descriptions of the error codes are shown below:

Error code	Description
16#1400	Reading the data from the control register (CR) in the module but no such CR number.
16#1401	The value inputted in the module is invalid.
16#1402	The module is not responding. Communication timeout.

6.16 Floating-point Number Instructions

6.16.1 List of Floating-point Number Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1500	–	FSIN	✓	Sine of the floating-point number
1501	–	FCOS	✓	Cosine of the floating-point number
1502	–	FTAN	✓	Tangent of the floating-point number
1503	–	FASIN	✓	Arcsine of the floating-point number
1504	–	FACOS	✓	Arccosine of the floating-point number
1505	–	FATAN	✓	Arctangent of the floating-point number
1506	–	FSINH	✓	Hyperbolic sine of the floating-point number
1507	–	FCOSH	✓	Hyperbolic cosine of the floating-point number
1508	–	FTANH	✓	Hyperbolic tangent of the floating-point number
1509	–	FRAD	✓	Converting the degree to the radian
1510	–	FDEG	✓	Converting the radian to the degree
1511	SQR	DSQR	✓	Square root of the binary number
1512	–	FSQR	✓	Square root of the floating-point number
1513	–	FEXP	✓	Exponent of the floating-point number
1514	–	FLOG	✓	Logarithm of the floating-point number
1515	–	FLN	✓	Natural logarithm of the binary floating-point number
1516	–	FPOW	✓	Power of the floating-point number
1517	RAND	–	✓	Random number

6.16.2 Explanation of Floating-point Number Instructions

API	Instruction code			Operand	Function
1500		FSIN	P	**S · D**	Sine of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source value

D : Sine value

Explanation:

1. Whether the source value specified by **S** is a radian or a degree depends on the state of SM695.
2. If SM695 is OFF, the source value specified by **S** is a radian. Radian=Degree×π/180.
3. If SM695 is ON, the source value specified by **S** is a degree.

   Degree=Radian×180/π. ($0^{o} \leqq$ Degree $\leqq 360^{0}$)
4. If the conversion result is 0, SM600 is ON.
5. The sine of the source value specified by **S** is stored in the register specified by **D**.
6. The relation between radians and sine values are shown below.

Example:

When X0.0 is ON, the binary-coded decimal value in X1.15~X1.0 is converted into the binary value, and the conversion result is stored in D0. The binary value in D0 is converted into the floating-point number, and the conversion result is stored in (D11, D10). The floating-point number in (D11, D10) is converted into the radian, and the conversion result is stored in (D21, D20). The sine of the radian in (D21, D20) is stored in (D31, D30), and the sine value is the floating-point number.

Additional remark:

1. If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

2. If SM695 is ON, and the value in S is less than 0 or larger than 360, the instruction is not executed, SM0 is ON, and the error code is 16#2003.

API	Instruction code			Operand	Function
1501		FCOS	P	**S · D**	Cosine of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source value

D : Cosine value

Explanation:

1. Whether the source value specified by **S** is a radian or a degree depends on the state of SM695.

2. If SM695 is OFF, the source value specified by **S** is a radian. Radian=Degree×π/180.

3. If SM695 is ON, the source value specified by **S** is a degree.

   Degree=Radian×180/π. ($0^{o} \leqq$ Degree $\leqq 360^{0}$)

4. If the conversion result is 0, SM600 is ON.

5. The cosine of the source value specified by **S** is stored in the register specified by **D**.

6. The relation between radians and cosine values are shown below.

Example:

When X0.0 is ON, the binary-coded decimal value in X1.15~X1.0 is converted into the binary value, and the conversion result is stored in D0. The binary value in D0 is converted into the floating-point number, and the conversion result is stored in (D11, D10). The floating-point number in (D11, D10) is converted into the radian, and the conversion result is stored in (D21, D20). The cosine of the radian in (D21, D20) is stored in (D31, D30), and the cosine value is the floating-point number.

Additional remark:

1. If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

2. If SM695 is ON, and the value in S is less than 0 or larger than 360, the instruction is not executed, SM0 is ON, and the error code is 16#2003.

API	Instruction code			Operand	Function
1502		FTAN	P	**S · D**	Tangent of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source value

D : Tangent value

Explanation:

1. Whether the source value specified by **S** is a radian or a degree depends on the state of SM695.
2. If SM695 is OFF, the source value specified by **S** is a radian. Radian=Degree×π/180.
3. If SM695 is ON, the source value specified by **S** is a degree.

   Degree=Radian×180/π. ($0^{o} \leqq$ Degree $\leqq 360^{0}$)
4. If the conversion result is 0, SM600 is ON.
5. The tangent of the source value specified by **S** is stored in the register specified by **D**.
6. The relation between radians and tangent values are shown below.

Example:

When X0.0 is ON, the binary-coded decimal value in X1.15~X1.0 is converted into the binary value, and the conversion result is stored in D0. The binary value in D0 is converted into the floating-point number, and the conversion result is stored in (D11, D10). The floating-point number in (D11, D10) is converted into the radian, and the conversion result is stored in (D21, D20). The tangent of the radian in (D21, D20) is stored in (D31, D30), and the tangent value is the floating-point number.

Additional remark:

1. If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

2. If SM695 is ON, and the value in S is less than 0 or larger than 360, the instruction is not executed, SM0 is ON, and the error code is 16#2003.

API	Instruction code			Operand	Function
1503		FASIN	P	**S · D**	Arcsine of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source value

D : Arcsine value

Explanation:

1. Arcsine value=$\sin^{-1}$

   The relation between sine values and arcsine values are shown below.

2. If the conversion result is 0, SM600 is ON.

Example:

When X0.0 is ON, the arcsine of the floating-point number in (D1, D0) is stored in (D11, D10). The arcsine value in (D11, D10) is converted into the degree, and the conversion result is stored in (D21, D20). The degree in (D21, D20) is converted into the integer, and the conversion result is stored in (D31, D30). The integer in (D31, D30) is converted into the binary-coded decimal value, and the conversion result is stored in Y0.15~Y0.0.

Additional remark:

1. The floating-point number specified by the operand **S** should be within the range between –1.0 and +1.0. If the floating-point number is not within the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1504		FACOS	P	**S · D**	Arccosine of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source value

D : Arccosine value

Explanation:

1. Arccosine value=$\cos^{-1}$

   The relation between cosine values and arccosine values are shown below.

2. If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, SM602 is ON.

3. If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, SM601 is ON.

4. If the conversion result is 0, SM600 is ON.

Example:

When X0.0 is ON, the arccosine of the floating-point number in (D1, D0) is stored in (D11, D10). The arccosine value in (D11, D10) is converted into the degree, and the conversion result is stored in (D21, D20). The degree in (D21, D20) is converted into the integer, and the conversion result is stored in (D31, D30). The integer in (D31, D30) is converted into the binary-coded decimal value, and the conversion result is stored in Y0.15~Y0.0.

Additional remark:

1. The floating-point number specified by the operand **S** should be within the range between –1.0 and +1.0. If the floating-point number is not within the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1505		FATAN	P	**S · D**	Arctangent of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source value

D : Arctangent value

Explanation:

1. Arctangent value=$\tan^{-1}$

2. The relation between tangent values and arctangent values are shown below.

3. If the conversion result is 0, SM600 is ON.

Example:

When X0.0 is ON, the arctangent of the floating-point number in (D1, D0) is stored in (D11, D10). The arctangent value in (D11, D10) is converted into the degree, and the conversion result is stored in (D21, D20). The degree in (D21, D20) is converted into the integer, and the conversion result is stored in (D31, D30). The integer in (D31, D30) is converted into the binary-coded decimal value, and the conversion result is stored in Y0.15~Y0.0.

Additional remark:

If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1506		FSINH	P	**S · D**	Hyperbolic sine of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

Explanation:

1. Hyperbolic sine value=$(e^{S}-e^{-S})/2$.

2. If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, the value in **D** is 16#7F800000, and SM602 is ON.

3. If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, the value in **D** is 16#7F800000, and SM601 is ON.

4. If the conversion result is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the hyperbolic sine of the floating-point number in (D1, D0) is stored in (D11, D10). The hyperbolic sine value in (D11, D10) is the floating-point number.

2. If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, SM602 is ON.

3. If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, SM601 is ON.

4. If the conversion result is 0, SM600 is ON.

Additional result:

If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1507		FCOSH	P	**S · D**	Hyperbolic cosine of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source value

D : Hyperbolic cosine value

Explanation:

1. Hyperbolic cosine value=$(e^{S}+e^{-S})/2$.

2. If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, the value in **D** is 16#7F800000, and SM602 is ON.

3. If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, the value in **D** is 16#FF800000, and SM601 is ON.

4. If the conversion result is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the hyperbolic cosine of the floating-point number in (D1, D0) is stored in (D11, D10). The hyperbolic cosine value in (D11, D10) is the floating-point number.

2. If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, SM602 is ON.

3. If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, SM601 is ON.

4. If the conversion result is 0, SM600 is ON.

Additional remark:

If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1508		FTANH	P	**S · D**	Hyperbolic tangent of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source value

D : Hyperbolic tangent value

Explanation:

1. Hyperbolic tangent value=$(e^{S}-e^{-S})/(e^{S}+e^{-S})$.

2. If the conversion result is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the hyperbolic tangent of the floating-point number in (D1, D0) is stored in (D11, D10). The hyperbolic tangent value in (D11, D10) is the floating-point number.

2. If the conversion result is 0, SM600 is ON.

Additional remark:

If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1509		FRAD	P	**S · D**	Converting the degree to the radian

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Data source (degree)

D : Conversion result (radian)

Explanation:

1. The equation below is used to convert degrees into radians.
2. Radian = Degree×(π/180).
3. If the conversion result is 0, SM600 is ON.

Example:

When X0.0 is ON, the degree in (D1, D0) is converted into the radian, and the conversion result is stored in (D11, D10). The radian in (D11, D10) is the floating-point number.

Additional remark:

If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1510		FDEG	P	**S**、**D**	Converting the radian to the degree

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Data source (radian)

D : Conversion result (Degree)

Explanation:

1. The equation below is used to convert radians into degrees.
2. Degree = Radian×(180/π).
3. If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, the value in **D** is 16#7F7FFFFF.
4. If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, the value in **D** is 16#7F7FFFFF.
5. If the conversion result is 0, SM600 is ON.

Example:

When X0.0 is ON, the radian in (D1, D0) is converted into the degree, and the conversion result is stored in (D11, D10). The degree in (D11, D10) is the floating-point number.

Additional remark:

If the value in **S** exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1511	D	SQR	P	**S · D**	Square root of the binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source device

D : Device in which the result is stored

Explanation:

1. The square root of the value in the device specified by **S** is calculated, and the result is stored in the device specified by **D**.

2. The operation result stored in **D** is an integer. If the floating-point number is rounded down to the nearest whole digit, SM601 is ON.

3. If the operation result stored in **D** is 0, SM600 is ON.

4. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example:

When X0.0 is ON, the square root of the value in D0 is calculated, and the result is stored in D10.

Additional remark:

1. The value in the device specified by **S** only can be a positive value. If the value in the device specified by **S** is a negative value, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1512		FSQR	P	**S**、**D**	Square root of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source device

D : Device in which the result is stored

Explanation:

1. The square root of the floating-point number in the register specified by **S** is calculated, and the result is stored in the register specified by **D**.

2. If the operation result stored in **D** is 0, SM600 is ON.

Example 1:

When X0.0 is ON, the square root of the floating-point number in (D1, D0) is calculated, and the result is stored in (D11, D10).

Additional remark:

1. The value in the device specified by **S** only can be a positive value. If the value in the device specified by **S** is a negative value, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1513		FEXP	P	**S · D**	An exponent of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source device

D : Device in which the operation result is stored

Explanation:

1. Exponentiation involves two numbers, the base e which represents 2.71828, and the exponent in the device specified by **S**.

2. EXP[**D**+1, **D**]=[**S**+1, **S**].

3. The number in the device specified by **S** can be a positive number or a negative number. The register specified by **D** should be a 32-bit register, and the number in the device specified by **S** should be a floating-point number.

4. The value in the register specified by **D** is e^{S}. (e is 2.71828, and **S** represents the source data.)

5. If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, the value in the register specified by **D** is 16#7F800000, and SM602 is ON.

6. If the operation result stored in **D** is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the value in (D1, D0) is converted into the floating-point number, and the conversion result is stored in (D11, D10).

2. When X0.1 is ON, the exponentiation with the value in (D11, D10) as the exponent is performed. The result is a floating-point number, and is stored in (D21, D20).

NETWORK 1
X0.0
DFLT
En
D0
S
D
D10
NETWORK 2
X0.1
FEXP
En
D10
S
D
D20

API	Instruction code			Operand	Function
1514		FLOG	P	S_1 · S_2 · **D**	Logarithm of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○					○
S_2	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S_1 : Device in which the base is stored

S_2 : Source device

D : Device in which the operation result is stored

Explanation:

6

1. The logarithm of the value in S_2 with respect to the value in S_1 is calculated, and the single-precision floating-point operation result is stored in **D**.

2. The values in S_1 and S_2 only can be positive values.

3. S_1^D=S_2.→**D**=$Log_{S1}S_2$.

4. Example: Suppose the values in S_1 and S_2 are 5 and 125 respectively. Find $\log_5 125$.

5. S_1^D=S_2.→5**D**=125.→**D**=$\log_5 125$=3.

6. If the operation result stored in **D** is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the values in (D1, D0) and (D3, D2) are converted into the floating-point numbers, and the conversion results are stored in (D11, D10) and (D13, D12) respectively.

2. When X0.1 is ON, the logarithm of the floating-point number in (D13, D12) with respect to the floating-point number in (D11, D10) is calculated, and the operation result is stored in (D21, D20).

Additional remark:

1. If the value in $\mathbf{S_1}$ is less than or equal to 1, or if the value in $\mathbf{S_2}$ is less or equal to 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1515		FLN	P	**S · D**	Natural logarithm of the binary floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S : Source device

D : Device in which the operation result is stored

Explanation:

1. The natural logarithm of the operand **S** is calculated by the single-precision floating-point operation.

2. The value in **S** only can be a positive value.

3. e^{D}=**S**.→The value in **D**=ln**S**.

4. If the operation result stored in **D** is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the value in (D1, D0) is converted into the floating-point number, and the conversion result is stored in (D11, D10).

2. When X0.1 is ON, the natural logarithm of the floating-point number in (D11, D10) is calculated, and the operation result is stored in (D21, D20).

Additional remark:

1. If the value in S is less than or equal to 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1516		FPOW	P	S_1 · S_2 · D	A power of the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●							○
S_2	●	●			●	●	●	●	●							○
D		●			●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S_1 : Device in which the base is stored

S_2 : Device in which the power is stored

D : Device in which the operation result is stored

Explanation:

1. The single-precision floating-point number in **S_1** is raised to the power of the value in **S_2**, and the he single-precision floating-point operation result is stored in **D**.

2. **D**=POW[**S_1**+1 · **S_1**]^[**S_2**+1 · **S_2**]

3. The value in **S_1** only can be a positive value, whereas the value in **S_2** can be a positive value or a negative value.

4. Suppose the values in **S_1** and **S_2** are 5 and 3 respectively. $D=5^3=125$.

5. If the absolute value of the operation result is large than the value which can be represented by the maximum floating-point number, the value in **D** is 16#7F7FFFFF, and SM602 is ON.

6. If the absolute value of the operation result is less than the value which can be represented by the minimum floating-point number, the value in **D** is 16#FF800000, and SM601 is ON.

7. If the operation result stored in **D** is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the values in (D1, D0) and (D3, D2) are converted into the floating-point numbers, and the conversion results are stored in (D11, D10) and (D13, D12) respectively.

2. When X0.1 is ON, the floating-point number in (D11, D10) is raised to the power of the floating-point number in (D13, D12), and the operation result is stored in (D21, D20).

3. When X0.2 is ON, the binary floating-point number in (D21, D20) is converted into the binary-coded decimal floating-point number, and the conversion result is stored in (D31, D30).

Additional remark:

1. If the value in $\mathbf{S_1}$ is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1517		RAND	P	S_1 · S_2 · **D**	Random number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1	●	●			●	●		●	●		○	○	○	○		
S_2	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Minimum value

S_2 : Maximum value

D : Device in which the result is stored

Explanation:

1. The instruction is used to generate the random number within the range between the minimum value in S_1 and the maximum value in S_2, and the result is stored in **D**.

2. If the value in S_1 is larger than the value in S_2, the values in S_1 and S_2 are taken as the maximum value and the minimum value respectively when the instruction is executed.

Example:

When X0.0 is ON, the random number within the range between the minimum value in D0 and the maximum value in D10 is generated, and the result is stored in D20.

Additional remark:

The values in S_1 and S_2 should be within the range between 0 and 0~32767. If the value in S_1 or S_2 exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

6.17 Real-time Clock Instructions

6.17.1 List of Real-time Clock Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1600	TRD	–	✓	Reading the time
1601	TWR	–	✓	Writing the time
1602	T+	–	✓	Adding the time
1603	T-	–	✓	Subtracting the time
1604	HOUR	–	–	Running-time meter
1605	TCMP	–	✓	Comparing the time
1606	TZCP	–	✓	Time zone comparison
1607	DST	–	✓	Daylight saving time
1608	WWON	–	–	Weekly working time setup

6.17.2 Explanation of Real-time Clock Instructions

API	Instruction code			Operand	Function
1600		TRD	P	**D**	Reading the time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

D : Device in which the result is stored

Explanation:

1. **D**: The device in which the current time is stored
2. The operand **D** occupies seven consecutive devices.
3. The built-in real-time clock in the CPU module provides the data relating to the year, the week, the moth, the day, the minute, and the second. The data is stored in SR391~SR397. The instruction TRD is used to read the current time into the seven registers.
4. The last two digits of the year number for A.D. are stored in SR391.

Example:

When M0 is ON, the current time is read from the real-time clock into D0~D6. The value 1 in SR397 represents Monday, the value 2 represents Tuesday, and by analogy, the value 7 represents Sunday.

Special data register	Item	Value		General data register	Item
SR391	Year (A.D.)	00~99	→	D0	Year (A.D.)
SR392	Month	1~12	→	D1	Month
SR393	Day	1~31	→	D2	Day
SR394	Hour	0~23	→	D3	Hour
SR395	Minute	0~59	→	D4	Minute
SR396	Second	0~59	→	D5	Second
SR397	Week	1~7	→	D6	Week

Additional remark:

1. If **D**+6 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. When SM220 is ON, the real-time clock is calibrated within ±30 seconds. If the value of the second in the real-time clock is within the range between 0 and 29, the value of the second is cleared to zero. If the value of the second in the real-time clock is within the range between 30 and 59, the value of the minute increases by one, and the value of the second is cleared to zero.

API	Instruction code			Operand	Function
1601		TWR	P	**S**	Writing the time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S					●	●		●	●							

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Data source

Explanation:

1. **S**: The device into which the setting value is written

2. The operand **S** occupies seven consecutive devices.

3. When users want to adjust the built-in real-time clock in the CPU module, they can use the instruction to write the correct current time into the built-in real-time clock.

4. When the instruction is executed, the new setting time is instantly written into the real-time clock in the PLC. Therefore, when the instruction is executed, users have to make sure that the new setting time is consistent with the time when the new setting time is written into the real-time clock.

Example:

When M0 is ON, the correct current time is written into the built-in real-time clock in the PLC.

New setting time

General data register	Item	Value		Special data register	Item
D20	Year (A.D.)	00~99	→	SR391	Year (A.D.)
D21	Month	1~12	→	SR392	Month
D22	Day	1~31	→	SR393	Day
D23	Hour	0~23	→	SR394	Hour
D24	Minute	0~59	→	SR395	Minute
D25	Second	0~59	→	SR396	Second
D26	Week	1~7	→	SR397	Week

Real time clock

Additional remark:

1. If the value in **S** exceeds the range, the operation error occurs, the instruction is not executed, SM is ON, and the error code in SR is 16#2003.

2. If **S**+6 exceeds the device range, the operation error occurs, the instruction is not executed, SM is ON, and the error code in SR is 16#2003.

3. If users declare the operand **S** in ISPSoft, the data type will be ARRAY [7] of WORD/INT.

API	Instruction code			Operand	Function
1602		T+	P	S_1 , S_2 , D	Adding the time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1					●	●		●	●		○					
S_2					●	●		●	●		○					
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Source device 1

S_2 : Source device 2

D : Device in which the result is stored

6

Explanation:

1. The value of the hour, the value of the minute, and the value of the second in the real-time clock specified by **S_2** are added to the value of the hour, the value of the minute, and the value of the second in the real-time clock specified by **S_1**, and the sum is stored in the register specified by **D**.

2. The operands **S_1**, **S_2**, and **D** each occupy three consecutive devices.

3. If the sum is larger than or equal to 24 hours, SM602 is ON, and the result gotten from the subtraction of 24 hours from the sum is stored in **D**.

4. If the sum is 0 (0 hour 0 minute 0 second), SM600 is ON.

Example:

When M0 is ON, the instruction T+ is executed. The alue of the hour, the value of the minute, and the value of the second in D10~D12 are added to the value of the hour, the value of the minute, and the value of the second in D0~D2, and the sum is stored in D20~D22.

D0 8 (Hour)		D10 6 (Hour)		D20 14 (Hour)
D1 10 (Minute)	+	D11 40 (Minute)	→	D21 50 (Minute)
D2 20 (Second)		D12 6 (Second)		D22 26 (Second)
8 hour 10 minute 20 second		6 hour 40 minute 6 second		14 hour 50 minute 26 second

Additional remark:

1. If the value in $\mathbf{S}_1$ or $\mathbf{S}_2$ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If $\mathbf{S}_1$+2, $\mathbf{S}_2$+2, or **D**+2 exceeds the device range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If users declare the operand $\mathbf{S}_1$ in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

4. If users declare the operand $\mathbf{S}_2$ in ISPSoft, the data type will be ARRAY [3] of WORD/IN.

5. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

API	Instruction code			Operand	Function
1603		T-	P	S_1 , S_2 , **D**	Subtracting the time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●		○					
S_2					●	●		●	●		○					
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Source device 1

S_2 : Source device 2

D : Device in which the result is stored

Explanation:

6

1. The value of the hour, the value of the minute, and the value of the second in the real-time clock specified by S_2 are subtracted from the value of the hour, the value of the minute, and the value of the second in the real-time clock specified by S_1, and the difference is stored in the register specified by **D**.

2. The operands S_1, S_2, and **D** all occupy three consecutive devices.

3. If the difference is a negative, SM601 is ON, and the result gotten from the addition of 24 hours to the difference is stored in **D**.

4. If the difference is 0 (0 hour 0 minute 0 second), SM600 is ON.

Example:

1. When M0 is ON, the instruction T- is executed. The value of the hour, the value of the minute, and the value of the second in D10~D12 are subtracted from the value of the hour, the value of the minute, and the value of the second in D0~D2, and the difference is stored in D20~D22.

2. If the difference is a negative, SM601 is ON.

Additional remark:

1. If the value in $\mathbf{S}_1$ or $\mathbf{S}_2$ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If $\mathbf{S}_1$+2, $\mathbf{S}_2$+2, or **D**+2 exceeds the device range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If users declare the operand $\mathbf{S}_1$ in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

4. If users declare the operand $\mathbf{S}_2$ in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

5. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

API	Instruction code			Operand	Function
1604		HOUR		**S** · **D_1** · **D_2**	Running-time meter

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●		○	○	○	○		
D_1								●								
D_2		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D_1		●			●	●							
D_2	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	

Symbol:

S : Time after which the output device is ON

D_1 : Current time

D_2 : Output device

Explanation:

6

1. **S**: The time after which the output device is ON (Unit: Hour)

   D_1: The current time (Unit: Hour)

   D_2: The output device

2. **S**: The time after which the output device is ON (Unit: Hour)

   The operand **S** used in the 16-bit instruction should be within the range between 1 and 32,767.

3. The instruction HOUR:

   D_1: The current time (Unit: Hour)

   The value in **D_1** should be within the range between 0 and 32,767.

   D_1+1: The current time which is less than one hour (Unit: Second)

   The value in **D_1**+1 should be within the range between 0 and 3,599.

   D1+2 is for system use only. The value in it cannot be altered when the instruction is executed. Otherwise, an error will occur.

When the current time is 32,767 hour 3,599 second, the timer stops counting. After the values in $\mathbf{D}_1$ and $\mathbf{D}_1$+1 are cleared to 0, the timer starts to count again.

4. When the time for which the input contact has been ON reaches the setting time, the ouput device is ON. When the time for which the input contact has been ON does not reach the setting time, the ouput device is not ON. This function allows users to manage the running time of the machine and the maintenance.

5. After the output device is ON, the timer countinues to count.

6. When the on-line editing is used, please reset the conditional contact to initialize the instruction.

Example 1:

The 16-bit instruction HOUR: When X0.0 is ON, the timer starts to count. When the time for which X0.0 has been ON reaches 100 hours, Y0.0 is ON. The current time is recorded in D0, and the current time which is less than one hour is recorded in D1. D2 is for system use. The value in it can not be altered. Otherwise, an error will occur.

Additional remark:

1. When **S** is less than or equal to 0, the instruction is not executed, and the state of the output device is unchanged.

2. If the value in $\mathbf{D}_1$ used in the instruction HOUR is less than 0, the state of the output device is unchanged.

3. If $\mathbf{D}_1$+2 used in the instruction HOUR exceeds the device range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the operand $\mathbf{D}_1$ used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

5. If the operand $\mathbf{D}_1$ used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [2] of DWORD/DINT.

API	Instruction code			Operand	Function
1605		TCMP	P	S_1、S_2、S_3、**S**、**D**	Comparing the time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●		○	○	○	○		
S_2	●	●			●	●		●	●		○	○	○	○		
S_3	●	●			●	●		●	●		○	○	○	○		
S					●	●		●								
D		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Hour of the setting time

S_2 : Minute of the setting time

S_3 : Second of the setting time

S : Current time

D : Comparison result

Explanation:

1. The value of the hour, the value of the minute, and the value of the second specified by S_1~S_3 are compared with the value of the hour, the value of the minute, and the value of the second in the devices starting from the device specified by **S**, and the comparison result is stored in **D**.

2. The hour of the current time is in the device specified by **S**, and the value of the hour should be within the range between 0 and 23. The minute of the current time is in the device specified by **S**+1, and the value of the minute should be within the range between 0 and 59. The second of the current time is in the device specified by **S**+2, and the value of the second should be within the range between 0 and 59.

3. The operand **D** occupies three consecutive devices. The comparison result is stored in **D**, **D**+1, and **D**+2.

4. Users generally use the instruction TRD to read the current time from the real-time clock first, and then they use the

instruction TCMP to compare the time.

5. If the setting time in $\mathbf{S}_1$~$\mathbf{S}_3$ is larger than the current time in **S**, **D** is ON, **D**+1 is OFF, and **D**+2 is OFF.

6. If the setting time in $\mathbf{S}_1$~$\mathbf{S}_3$ is equal to the current time in **S**, **D** is OFF, **D**+1 is ON, and **D**+2 is OFF.

7. If the setting time in $\mathbf{S}_1$~$\mathbf{S}_3$ is less than the current time in **S**, **D** is OFF, **D**+1 is OFF, and **D**+2 is ON.

Example:

1. When X0.0 is ON, the instruction is executed. The setting time 12 hour 20 minute 45 second is compared with the current time in D20~D22, and the comparison result is stored in M10~M12. When X0.0 is switched from ON to OFF, the instruction is not executed. Besides, the state of M10, the state of M11, and the state of M12 remain the same as those before X0.0's being ON.

2. If users want to get the comparison result ≧, ≦, or ≠, they can connect M10~M12 is series or in parallel.

Additional remark:

1. If **S**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in **S** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the values in $\mathbf{S}_1$~$\mathbf{S}_3$ exceed the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

5. If users declare the operand **S** in ISPSoft, the data type will be ARRAY [3] of WORD.

6. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [3] of BOOL.

API	Instruction code			Operand	Function
1606		TZCP	P	**S_1 · S_2 · S · D**	Time zone comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●							
S_2					●	●		●	●							
S					●	●		●								
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Lower limit time

S_2 : Upper limit time

S : Current time

D : Comparison result

Explanation:

1. The instruction is used to compare the current time specified by **S** with the lower limit time specified by **S_1**, and compare the current time specified by **S** with the upper limit time specified by **S_2**, and the comparison result is stored in **D**.

2. The hour of the lower limit time is in the device specified by **S_1**, the minute of the lower limit time e is in the device specified by **S_1**+1, and the second of the lower limit time is in the device specified by **S_1**+2.

3. The hour of the upper limit time is in the device specified by **S_2**, the minute of the upper limit time e is in the device specified by **S_2**+1, and the second of the upper limit time is in the device specified by **S_2**+2.

4. The hour of the current time is in the device specified by **S**, the minute of the current time e is in the device specified by **S**+1, and the second of the current time is in the device specified by **S**+2.

5. The time in the device specified by **S_1** must be less than the time in the device specified by **S_2**. If the time in the device specified by **S_1** is larger than the time in the device specified by **S_2**, the time in the device specified by **S_1** will be taken as the upper/lower limit time during the execution of the instruction TZCP.

6. Users generally use the instruction TRD to read the current time from the real-time clock first, and then they use the instruction TZCP to compare the time.

7. If the current time in the device specified by **S** is less than the lower limit time in the device specified by $\mathbf{S}_1$, and is less than the upper limit time in the device specified by $\mathbf{S}_2$, **D** is ON. If the current time in the device specified by **S** is larger than the lower limit time in the device specified by $\mathbf{S}_1$, and is larger than the upper limit time in the device specified by $\mathbf{S}_2$, **D**+2 is ON. In other conditions, **D**+1 is ON.

Example:

When X0.0 is ON, the instruction TZCP is executed. M10, M11, or M12 is ON. When X0.0 is OFF, the instruction TZCP is not executed, the state of M10, the state of M11, and the state of M12 remain the same as those before X0.0's being ON.

Additional remark:

1. If $\mathbf{S}_1$+2, $\mathbf{S}_2$+2, **S**+2, or **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the values in $\mathbf{S}_1$, $\mathbf{S}_2$, and **S** exceed the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003

3. If users declare the operand $\mathbf{S}_1$ in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

4. If users declare the operand $\mathbf{S}_2$ in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

5. If users declare the operand **S** in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

6. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [3] of BOOL.

API	Instruction code			Operand	Function
1607		DST	P	**S** · **S_1** · **S_2** · **S_3** · **S_4** · **S_5** · **D**	Daylight saving time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●					○	○		
S_1					●	●		●					○	○		
S_2					●	●		●					○	○		
S_3					●	●		●					○	○		
S_4					●	●		●					○	○		
S_5					●	●		●					○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
S_5		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

6

Symbol:

S : Disable/enable/read the daylight saving time

S_1 : Month of the daylight saving start time

S_2 : Date of the daylight saving start time

S_3 : Month of the daylight saving end time

S_4 : Date of the daylight saving end time

S_5 : Duration of the daylight saving time (minutes)

D : The state of the daylight saving function

Explanation:

1. When the operand **S** is 0, the function of daylight saving time is disabled. When the operand **S** is 1, the function of daylight saving time is enabled. When the operand **S** is 2, the function of daylight saving time is being read. When the value shown in the **S** is not in the range of 0~2, it means the function of daylight saving time is being read.

2. After the instruction is executed, the state of the daylight saving time will be stored in **D**. When the value in **D** is OFF, the daylight saving time is disabled. When the value in **D** is ON, the daylight saving time is enabled.

3. **S_1**、**S_2**: Settings for the month and the date to start the daylight saving time

4. **S_3**、**S_4**: Settings for the month and the date to end the daylight saving time

5. **S_5**: Settings for the duration of the daylight saving time; unit: minute

6. When the instruction is executed to disable or enable the function of daylight saving time, the system time of the PLC will not be affected.

7. When the function of daylight saving time is enabled and the system runs for the first time during the start time (**S_1**, **S_2**), the system time will add the value set in **S_5** for one time. When the function of daylight saving time is disabled and the system runs for the first time during the end time (**S_1**, **S_2**), the system time will subtract the value set in **S_5** for one time.

8. When the **S** is set to enable the daylight saving time, the start/end month should be set between 1~12 and the start/end date should be set between 1~31. The start date should be set before the end date, for example, do not set the start date as Oct. 1st and the end date as April 1st. And the value should be set according to the actual calendar, for example if users set a date as April 31st but there is no such date, this function will not be enabled and the SM0 will be ON, the error code in SR0 is 16#200B.

9. When the **S** is set to enable the daylight saving time, the settings range should be within 1~1439 minutes, if not this function will not be enabled and the SM0 will be ON, the error code in SR0 is 16#200B.

10. When the **S** is set to disable the daylight saving time, the values in **S_1**~**S_5** are irrelevant.

11. When the **S** is set to read the state of the daylight saving function and the output state of **D** is ON, the PLC will save the setting values in operand **S_1**~**S_5**. The device is be set to **D** while the S is set to read. If the device is set to K or 16#, the values will not be saved. And the SM0 will be ON, the error code in SR0 is 16#2003.

Example 1:

To enable the daylight saving function and set the daylight starting on April 1st and ending on September 3rd and the duration is 60 minutes.

When the X0.0 is ON, the M0 will be ON too, meaning the daylight saving function is enabled.

That is the PLC system time will add 60 minutes when the date April 1st comes, and when the date September 3rd comes, the PLC system will subtract the 60 minutes to stop the daylight saving.

Example 2:

Use the instruction DST or the HWCONFIG in ISPSoft to read the daylight saving state.

When the X0.0 is ON, the settings concerning the daylight saving will be stored in the assigned registers; see the examples below:

Device	Setting Value	Description
D1	4	Starting month: April
D2	1	Starting date: 1st
D3	9	Ending month: September
D4	3	Ending date: 1st
D5	60	Duration: 60 minutes
M1	ON	The function of daylight saving time is ON.

API	Instruction code			Operand	Function
1608		WWON		S_1 · S_2 · D	Weekly working time setup

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								○								
S_2								○								
S_3								○								
S_4								○								
D		○	○	○												

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : The hour to start working (occupies 7 consecutive devices)

S_2 : The minute to start working (occupies 7 consecutive devices)

S_3 : The hour to stop working (occupies 7 consecutive devices)

S_4 : The minute to stop working (occupies 7 consecutive devices)

D : Output control

Explanation:

1. **S_1~ S_1+6** allows users to set the hour to start working on Sunday / Monday / Tuesday / Wednesday / Thursday / Friday / Saturday respectively. This operand occupies 7 consecutive devices, users can use the variables in ARRAY to declare the operands.

2. **S_2~ S_2+6 S_1~ S_1+6** allows users to set the minutes to start working on Sunday / Monday / Tuesday / Wednesday / Thursday / Friday / Saturday respectively. This operand occupies 7 consecutive devices, users can use the variables in ARRAY to declare the operands.

3. **S_3~ S_3+6** allows users to set the hour to stop working on Sunday / Monday / Tuesday / Wednesday / Thursday / Friday / Saturday respectively. This operand occupies 7 consecutive devices, users can use the variables in ARRAY to declare the operands.

4. **S_4~ S_4+6** allows users to set the minutes to stop working on Sunday / Monday / Tuesday / Wednesday / Thursday /

Friday / Saturday respectively. This operand occupies 7 consecutive devices, users can use the variables in ARRAY to declare the operands.

5. When the hour value in **S_1** is larger than the value set in **S_3**, it means the time to stop working is the next day. For example when users set the time to start working at 18:00 on Monday and the time to stop working at 6:00 , it means the time to stops working is at 6:00 Tuesday.

	Start working time						Stop working time				
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute
Sunday	**S_1**	24		**S_2**	00		**S_3**	24		**S_4**	00
Monday	**S_1**+1	18		**S_2**+1	00		**S_3**+1	06		**S_4**+1	00

6. The setting rage for the hour is 0~23. When the setting value is out of range, this function will not be enabled. The setting rage for the minute is 0~59. When the setting value is out of range, this function will be enabled but will use 0 as the setting value.

7. When it is required to set the work time to be more than 1 day, users can set the hour as 24 and that means the system will not check the start working time and the stop working time. For example, to set the start working time to 8 am Monday and the stop working time to 8pm Tuesday, settings will be like this **S_1**+1=8, **S_3**+1=24, **S_1**+2 =24 and **S_3**+2=20. See the formula below:

	Start working time						Stop working time				
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute
Sunday	**S_1**	24		**S_2**	00		**S_3**	24		**S_4**	00
Monday	**S_1**+1	08		**S_2**+1	00		**S_3**+1	24		**S_4**+1	00
Tuesday	**S_1**+2	24		**S_2**+2	00		**S_3**+2	20		**S_4**+2	00

8. This instruction should work with the real-time clock. Before operating, please make sure the battery is securely installed and working fine.

9. There is no limit on the number of times the instruction can be executed but the output control device **D** cannot be used repeatedly. If users use the device **D** repeatedly, only the last output result from the instruction WWON will be executed.

8. If there are more than 1 work hours are needed, please use the instruction WWON for several times as required. Please note the output control device **D** cannot be used repeatedly.

Example 1:

Set a working time from 8:00 to 18:00 from Monday to Friday and no work on Saturday and Sunday.

The settings for the device D:

	Start working time						Stop working time				
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute
Sunday	D0	24		D10	00		D20	24		D30	00
Monday	D1	08		D11	00		D21	18		D31	00
Tuesday	D2	08		D12	00		D22	18		D32	00
Wednesday	D3	08		D13	00		D23	18		D33	00
Thursday	D4	08		D14	00		D24	18		D34	00
Friday	D5	08		D15	00		D25	18		D35	00
Saturday	D6	24		D16	00		D26	24		D36	00

When M0 is ON, the Y0.0 is ON from 8:00 to 18:00 from Monday to Friday and for other times the Y0.0 is OFF.

Example 2:

Set a working time from 18:00 Monday to 08:00 Tuesday and from 18:00 Tuesday to 08: 00 Wednesday. Follow this pattern to 08:00 Saturday and no work on Sunday.

The settings for the device D:

	Start working time						Stop working time				
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute
Sunday	D0	24		D10	00		D20	24		D30	00
Monday	D1	18		D11	00		D21	08		D31	00
Tuesday	D2	18		D12	00		D22	08		D32	00
Wednesday	D3	18		D13	00		D23	08		D33	00
Thursday	D4	18		D14	00		D24	08		D34	00
Friday	D5	18		D15	00		D25	08		D35	00
Saturday	D6	24		D16	00		D26	24		D36	00

When M0 is ON, the Y0.0 is ON from 18:00 to 8:00 the next day from Monday to Friday and for other times the Y0.0 is OFF.

Example 3:

Set a working time from 08:00 to 12:00 and from 14:00 to 17:30 from Monday to Friday. No work on Saturday and Sunday.

The settings in the morning for the device D:

	Start working time						Stop working time				
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute
Sunday	D0	24		D10	00		D20	24		D30	00
Monday	D1	08		D11	00		D21	12		D31	00
Tuesday	D2	08		D12	00		D22	12		D32	00
Wednesday	D3	08		D13	00		D23	12		D33	00
Thursday	D4	08		D14	00		D24	12		D34	00
Friday	D5	08		D15	00		D25	12		D35	00
Saturday	D6	24		D16	00		D26	24		D36	00

The settings in the afternoon for the device D:

	Start working time						Stop working time				
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute
Sunday	D40	24		D50	00		D60	24		D70	00
Monday	D41	14		D51	00		D61	17		D71	30
Tuesday	D42	14		D52	00		D62	17		D72	30
Wednesday	D43	14		D53	00		D63	17		D73	30
Thursday	D44	14		D54	00		D64	17		D74	30
Friday	D45	14		D55	00		D65	17		D75	30
Saturday	D46	24		D56	00		D66	24		D76	00

When M0 is ON, the Y0.0 is ON from 08:00 to 12:00 and 14:00 to 17:30 from Monday to Friday and for other times the Y0.0 is OFF.

6.18 Peripheral Instructions

6.18.1 List of Peripheral Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1700	TKY	DTKY	–	Ten-key keypad
1701	HKY	DHKY	–	Sixteen-key keypad
1702	DSW	–	–	DIP switch
1703	ARWS	–	–	Arrow keys
1704	SEGL	–	–	Seven-segment display with latches

6.18.2 Explanation of Peripheral Instructions

API	Instruction code			Operand	Function
1700	D	TKY		**S** · **D_1** · **D_2**	Ten-key keypad

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	○															
D_1					●	●	●	●								
D_2		○	○	○				○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												
D_1		●	●		●	●	●						
D_2	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol:

S : Initial device

D_1 : Device in which the value is stored

D_2 : Output signal

6

Explanation:

1. The ten external inputs starting from the input specified by **S** represents 0~9 in the decimal system. They are connected to ten keys. Users can enter a four-digit decimal value 0~9,999 (16-bit instruction) or an eight-digit decimal value 0~99,999,999 (32-bit instruction) by pressing the keys in order. The decimal value is stored in **D_1**, and the output signals are stored in **D_2**.

2. The operand **S** occupies ten bits.

3. The operand **D_2** occupies eleven bits. Please do not change the states of the bits during the execution of the instruction.

4. When the conditional contact is not enabled, the eleven bits starting from the bit specified by **D_2** is OFF.

5. When the on-line editing is used, please reset the conditional contact to initialize the instruction.

6. Only when **D_1** uses 32-bit instructions, the 32-bit counter can be used.

Example:

1. The ten external inputs starting from X0.0 is connected to ten keys which represent 0~9 in the decimal system. When M0 is ON, the instruction is executed. The value that users enter is stored as a binary value in D0, and the output signals are stored in M10~M19.

Note: The digital input module AH16AM10N-5A is used in this example.

2. If the keys connected to X0.5, X0.3, X0.0, and X0.1 are pressed in the order shown in the timing chart, the result 5,301 is stored in D0. The maximum value which can be stored in D0 is 9,999. If the value exceeds four digits, the first digit from the left overflows.

3. After the key connected to the X0.2 is pressed and before other keys are pressed, M12 is ON. The same applies to other keys.

4. When a key connected to the input within the range between X0.0 and X0.9 is pressed, the corresponding output within the range between M10 and M19 is ON.

5. When one of the keys is pressed, M20 is ON.

6. When the conditional contact M0 is switched OFF, the value which was stored in D0 is unchanged. However, M10~M20 are switched OFF.

Additional remark:

1. If users declare the operand **S** in ISPSoft, the data type will be ARRAY [10] of BOOL.

2. If users declare the operand $\mathbf{D_2}$ in ISPSoft, the data type will be ARRAY [11] of BOOL.

API	Instruction code			Operand	Function
1701	D	HKY		S_1 · S_2 · D_1 · D_2 · D_3	Sixteen-key keypad

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2								●								
D_1		○														
D_2					●	●	●	●								
D_3		○	○	○				○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2		●			●	●							
D_1	●												
D_2		●	●		●	●	●						
D_3	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	AS

Symbol:

S_1 : Initial input device

S_2 : For system use only

D_1 : Initial output device

D_2 : Device in which the value is stored

D_3 : Output signal

Explanation:

1. The four external inputs starting from the input specified by **S** are connected to the four external outputs starting from the output specified by D_1 to form a 16-key keypad. The value that users enter by pressing the keys is stored in D_2, and the output signals are stored in D_3. If several keys are pressed simultaneously, the value which is smaller is stored.

2. The value that users enter by pressing the keys is temporarily stored in D_2. If the 16-bit instruction HKY is executed, the maximum value which can be stored in D_2 is 9,999. If the value exceeds four digits, the first digit from the left overflows. If the 32-bit instruction DHKY is executed, the maximum value which can be stored in D_2 is 9,999. If the value exceeds eight digits, the first digit from the left overflows.

3. After the execution of the instruction is complete, SM692 is ON. That is to say, SM692 is ON for a scan cycle after the execution of the matrix scan is complete.

4. Only when **D_2** uses 32-bit instructions, the 32-bit counter can be used.

Example:

1. The four external inputs X0.0~X0.3 are connected to the four external outputs Y0.0~Y0.3 to form a 16-key keypad. When X1.0 is ON, the instruction is executed. The value that users enter is stored as a binary value in D0, and the output signals are stored in M0~M7.

The function of SM691:

- If SM691 is ON, 0~F are taken as hexadecimal values in the execution of the instruction HKY.

  ■ Numeric keys:

- If SM691 is OFF, A~F are taken as function keys in the execution of the instruction HKY.

  ■ Numeric keys:

■ Function keys:

◆ When A is pressed, M0 keeps ON. When D is pressed, M0 is switched OFF, and M3 keeps ON.

◆ If several function keys are pressed, the key which is pressed first has priority.

2. Output signals:

- When a key within the range between A and F is pressed, M6 is ON.
- When a key within the range between 0 and 9 is pressed, M7 is ON.

3. When the conditional contact X1.0 is switched OFF, the value which was stored in D0 is unchanged. However, M0~M7 are switched OFF.

4. The external wiring:

Note: The transistor output module AH16AP11T-5A is used in this example.

Additional remark:

1. When this instruction is executed, a too long or a too short scan cycle time will cause the state of the switches not be read correctly. Use the following tips to solve the issues.

- When the scan cycle is too short, the I/O may not be able to respond in time and the correct states of the inputs cannot be read. Users can set a fixed scan time to solve this issue.
- When the scan cycle is too long, the switch may become slow to react. Users can write this instruction to the timer interrupt task to set a fixed to execute this instruction.

2. If users declare the operand S in ISPSoft, the data type will be ARRAY [4] of BOOL.

3. If users declare the operand D1 in ISPSoft, the data type will be ARRAY [4] of BOOL.

4. If users declare the operand D3 in ISPSoft, the data type will be ARRAY [8] of BOOL.

API	Instruction code			Operand	Function
1702		DSW		S_1、S_2、D_1、D_2、n	DIP switch

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2								●								
D_1		○														
D_2					●	●		●								
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2		●			●	●							
D_1	●												
D_2		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Initial input device

S_2 : For system use only

D_1 : Initial output device

D_2 : Device in which the value is stored

n : Number of DIP switches

Explanation:

1. The four or eight external inputs starting from the input specified by S_1 are connected to the four external outputs starting from the output specified by D_1 to form a four-digit DIP switch or two four-digit DIP switches. The value that users enter by pressing the DIP switch is stored in D_2. Whether there is one four-digit DIP switch or two four-digit DIP switches depends on **n**.

2. If **n** is 1, the operand D_2 occupies one register. If **n** is 2, the operand D_2 occupies two registers.

3. S_2 and S_2+1, which are for system use only, occupy two devices. Please do not alter the values in these devices.

4. After the execution of the instruction is complete, SM694 is ON for a scan cycle.

5. When the conditional contact is not enabled, the four external outputs starting from the output specified by D_1 keep OFF.

6. When the on-line editing is used, please reset the conditional contact to initialize the instruction.

Example:

1. X0.0~X0.3 are connected to Y0.0~Y0.3 to form the first DIP switch, and X0.4~X0.7 are connected to Y0.0~Y0.3 to form the second DIP switch. When X1.0 is ON, the instruction is executed. The value that users enter by pressing the first DIP switch is converted into the binary value, and the conversion result is stored in D20. The value that users enter by pressing the second DIP switch is converted into the binary value, and the conversion result is stored in D21.

2. When X1.0 is ON, Y0.0~Y0.3 are ON cyclically. After the execution of the instruction is complete, SM694 is ON for a scan cycle.

3. The outputs Y0.0~Y0.3 must be transistors.

4. The DIP switches:

Note: The transistor output module AH16AP11T-5A is used in this example.

Additional remark:

1. If **n** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If users declare the operand $\mathbf{D_1}$ in ISPSoft, the data type will be ARRAY [4] of BOOL.

API	Instruction code			Operand	Function
1703		ARWS		S_1、S_2、D_1、D_2、n	Arrow keys

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2								●								
D_1					●	●		●								
D_2		○														
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2		●			●	●							
D_1		●			●	●							
D_2	●												
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Initial input device

S_2 : For system use only

D_1 : Device in which the setting value is stored

D_2 : Initial output device

n : Positive/Negative logic

Explanation:

1. If the instruction is executed, S_1 is defined as the down arrow, S_1+1 is defined as the up arrow, S_1+2 is defined as the right arrow, and S_1+3 is defined as the left arrow. The setting value is stored in D_1, and it should be within the range between 0 and 9,999.

2. The operand S_1 occupies four consecutive bit devices.

3. S_2 is for system use only. Please do not alter the value in it.

4. The operand D_2 occupies eight consecutive bit devices.

5. When the conditional contact is disabled, the eight bit devices starting from the bit device specified by D_2 keep OFF.

6. The operand n should be within the range between 0 and 3. Please refer to the additional remark on the instruction API1704 SEGL for more information.

7. When the on-line editing is used, please reset the conditional contact to initialize the instruction.

Example:

1. If the instruction is executed, X0.0 is defined as the down arrow, X0.1 is defined as the up arrow, X0.2 is defined as the right arrow, and X0.3 is defined as the left arrow. The setting value is stored in D20, and it should be within the range between 0 and 9,999.

2. When X1.0 is ON, the digit in the place 10^3 is selected. If the left arrow is pressed, the places are selected in sequence ($10^3 \rightarrow 10^0 \rightarrow 10^1 \rightarrow 10^2 \rightarrow 10^3 \rightarrow 10^0$).

3. If the right arrow is pressed, the places are selected in sequence ($10^3 \rightarrow 10^2 \rightarrow 10^1 \rightarrow 10^0 \rightarrow 10^3 \rightarrow 10^2$). The LED indicators with the corresponding places are connected to Y0.4~Y0.7. When the digits in the places are selected in sequence, the LED indicators are ON in sequence.

4. If the up arrow is pressed, the digit in the place selected changes (0→1→2→…8→9→0→1). If the down arrow is pressed, the digit in the place selected changes (0→9→8→…1→0→9). The new digit is shown on seven-segment display.

Four-digit seven-segment display

The four keys are used to select the place and change the digit

Additional remark:

1. If **n** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If users declare the operand $\mathbf{S_1}$ in ISPSoft, the data type will be ARRAY [4] of BOOL.

3. If users declare the operand $\mathbf{D_2}$ in ISPSoft, the data type will be ARRAY [8] of BOOLL.

API	Instruction code			Operand	Function
1704		SEGL		**S_1 · S_2 · D · n**	Seven-segment display with latches

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●		○	○				
S_2								●								
D		○														
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D	●												
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Source device

S_2 : For system use only

D : Initial output device

n : Positive/Negative logic

Explanation:

1. The eight external outputs starting from the output specified by **D** are connected to a four-digit seven-segment display, or the twelve external outputs starting from the output specified by **D** are connected to two four-digit seven-segment displays. Every place is equipped with a driver which converts a binary-coded decimal value into seven-segment data, and every driver is equipped with a latch which can be used to store state information.

2. The value in **S_1** is the value which will be shown on first seven-segment display, and the value in **S_1**+1 is the value which will be shown on second seven-segment display.

3. **S_2** is for system use only. Please do not alter the value in it.

4. The operand **n** should be within the range between 0 and 7. Please refer to the additional remark for more information.

5. Whether there is one four-digit seven-segment display or two four-digit seven-segment displays, and whether an output is a positive logic output or a negative logic output depend on **n**.

6. If there is one four-digit seven-segment display, eight outputs are occupied. If there are two four-digit

seven-segment displays, twelve outputs are occupied.

7. When the instruction is executed, the outputs are ON cyclically. If the conditional contact is switched from OFF to ON during the execution of the instruction, the outputs are ON cyclically again.

8. After the execution of the instruction is complete, SM693 is ON for a scan cycle.

Example:

1. When X1.0 is ON, the instruction is executed. Y0.0~Y0.4 form a circuit. The value in D10 is converted into the binary-coded decimal value, and the conversion result is shown on first seven-segment display. The value in D11 is converted into the binary-coded decimal value, and the conversion result is shown on second seven-segment display. If the value in D10 or D11 exceeds 9,999, the operation error occurs.

2. When X1.0 is ON, Y0.4~Y0.7 are ON cyclically. It takes twelve scan cycles for Y0.4~Y0.7 to be ON. After the execution of the instruction is complete, SM693 is ON for a scan cycle.

3. If there is on four-digit seven-segment display, **n** is within the range between 0 and 3.

   - After the pins 1, 2, 4, and 8 are connected in parallel, they are connected to Y0.0~Y0.3 on the PLC, and the latches are connected to Y0.4~Y0.7 on the PLC.
   - When X1.0 is ON, the instruction is executed. Y0.4~Y0.7 are ON cyclically, and the value in D10 is shown on seven-segment display.

4. If there are two four-digit seven-segment displays, **n** is within the range between 4 and 7.

   - After the pins 1, 2, 4, and 8 are connected in parallel, they are connected to Y0.8~Y0.11 on the PLC, and the latches are connected to Y0.4~Y0.7 on the PLC.
   - The value in D10 is shown on first seven-segment display, and the value in D11 is shown on second seven-segment display. If the values in D10 and D11 are 1234 and 4321 respectively, 1234 is shown on second seven-segment display.

5. The wiring:

Note: The transistor output module AH16AN01T-5A is used in this example.

Additional remark:

- Whether an output is a positive output or a negative output, and whether there is one four-digit seven-segment display or two four-digit seven-segment displays depend on **n**.
- The outputs on the PLC should be NPN transistors whose collectors are open collectors. Besides, an output has to connect a pull-up resistor to the DC power supply (less than 30 V DC). Therefore, when an output is ON, a signal of low potential is output.
- The negative logic:

Binary-coded decimal value				**Output (Binary-coded decimal code)**				**Signal**			
b_3	**b_2**	**b_1**	**b_0**	**8**	**4**	**2**	**1**	**A**	**B**	**C**	**D**
0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	1	1	1	1	0
0	0	1	0	0	0	1	0	1	1	0	1
0	0	1	1	0	0	1	1	1	1	0	0
0	1	0	0	0	1	0	0	1	0	1	1
0	1	0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0	1	0	0	1
0	1	1	1	0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	1	0	1	1	0

- The positive logic:

Binary-coded decimal value				Output (Binary-coded decimal code)				Signal			
b_3	b_2	b_1	b_0	8	4	2	1	A	B	C	D
0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	0	0	1	0
0	0	1	1	1	1	0	0	0	0	1	1
0	1	0	0	1	0	1	1	0	1	0	0
0	1	0	1	1	0	1	0	0	1	0	1
0	1	1	0	1	0	0	1	0	1	1	0
0	1	1	1	1	0	0	0	0	1	1	1
1	0	0	0	0	1	1	1	1	0	0	0
1	0	0	1	0	1	1	0	1	0	0	1

- The latch:

Positive logic		Negative logic	
Latch	Signal	Latch	Signal
1	0	0	1

- The setting value of the parameter **n**:

Number of seven-segment displays	One				Two			
Output (Binary-coded decimal code)	+		-		+		-	
Latch	+	-	+	-	+	-	+	-
n	0	1	2	3	4	5	6	7

‘ + ’: Positive logic

‘ - ’: Negative logic

- Users can edit the parameters in **n** to modify the logics for the output transistor and the input of the seven-segment display.

- The connection of the common-anode four-digit seven-segment display with IC 7447 is as follows.

6.19 Communication Instructions

6.19.1 List of Communication Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1806	LRC	–	–	Longitudinal parity check
1807	CRC	–	–	Cyclic Redundancy Check
1808	MODRW	–	–	Reading/Writing the MODBUS data
1812	COMRS	–	–	Sending and receiving communication data
1813	COMDF	–	✓	Setting the communication format for a serial communication port
1814	VFDRW	–	–	Serial communication instruction exclusive for Delta AC motor drive
1815	ASDRW	–	–	Serial communication instruction exclusive for Delta servo drive
1816	CCONF	–	✓	Setting the parameters in the data exchange table of a communication port
1817	MODRWE	–	–	Reading and writing Modbus data without any flag used

6.19.2 Explanation of Communication Instructions

API	Instruction code			Operand	Function
1806		LRC		**S · n · D**	Longitudinal parity check

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S								●	●							
n								●	●				○	○		
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S : Initial device to which the LRC is applied

n : Number of bytes

D : Initial device in which the operation result is stored

Explanation:

1. Please refer to the additional remark on the instruction LRC for more information about the LRC check code.

2. The operand **n** should be an even number, and should be within the range between 1 and 1000. If **n** is not within the range, the operation error occurs, the instruction is not executed, SM0 and SM1 are ON, and the error code in SR0 is 16#200B.

3. The 16-bit conversion mode: When SM606 is OFF, the hexadecimal data in the device specified by **S** is divided into the high 8-bit data and the low 8-bit data. The LRC is applied to every byte, and the operation result is stored in the high 8-bit and the low 8-bit in the device specified by **D**. The number of bytes depends on **n**.

4. The 8-bit conversion mode: When SM606 is ON, the hexadecimal data in the device specified by **S** is divided into the high 8-bit data (invalid data) and the low 8-bit data. The LRC is applied to every byte, and the operation result is stored in the low 8-bit in the two registers. The number of bytes depends on **n**. (The values of the high 8 bits in the two registers are 0.)

Example:

1. The PLC is connected to the VFD-S series AC motor drive (ASCII mode: SM210 is OFF; 8-bit mode: SM606 is ON.). The PLC sends the command, and reads the data in the six devices at the addresses starting from 16#2101 in the VFD-S series AC motor drive.

   PLC⇨VFD-S

   The PLC sends “ : 01 03 2101 0006 D4 CR LF”.

The PLC sends the data.

Register	Data		Description	
D100 Low 8 bits	‘ : ’	16#3A	STX	
D101 Low 8 bits	‘0’	16#30	ADR 1	AD (1, 0) is the station address of the AC motor drive.
D102 Low 8 bits	‘1’	16#31	ADR 0	
D103 Low 8 bits	‘0’	16#30	CMD 1	CMD (10) is the command code.
D104 Low 8 bits	‘3’	16#33	CMD 0	
D105 Low 8 bits	‘2’	16#32	Initial data address	
D106 Low 8 bits	‘1’	16#31		
D107 Low 8 bits	‘0’	16#30		
D108 Low 8 bits	‘1’	16#31		
D109 Low 8 bits	‘0’	16#30	Number of data (counted by the word)	
D110 Low 8 bits	‘0’	16#30		
D111 Low 8 bits	‘0’	16#30		
D112 Low 8 bits	‘6’	16#36		
D113 Low 8 bits	‘D’	16#44	LRC CHK 0	LRC CHK (01) is the error checking code.
D114 Low 8 bits	‘4’	16#34	LRC CHK 1	
D115 Low 8 bits	CR	16#0D	END	
D116 Low 8 bits	LF	16#0A		

LRC CHK (01) above is the error checking code. It can be calculated by means of the instruction LRC. (8-bit mode: SM606 is ON.)

LRC check code: 16#01+16#03+16#21+16#01+16#00+16#06=16#2C

The two's complement of 16#2C is 16#D4. 'D' (16#44) is stored in the low 8-bit in D113, and '4' (16#34) is stored in the low 8-bit in D114.

Additional remark:

1. The format of the communication data in the ASCII mode:

STX	' : '	The start-of-text character is ' : ' (16#3A).
Address Hi	' 0 '	Communication address:
Address Lo	' 1 '	The 8-bit address is composed of two ASCII codes.
Function Hi	' 0 '	Function code:
Function Lo	' 3 '	The 8-bit function code is composed of two ASCII codes.
DATA (n-1)	' 2 '	Data: The n×8-bit data is composed of 2n ASCII codes.
	' 1 '	
	' 0 '	
…….	' 2 '	
	' 0 '	
DATA 0	' 0 '	
	' 0 '	
	' 2 '	
LRC CHK Hi	' D '	LRC check code:
LRC CHK Lo	' 7 '	The 8-bit check code is composed of two ASCII codes.
END Hi	CR	End-of-text character:
END Lo	LF	END Hi=CR (16#0D)· END Lo=LF (16#0A)

2. LRC check code: The values starting from the communication address to the data are added up. The two's complement of the sum gotten is the LRC check code.

   Example: 16#01+16#03+16#21+16#02+16#00+16#02=16#29

   The two's complement of 16#29 is 16#D7.

API	Instruction code			Operand	Function
1807		CRC		**S · n · D**	Cyclic Redundancy Check

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S								●	●							
n								●	●				○	○		
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S : Initial device to which the CRC is applied

n : Number of bytes

D : Initial device in which the operation result is stored

Explanation:

1. Please refer to the additional remark on the instruction CRC for more information about the CRC check code.

2. The operand **n** should be within the range between 1 and 1000. If **n** is not within the range, the operation error occurs, the instruction is not executed, SM0 and SM1 are ON, and the error code in SR0 is 16#200B.

3. The 16-bit conversion mode: When SM606 is OFF, the hexadecimal data in the device specified by **S** is divided into the high 8-bit data and the low 8-bit data. The CRC is applied to every byte, and the operation result is stored in the high 8-bit and the low 8-bit in the device specified by **D**. The number of bytes depends on **n**.

4. The 8-bit conversion mode: When SM606 is ON, the hexadecimal data in the device specified by **S** is divided into the high 8-bit data (invalid data) and the low 8-bit data. The CRC is applied to every byte, and the operation result is stored in the low 8-bit in the two registers. The number of bytes depends on **n**.

Example:

1. The PLC is connected to the VFD-S series AC motor drive (RTU mode: SM210 is ON; 16-bit mode: SM606 is ON.). The value 16#12, which will be written into the device at 16#2000 in the VFD-S series AC motor drive, is written into the device in the PLC first.

PLC⇨VFD-S

The PLC sends 01 06 2000 0012 02 07.

The PLC sends the data.

Register	Data	Description
D100 Low 8 bits	16#01	Address
D101 Low 8 bits	16#06	Function
D102 Low 8 bits	16#20	Data address
D103 Low 8 bits	16#00	
D104 Low 8 bits	16#00	Data
D105 Low 8 bits	16#12	
D106 Low 8 bits	16#02	CRC CHK 0
D107 Low 8 bits	16#07	CRC CHK 1

CRC CHK (01) above is the error checking code. It can be calculated by means of the instruction CRC. (8-bit mode: SM606 is ON.)

CRC check code: 16#02 is stored in the low 8-bit in D106, and 16#07 is stored in the low 8-bit in D107.

Additional remark:

1. The format of the communication data in the RTU mode:

START	Time interval
Address	Communication address: 8-bit binary address
Function	Function code: 8-bit binary code
DATA (n-1)	Data: n×8-bit data
…….	
DATA 0	
CRC CHK Low	CRC check code: The 16-bit check code is composed of two 8-bit binary codes.
CRC CHK High	
END	Time interval

2. CRC check code: The check code starts from the address to the data. The operation rule is as follows.

Step 1: Suppose the data in the 16-bit register (the register in which the CRC check code is stored) is 16#FFFF.

Step 2: The logical operator XOR takes the first 8-bit message and the low 8-bit data in the 16-bit register, and performs the logical exclusive OR operation on each pair of corresponding bits. The operation result is stored in the 16-bit register.

Step 3: The values of the bits in the 16-bit registers are shifted by one bit to the right. The value of the highest bit becomes 0.

Step 4: If the value of the right-most bit which is shifted to the right is 0, the data gotten from step 3 is stored in the 16-bit register. Otherwise, the logical operator XOR takes 16#A001 and the data in the 16-bit register, and performs the logical exclusive OR operation on each pair of corresponding bits. The operation result is stored in the 16-bit register.

Step 5: Repeat step 3 and step 4, and perform the operation on the 8-bit message.

Step 6: Repeat step 2~step 5, and get the next 8-bit message. Perform the operations on all messages. The final result in the 16-bit register is the CRC check code. Notice that the low 8-bit data in the 16-bit register is interchanged with the high 8-bit data in the 16-bit register before the CRC check code is put into the check code of the message.

API	Instruction code			Operand	Function
1808		MODRW		S_1 · S_2 · S_3 · S · n	Reading/Writing the MODBUS data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S								●								
n								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S	●	●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Unit address

S_2 : Function code

S_3 : Device address

S : Register involved in the reading/writing of the data

n : Data length

Explanation:

1. The operand S_1 should be within the range between 0 and 254; 0 is the broadcasting mode.

2. S_2: The function code

   For example:

Function code	Description	Data length	Devices that support devices for slaves
01	PLC reads the data from several bit devices.	1~1600	X, Y, M, SM, S, T, C, HC
02	PLC reads the data from several bit devices.	1~1600	X, Y, M, SM, S, T, C, HC
03	PLC reads the data from several word devices.	1~100	X, Y, SR, D, T, C, HC, E

Function code	Description	Data length	Devices that support devices for slaves
04	PLC reads the data from several word devices.	1~100	X
05	PLC writes the state into a bit device.	1	Y, M, SM, S, T, C, HC
06	PLC writes the data into a word device.	1	Y, SR, D, T, C, HC, E
0F	PLC writes the states into several bit devices.	1~1600	Y, M, SM, S, T, C, HC
10	PLC writes the data into several word devices.	1~100	Y, SR, D, T, C, HC, E

Only the function codes mentioned above are supported, and other function codes can not be executed. Please refer to the examples below.

3. **S_3**: The device address. If the address is illegal for the designated communication device, the communication device will respond with an error message. For example, the device address 16#8000 is illegal in DVP-ES2.

   S: The register involved in the reading/writing of the data
   The data which will be written into the external equipment is stored in the register in advance.

   The data which is read from the external equipment is stored in the register.

4. **n**: The length of the data

   For the word type communication function code, the data length cannot exceed 100 words.
   For the bit type (Bool) communictication function code, the data lengthe ranges from 1 to 1600bits.

5. The functions of **S_3**, **S**, and **n** vary with the function code used.

Function code	S_3	S	n
H01	Address from which the data is read.	Register in which the data read is stored.	Length of data read
H02	Address from which the data is read.	Register in which the data read is stored.	Length of data read
H03	Address from which the data is read.	Register in which the data read is stored.	Length of data read
H04	Address from which the data is read.	Register in which the data read is stored.	Length of data read
H05	Address into which the	Status value written	No meaning

Function code	S_3	S	n
	data is written.		
H06	Address into which the data is written.	Register in which the data written is stored	No meaning
H0F	Address into which the data is written.	Register in which the data written is stored	Length of data written
H10	Address into which the data is written.	Register in which the data written is stored	Length of data written

6. There is no limitation on the times of using this instruction, however only one instruction can be executed on the same COM port at a time.

7. If the communication timeout occurs, timeout flags are ON. After the problem is solved, users have to reset timeout flags to OFF. When using the instruction MODRW, the timeout value cannot be 0, and the value should be set between 100 ~ 32767ms; when the value is set to 0, it will be seen as 200ms.

8. In the MODBUS ASCII mode, users only need to set up the data (non-ASCII mode) for transmission, the instruction will convert the non-ASCII mode to the ASCII mode, consisting of the head code (:), the converted ASCII code, checksum (LRC) and tail code (CRLF). The data which is received is stored as the ASCII character in the internal register. The PLC automatically converts the data into the hexadecimal value, and if the communication data is correct, the conversion result will be stored in **S**. and the completion flag SM will be ON.

9. In the MODBUS RTU mode, users only need to set up the data for transmission, the instruction will add the checksum (CRC) and the data which is received is stored as the ASCII character in the internal register. The PLC automatically converts the data into the hexadecimal value, and if the communication data is correct, the conversion result will be stored in **S**.

Communication protocol setup example:

1. The following exapmles use PLC communication port 1 and special registers to demonstrate how to setup a communication protocol.

2. Users can set up the PLC communication port via the HWCONFIG of ISPSoft, or via the relative spcial registers, or API1813 COMDF to set up the communication. Please refer to ISPSoft manual for setups in HWCONFIG. As for communication register setups (SM, SR), please refer to section 6.19.3 for more details.

3. The communication setup for this example is RS485 ASCII, 9600, 8, E, 1 (SR209=16#0025).

4. Set the communication timeout to 3000ms (SR210=3000).

5. Set the communication mode to ASCII mode (SM210=OFF).

6. Enable the communication protocol (SM209=ON).

For users who set up the communication port via the instruction API1813 COMDF, this step can be ignored.

For users who set up the communication port in ISPSoft－ > HWCONFIG－ > COM Port, this step can be ignored.

Example 1:

1. Function code 01 (16#01): The PLC reads the data from several bit devices which are not discrete input devices. (16 peiece of data is read in this example.) For the function code 02, the operation is the same as the function code 01.

AS CPU module series is connected to the DVP-ES2 series PLC.

When SM96 and X0.0 are on, AS CPU module series sends and receives the Y0~Y17 commands from DVP-ES2.

When the address of Y0 is 16#0500, the states of Y0~Y17 in DVP-ES2 are listed below:

Device	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
State	ON	ON	OFF	ON	OFF	OFF	ON	OFF
Value	D				2			
Device	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
State	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
Value	0				4			

The operands of the instruction MODRW:

Operand	Description	Device
S_1	Unit address	16#0001
S_2	Function code	16#0001
S_3	Device address	16#0500

S	Register involved in the reading/writing of the data	D10.0
n	Data length	16

ASCII mode:

The ASCII codes do not need to be converted intentionally and they are all expressed in the 16# values.

- AS sends the communication command: “ : 01 01 05 00 00 10 E9 CR LF”.
- AS receives the communication command: “ : 01 01 02 D2 04 26 CR LF”.

RTU mode:

- AS sends the communication command: “01 01 05 00 00 10 3D 0A”.
- AS receives the communication command: “01 01 02 D2 04 E4 9F”.

If the format is correct, SM100 will be ON.

2. The responding messages of DVP-ES2 will be stored in registers D10.0 to 10.15. (The data read is D10.15~D10.0=16#04D2.)

Device	D10.7	D10.6	D10.5	D10.4	D10.3	D10.2	D10.1	D10.0
State	ON	ON	OFF	ON	OFF	OFF	ON	OFF
Value	D				2			
Device	D10.15	D10.14	D10.13	D10.12	D10.11	D10.10	D10.9	D10.8
State	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
Value	0				4			

3. After the receiving of the data sent back from DVP-ES2 is completed, the data format sent back from DVP-ES2 will be confirmed and determined whether it is correct. If no error occurs in the format, corresponding special flags SM100 will be on, if not SM102 will be ON.

Example 2:

1. Function code 03 (16#03): The PLC reads the data from several bit devices which are not discrete input devices. (8 peiece of data is read in this example.) For the function code 04, the operation is the same as the function code 03.

2. AS CPU module series is connected to the DVP-ES2 series PLC.

   When SM96 and X0.0 are on, AS CPU module series sends and receives the D32~D39 from DVP-ES2.

3. When the address of D32 is 16#1020, the values of D32~D39 in DVP-ES2 are listed below:

Device	D32	D33	D34	D35	D36	D37	D38	D39
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

The operands of the instruction MODRW:

Operand	Description	Device
S_1	Unit address	16#0001
S_2	Function code	16#0003
S_3	Device address	16#1020
S	Register involved in the reading/writing of the data	D10
n	Data length	8

ASCII mode:

The ASCII codes do not need to be converted intentionally and they are all expressed in the 16# values.

- AS sends the communication command: “：01 03 10 20 00 08 C4 CR LF”.
- AS receives the communication command: “：01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC AA CR LF”.

RTU mode:

- AS sends the communication command: “01 03 10 20 00 08 41 06”.
- AS receives the communication command: “01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC 90 FE”.

If the format is correct, SM100 will be ON.

- The responding messages of DVP-ES2 will be stored in registers D10 to D17.
- Values in D10~D17 :

Device	D10	D11	D12	D13	D14	D15	D16	D17
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

4. After the receiving of the data sent back from DVP-ES2 is completed, the data format sent back from DVP-ES2 will be confirmed and determined whether it is correct. If no error occurs in the format, corresponding special flags SM100 will be on, if not SM102 will be ON.

Example 3:

1. Function code 05 (16#05): PLC writes the state into a bit device. The device is set to ON in this example.

2. AS CPU module series is connected to the DVP-ES2 series PLC. D10.0 is ON and the Y0 of the DVP-ES2 series PLC is also ON. When SM96 and X0.0 are ON, the PLC can set the state of Y0.

The operands of the instruction MODRW:

Operand	Description	Device
S_1	Unit address	1
S_2	Function code	16#0005
S_3	Device address	16#0500
S	Register involved in the reading/writing of the data	D10.0
n	Data length (no meaning in here)	1

ASCII mode:

The ASCII codes do not need to be converted intentionally and they are all expressed in the 16# values.

- AS sends the communication command: “ : 01 05 05 00 FF 00 F6 CR LF”
- AS receives the communication command: “ : 01 05 05 00 FF 00 F6 CR LF”

RTU mode:

- AS sends the communication command: “01 05 05 00 FF 00 8C F6”
- AS receives the communication command: “01 05 05 00 FF 00 8C F6”

If the format is correct, SM100 will be ON.

3. After the receiving of the data sent back from DVP-ES2 is completed, the data format sent back from DVP-ES2 will be confirmed and determined whether it is correct. If no error occurs in the format, corresponding special flags SM100 will be on, if not SM102 will be ON.

4. When the DVP-ES2 receives this instruction, the Y0 will be ON.

5. The parameter n will not be used in here, since this function code here is for to write.

Example 4:

1. Function code 06 (16#06): PLC writes the state into a word device.

2. AS CPU module series is connected to the DVP-ES2 series PLC.

3. Suppose D10 is 16#55AA (waiting to write data to the device T0 of the DVP-ES2).

   When SM96 and X0.0 are ON, the PLC can write data to the T0 of the DVP-ES2 series PLC. The address of T0 is 16#0600.

   The operands of the instruction MODRW:

Operand	Description	Device
S_1	Unit address	1
S_2	Function code	16#0006
S_3	Device address of T0	16#0600
S	Register T0 involved in the reading/writing of the data	D10
n	Data length (no meaning in here)	0

ASCII mode:

The ASCII codes do not need to be converted intentionally and they are all expressed in the 16# values.

- AS sends the communication command: “ : 01 06 06 00 55 AA F4 CR LF”
- AS receives the communication command: “ : 01 06 06 00 55 AA F4 CR LF”

RTU mode:

- AS sends the communication command: “01 06 06 00 55 AA 36 6D”
- AS receives the communication command: “01 06 06 00 55 AA 36 6D”

If the format is correct, SM100 will be ON.

4. After the receiving of the data sent back from DVP-ES2 is completed, the data format sent back from DVP-ES2 will be confirmed and determined whether it is correct. If no error occurs in the format, corresponding special flags SM100 will be on, if not SM102 will be ON.
5. When the DVP-ES2 receives this instruction, it will write data stored in the device D10 to the device T0 of the DVP-ES2.
6. The parameter n will not be used in here, since this function code here is for to write.

Example 5:

1. Function code 0F (16#0F): PLC writes the states into several bit devices.

2. AS CPU module series is connected to the DVP-ES2 series PLC.

- Suppose D10.15~D10.0=16#04D2 (waiting to write the state of Y0~Y17 of the DVP-ES2)

Device	D10.7	D10.6	D10.5	D10.4	D10.3	D10.2	D10.1	D10.0
State	ON	ON	OFF	ON	OFF	OFF	ON	OFF
Value	D				2			
Device	D10.15	D10.14	D10.13	D10.12	D10.11	D10.10	D10.9	D10.8
State	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
Value	0				4			

When SM96 and X0.0 are ON, the PLC can set the state of Y0~Y17 of the DVP-ES2. The address of Y0 is 16#0500.

The operands of the instruction MODRW:

Operand	Description	Device
S_1	Unit address	1
S_2	Function code	16#000F
S_3	Device address of Y0	16#0500
S	Registers Y0~Y17 involved in the reading/writing of the data	D10.0
n	Data length	16

ASCII mode:

The ASCII codes do not need to be converted intentionally and they are all expressed in the 16# values.

- AS sends the communication command: “ : 01 0F 0500 0010 02 D2 04 03 CR LF”
- AS receives the communication command: “ : 01 0F A0 00 00 10 40 CR LF”

RTU mode:

- AS sends the communication command: “01 0F 05 00 00 10 02 D2 04 EA 43”
- AS receives the communication command: “01 0F A0 00 00 10 76 07”

If the format is correct, SM100 will be ON.

3. After the receiving of the data sent back from DVP-ES2 is completed, the data format sent back from DVP-ES2 will be confirmed and determined whether it is correct. If no error occurs in the format, corresponding special flags SM100 will be on, if not SM102 will be ON.

Device	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
State	ON	ON	OFF	ON	OFF	OFF	ON	OFF
Value	D				2			
Device	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
State	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
Value	0				4			

4. The parameter n will not be used in here, since this function code here is for to write.

Example 6:

1. Function code 10 (16#10): PLC writes the states into several word devices.

2. AS CPU module series is connected to the DVP-ES2 series PLC.

3. Suppose the values for D20~27 are listed below. (waiting to write data to the devices T0~7 of the DVP-ES2).

Device	D20	D21	D22	D23	D24	D25	D26	D27
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

When SM96 and X0.0 are ON, the PLC can write data to the T0~7 of the DVP-ES2 series PLC. The address of T0 is 16#0600.

The operands of the instruction MODRW:

Operand	Description	Device
S_1	Unit address	1
S_2	Function code	16#0010
S_3	Device address of T0	16#0600
S	Register T0~7 involved in the reading/writing of the data	D20
n	Data length (no meaning in here)	8

ASCII mode:

The ASCII codes do not need to be converted intentionally and they are all expressed in the 16# values.

- AS sends the communication command: " : 01 10 0600 00 08 10 1234 5678 1122 3344 5566 7788 99AA BBCC 8F CR LF"
- AS receives the communication command: " : 01 10 06 00 00 08 E1 CR LF"

RTU mode:

- AS sends the communication command: "01 10 06 00 00 08 10 1234 5678 1122 3344 5566 7788 99AA BBCC 0B 0C"
- AS receives the communication command: "01 10 06 00 00 08 C1 47"

If the format is correct, SM100 will be ON.

5. After the receiving of the data sent back from DVP-ES2 is completed, the data format sent back from DVP-ES2 will be confirmed and determined whether it is correct. If no error occurs in the format, corresponding special flags SM100 will be on, if not SM102 will be ON. When the DVP-ES2 receives this instruction, it will write data stored in the devices D20~27 to the device T0~7 of the DVP-ES2.

Device	T0	T1	T2	T3	T4	T5	T6	T7
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

6. The parameter n will not be used in here, since this function code here is for to write.

Additional remark:

1. If the value in $\mathbf{S_1}$ or $\mathbf{S_2}$ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the device specified by **S** is not sufficient to contain the **n** pieces of data, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If **n** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
4. If the function code specified by $\mathbf{S_2}$ is related to the bit device, the device specified by **S** has to be the bit device. Otherwise, the operation error occurs, the instruction is not executed, and the error code in SR0 is 16#2003.
5. If the function code specified by $\mathbf{S_2}$ is related to the word device, the device specified by **S** has to be the word device. Otherwise, the operation error occurs, the instruction is not executed, and the error code in SR0 is 16#2003.

6. If the communication command is 0x05 or 0x06, **n** does not work. The state or the data is written into one bit device or one word device.

7. If sending flags SM96 and SM97 are not ON, the instruction MODRW is not executed.

8. If the communication timeout occurs, the timeout flags SM104 and SM105 are ON, and the receving flags SM98 and SM99 are OFF.

9. If the error occurs during the reception of the data, the error flags SM102 and SM103 are ON, and the receving flags SM98 and SM99 are OFF.

10. If the function code specified by $\mathbf{S_2}$ is related to the word device, the device in the external equipment with which the PLC communicates has to be the word device. If the function code specified by $\mathbf{S_2}$ is related to the bit device, the device in the external equipment with which the PLC communicates has to be the bit device.

11. As for communication register setups (SM, SR), please refer to section 6.19.3 for more details.

API	Instruction code	Operand	Function

1812		COMRS		S_1 · S_2 · S_3 · D_1 · D_2	Self-defined sending and receiving communication data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●							
S_3								●	●				○	○		
D_1								●								
D_2								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
D_1		●			●	●							
D_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Communication port number (1~2, 11~12)

S_2 : Source of the data which is sent

S_3 : Length of the data which is sent

D_1 : Initial device in which communication data received is stored

D_2 Condition of ending the receiving of data

Explanation:

1. **S_1** is a communication port number, COM1 is number 1, COM 2 is number 2, Card 1 is number 11 and Card 2 is number 12. If the data is out of the communication port range, the instruction will not execute any sending or receiving.

2. If a specific character or specific characters are used as the condition of ending the receiving of data, it is suggested that the instruction should be applied to ASCII data. If the instruction is not applied to ASCII data, it is suggested that a timeout period should be used as the condition of ending the receiving of data.

3. **S_2**: Source of the data which is sent

   S_3: Length of the data which is sent

If **S_2** is D100 and **S_3** is 10, the values in the low bytes in D100~D109 will be sent through the communication port specified by **S_1**.

4. If the setting value in **S_3** is 0, no string will be sent. The maximum number of characters which can be sent is 256 words.

5. **D_1**: Length of the data which has been received.

   D_1+1~**D_1**+n: Devices in which the data received is stored

   If **D_1** is D200, the value in **D_2** is 3, and the value in **D_2**+1 is 16#0D0A, the data received will be stored in the low bytes in the devices starting from D201 (the high bytes will be unchanged), the receiving of data will not stop until the consecutive stop characters 16#0D and 16#0A are received, the length of the data received will be written to D200 after 16#0D and 16#0A are received, and a completion flag will be set to ON after the receiving of data stops.

6. **D_2**: Mode of receiving data

   D_2+1: Condition of ending the receiving of data

   D_2 and **D_2**+1 are described below.

D_2	**Mode of receiving data**	**Setting value in D_2+1**	**Remark**
0	Not receiving communication data	Unused	After the sending of data is complete, a completion flage will be set to ON.
1	When the time which passes before the next piece of data is received exceeds the time set in **D_2**+1, the receiving of data is complete.	The setting value in **D_2**+1 is time. The unit of measurement for time is 1 millisecond. The setting value in **D_2**+1 is in the range of 5 to 3000.	If the time that users set is greater than 3000 milliseconds, the value in **D_2**+1 will be 3000. If the time that users set is less than 5 milliseconds, the value in **D_2**+1 will be 5.
2	The data received ends with a specific character.	The setting value in **D_2**+1 is a specific character.	If a specific character is 16#0A, the value in **D_2**+1 will be 16#000A.
3	The data received ends with two consecutive specific characters.	The setting value in **D_2**+1 is two specific characters.	If two specific characters are 16#0D and 16#0A, the value in **D_2**+1 will be 16#0D0A.
4	The data received starts with a specific character. When the time which passes before the next	A specific character is stored in the high byte in **D_2**+1, and time is	If a start character is 16#3A, and time is 15 milliseconds, the value in **D_2**+1 will be 16#3A0F.

	piece of data is received exceeds the time set in **D_2**+1, the receiving of data is complete.	stored in the low byte in **D_2**+1. (The time set in the low byte in **D_2**+1 is in the range of 5 milliseconds to 255 millseconds.)	
5	The data received starts with a specific character, and ends with a specific character.	The setting value in **D_2**+1 is a specific start character, and a specific end character.	If a start character is 16#3A, and a stop character is 16#0A, the value in **D_2**+1 will be 16#3A0A.
6	A specific quantity of data is received.	The setting value in **D_2**+1 is the length of the data which is received. Setting range is 1~256.	If users want to receive 10 characters, the value in **D_2**+1 is 10.
7	The data received ends with a specific character and generate communication interruptions.	The setting value in **D_2**+1 is a specific end character.	If an end character is 16#0A, the value in **D_2**+1 will be 16#000A.
8	Set the quantity of data received and then generate communication interruptions.	The setting value in **D_2**+1 is the length of the data which is received. Setting range is 1~256.	If users want to receive 10 characters, the value in **D_2**+1 is 10.
9	The data received ends with a specific character or a specific quantity of data received; when either condition is met, the transmission is complete.	A specific end character is stored in the high byte in **D_2**+1, and time is stored in the low byte in **D_2**+1. (The time set in the low byte in **D_2**+1 is in the range of 1 milliseconds to 255 millseconds.)	If an end character is 16#0A, and time is 15 milliseconds, the data length is 15 words, the value in **D_2**+1 will be 16#0A0F.
Others	If the mode used is not a mode which is supported, the instruction will not be executed.		

7. Except mode 6 and 8, when data received in D_2 exceeds the maximum range of the received data length 256 words and no ending character is received, the instruction will stop executing and will regard this opearation as a receiving error. D_1+0 is 0 and D_1+1~ will not be inputted in the received data.

8. The relation among communication port, related spcial auxiliary relays, and a related special data register are described in section 6.19.3.

9. Timing diagrams

- Mode of receiving data: 0

  When data is sent, users can not cancel the sending of the data. If the conditional contact preceding the instruction is not enable, the data will still be sent, but a completion flag will not be set to ON after the sending of the data is complete.

- Mode of receiving data: 1 or 4

Description:

①→ Users start/stop the execution of the instruction.

②→ Time in which data is sent

The period of time in which data is sent is not measured.

③→ After the first character is received, the time which passes before the next character is received will be measured. Whenever a character is received, the time measured is cleared. The completion flag will not be set to ON until the time measured is greater than the setting value in D_2+1.

④→ If the instruction is still enabled after users reset the completion flag or the communication flag, the next communication data is sent automatically when the instruction is scanned in the next cycle.

⑤→ When the PLC begins to receive data, it begins to measure the time which passes. The communication timeout will not be set to ON until the time measured exceeds the timeout period set. It is suggested that the timeout period set should be longer than the time set in **D_2**+1.

- Mode of receiving data: 2, 3, 5, 6, or 9.

Description:

① → Users start/stop the execution of the instruction.

② → Time in which data is sent

The period of time in which data is sent is not measured.

③ → After the first character is received, the time which passes before the next character is received will be measured. Whenever a character is received, the time measured is cleared. A communication timeout flag will not be set to ON until the time measured exceeds the timeout period set.

④ → If the instruction is still enabled after users reset a completion flag or a communication flag, the next communication data is sent automatically when the instruction is scanned in the next cycle.

10. Mode of sending data/Mode of receiving data

8-bit mode: The command which is edited is stored in the initial transmission device, and the command which will be sent include the head code and the tail code. The 16-bit data is divided into the high 8-bit data and the low 8-bit data. The high 8-bit data is ignored, and the low 8-bit data can be sent or received. (Take standard Modbus for example.)

Sending the data: (PLC→External equipment)

Receiving the data: (External equipment→PLC)

16-bit mode: The command which is edited is stored in the initial transmission device, and the command which will be sent include the head code and the tail code. When SM106/SM107 is OFF, the 16-bit data is divided into the high 8-bit data and the low 8-bit data.

Sending the data: (PLC→External equipment)

Receiving the data: (External equipment→PLC)

The data which the PLC receives from the external equipment includes the head and the tail code. Therefore, users have to be aware of the setting of a length.

11. When the mode is 7 or 8, the corresponding communication port and the interruption number are listed below.

Communicaiton port number	COM1	COM2	Card1	Card2
Interruption number	I300	I302	I304	I306

The followings use COM1 (RS485) as examples.

Example 1:

The mode in **D_2** is 0 (not receiving communication data) and set the mode of sending data/Mode of receiving data to 8-bit mode (SM106=ON).

1. The length for the data to be sent: D20=4.
2. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.
3. Set D10=16#0000 (sending data only, not receiving data).
4. Enable the contact X0.0.
5. PLC sends out 4 pieces of data.
6. Sending data: PLC→External equipment 31 32 33 34.
7. Since receiving data is not required, after the PLC sends out all the data, the operation ends. SM100=0.
8. For another data sending, users can set the flag SM100 to OFF to start the operation again.

Example 2:

The mode in **D_2** is 1 (setting the time value to 5~3000ms) and set the mode of sending data/Mode of receiving data to 16-bit mode (SM106=OFF).

1. The length for the data to be sent: D20=4.
2. The contents for the data to be sent: D100=16#3231, D101=16#3433.
3. Set D10=16#0001 (mode: 1), D11=300 (set the time value to 300ms).
4. Enable the contact X0.0.
5. PLC sends out 4 pieces of data.
6. Sending data: PLC→External equipment 31 32 33 34.
7. After the external equipment received the data from the PLC, 5 consecutive data will be sent to the PLC and each sent is with less than 20ms. External equipment 35 36 37 38 39.
8. D200=5 (number of the data received), the content of the received data: D201=16#3635, D202=16#3837, D203=16#0039.
9. SM100=ON: the reception of data is complete.
10. For another data sending, users can set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 will be ON and then the PLC will start receiving data. The interval time of every data reception is set in D11. When the interval time exceeds the set time and no data is coming in, the SM100 will be ON.

Example 3:

The mode in **D_2** is 2 (The data received ends with a specific character.) and set the mode of sending data/Mode of receiving data to 8-bit mode (SM106=ON).

1. The length for the data to be sent: D20=0, meaning the PLC will not send data but only receives data.

2. Set D10=16#0002 (mode: 2), D11=16#000A (the ending character is 16#0A).

3. Enable the contact X0.0.

4. PLC is waiting to receive data from the external equipment. (D20=0, meaning the PLC will not send data to the external equipment.).

5. The external equipment sends data to the PLC.

   External equipment → PLC 31 32 33 34 35 0A.

6. D200=6 (number of the data received), the content of the received data: D201=16#0031, D202=16#0032, D203=16#0033, D201=16#0034, D202=16#0035, D203=16#000A

7. SM100=ON: the reception of data is complete.

8. For another data sending, users can set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 will be ON and then the PLC will start receiving data till the ending character (16#0A) is received. When the reception of data is complete, SM100 is ON. If the communication timeout occurs but the ending character (16#0A) is still not received, the communication timeout flag SM104 will be ON.

Example 4:

The mode in **D_2** is 3 (The data received ends with two specific character.) and set the mode of sending data/Mode of receiving data to 16-bit mode (SM106=OFF).

The example uses DVP-ES2 as the external equipment and is going to write H1234 to the D100 of the DVP-ES2.

1. The length for the data to be sent: D20=17.

2. The contents for the data to be sent: D100=16#303A, D101=16#3031, D102=16#3136, D103=16#3630, D104=16#3134, D105=16#3332, D106=16#3334, D107=16#0D46, D108=16#000A.

3. Set D10=16#0003 (mode: 3), D11=16#0D0A (the ending character is 16#0D and 16#0A).

4. Enable the contact X0.0

5. PLC sends out 17 pieces of data.

   Sending data: PLC→External equipment 3A 30 31 30 36 31 30 36 34 31 32 33 34 33 46 0D 0A

   (ASCII code: 0106106412343FCRLF)

6. The external equipment receives data from the PLC and the last 2 data are 16#0D and 16#0A.

   External equipment → PLC 3A 30 31 30 36 31 30 36 34 31 32 33 34 33 46 0D 0A

   (ASCII code: 0106106412343FCRLF)

7. D200=17 (number of the data received), the content of the received data: D201=16#303A, D202=16#3031, D203=16#3136, D204=16#3630, D205=16#3134, D206=16#3332, D207=16#3334, D208=16#0D46, D209=16#000A.

8. SM100=ON: the reception of data is complete.

9. For another data sending, users can set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 will be ON and then the PLC will start receiving data till the ending character (16#0D0A) is received. When the reception of data is complete, SM100 is ON. If the communication timeout occurs but the ending character (16#0D0A) is still not received, the communication timeout flag SM104 will be ON.

Example 5:

The mode in **D_2** is 4 (The data received starts with a specific character and set the time value to 5~255ms.) and set the mode of sending data/Mode of receiving data to 8-bit mode (SM106=ON).

1. The length for the data to be sent: D20=4.

2. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.

3. Set D10=16#0004 (mode: 4), D11=16#3A0F (the starting character is 16#3A and set the time value to 16#0F, meaning 15ms).

4. Enable the contact X0.0.

5. PLC sends out 4 pieces of data.

   Sending data: PLC→External equipment 31 32 33 34

6. The external equipment receives data from the PLC and then sends 7 consecutive words to the PLC with an interval of 1ms between each sending.

   External equipment → PLC 30 3A 35 36 37 38 39

7. D200=6 (number of the data received), the content of the received data: D201=16#003A, D202=16#0035, D203=16#0036, D204=16#0037, D205=16#0038, D206=16#0039.

8. SM100=ON: the reception of data is complete.

9. For another data sending, users can set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 will be ON and then the PLC is ready to receive data. When receiving the starting character 16#3A, the PLC will start receiving data. The interval time of every data reception is set in D11. When the interval time exceeds the set time 16#0F (15ms) and no data is coming in, the SM100 will be ON.

Example 6:

The mode in $\mathbf{D_2}$ is 5 (The data received starts and ends with a specific character) and set the mode of sending data/Mode of receiving data to 16-bit mode (SM106=OFF).

The example uses DVP-ES2 as the external equipment and is going to read data in the D100 of the DVP-ES2.

1. The length for the data to be sent: D20=17.

2. The contents for the data to be sent: D100=16#303A, D101=16#3031, D102=16#3133, D103=16#3630, D104=16#3034, D105=16#3030, D106=16#3831, D107=16#0D37, D108=16#000A

3. Set D10=16#0005 (mode: 5), D11=16#3A0A (the starting character is 16#3A and the ending character is 16#0A).

4. Enable the contact X0.0.

5. PLC sends out 17 pieces of data.

   Sending data: PLC→External equipment 3A 30 31 30 36 31 30 36 34 31 32 33 34 33 46 0D 0A

   (ASCII code: 0106106412343FCRLF)

6. The external equipment receives data from the PLC and the last 2 data are 16#0D and 16#0A.

   External equipment → PLC 3A 30 31 30 36 31 30 36 34 31 32 33 34 33 46 0D 0A

   (ASCII code: 0106106412343FCRLF)

7. D200=15 (number of the data received), the content of the received data: D201=16#303A, D202=16#3031, D203=16#3033, D204=16#3132, D205=16#3332, D206=16#4234, D207=16#0D34, D208=16#000A.

8. SM100=ON: the reception of data is complete.

9. For another data sending, users can set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 will be ON and then the PLC is ready to receive data. When receiving the starting character 16#3A, the PLC will start receiving data till the ending character 16#0A is received. And the the SM100 will be ON. If the communication timeout occurs but the starting character 16#3A or the ending character 16#0A is still not received, the communication timeout flag SM104 will be ON.

Example 7:

The mode in $\mathbf{D_2}$ is 6 (the received data length) and set the mode of sending data/Mode of receiving data to 8-bit mode (SM106=ON).

1. The length for the data to be sent: D20=4.

2. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.

3. Set D10=16#0006 (mode: 6), D11=16#0008 (8 pieces of data to be received).

4. Enable the contact X0.0.

5. PLC sends out 4 pieces of data.

   Sending data: PLC→External equipment 31 32 33 34

6. The external equipment receives data from the PLC and then sends 8 consecutive data to the PLC.

   External equipment → PLC 32 33 34 35 36 37 38 39

7. D200=8 (number of the data received), the content of the received data: D201=16#0032, D202=16#0033, D203=16#0034, D204=16#0035, D205=16#0036, D206=16#0037, D207=16#0038, D208=16#0039.

8. SM100=ON: the reception of data is complete.

9. For another data sending, users can set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 will be ON and then the PLC is ready to receive data. When receiving the set quantity of data, the SM100 will be ON. If the communication timeout occurs but the set quantity of data is still not received, the communication timeout flag SM104 will be ON.

Example 8:

The mode in $\mathbf{D_2}$ is 7 (The data received ends with a specific character and generates communication interruptions.) and set the mode of sending data/Mode of receiving data to 8-bit mode (SM106=ON).

Communication interruption programs:

1. Clear the interruption: D30=0

2. The length for the data to be sent: D20=4.

3. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.

4. Set D10=16#0007 (mode: 7), D11=16#000A (16#0A is the ending character).

5. Enable the contact X0.0.

6. PLC sends out 4 pieces of data.

   Sending data: PLC→External equipment 31 32 33 34

7. D30=0 (the programs in the interruption are not executed.)

8. The external equipment sends data to the PLC.

   External equipment → PLC 31 32 33 34 35 0A

9. D200=6 (number of the data received), the content of the received data: D201=16#0031, D202=16#0032, D203=16#0033, D201=16#0034, D202=16#0035, D203=16#000A.

10. SM100=ON: the reception of data is complete.

11. D30=1 (the interruption is trigger and then the INC D30 is executed.)

12. For another data sending, users can set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 will be ON and then the PLC is ready to receive data. When receiving the set ending character (16#06), the SM100 will be ON. If the communication timeout occurs but the set ending character is still not received, the communication timeout flag SM104 will be ON.

Example 9:

The mode in **D_2** is 8 (The set quantity of data is received and generates communication interruptions.) and set the mode of sending data/Mode of receiving data to 8-bit mode (SM106=ON).

Communication interruption programs:

1. Clear the interruption: D30=0

2. The length for the data to be sent: D20=4.

3. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.

4. Set D10=16#0008 (mode: 8), D11=16#0008 (8 pieces of data to be received).

5. Enable the contact X0.0.

6. PLC sends out 4 pieces of data.

   Sending data: PLC→External equipment 31 32 33 34

7. D30=0 (the programs in the interruption are not executed.)

8. The external equipment receives data from the PLC and then sends 8 consecutive data to the PLC.

9. External equipment → PLC 32 33 34 35 36 37 38 39

10. D200=8 (number of the data received), the content of the received data: D201=16#0032, D202=16#0033, D203=16#0034, D204=16#0035, D205=16#0036, D206=16#0037, D207=16#0038, D208=16#0039.

11. SM100=ON: the reception of data is complete.

12. D30=1 (the interruption is trigger and then the INC D30 is executed.)

13. For another data sending, users can set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 will be ON and then the PLC is ready to receive data. When receiving the set quantyity of data, the SM100 will be ON. If the communication timeout occurs but the set quantity of data is still not received, the communication timeout flag SM104 will be ON.

Example 10:

The mode in **D_2** is 9 (The set ending character or the set quantity of data is received) and set the mode of sending data/Mode of receiving data to 8-bit mode (SM106=ON).

1. The length for the data to be sent: D20=4.

2. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.

3. Set D10=16#0009 (mode: 9), D11=16#0A0F (the ending character is 16#0A and the set data length is 16#0F).

4. Enable the contact X0.0.

5. PLC sends out 4 pieces of data.

   Sending data: PLC→External equipment 31 32 33 34

6. The external equipment receives data from the PLC and then sends 15 pieces of data to the PLC.

   External equipment → PLC 31 32 33 34 35 0A 41 42 43 44 45 46 47 48 49

7. D200=6 (number of the data received), the content of the received data: D201=16#0031, D202=16#0032, D203=16#0033, D204=16#0034, D205=16#0035, D206=16#000A.

   The PLC stops receiving data after the 6th of data is received.

8. SM100=ON: the reception of data is complete.

9. For another data sending, users can set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 will be ON and then the PLC is ready to receive data. When receiving the set ending character or the set quantity of data, the SM100 will be ON. If the communication timeout occurs but the set ending character or the set quantity of data is still not received, the communication timeout flag SM104 will be ON.

Additional remark:

1. There is no limit on the number of times the communication instruction COMRS can be executed. However, every communication port can only be enabled by one communication instruction, and the communication instructions which follow will not be executed.
2. When COMRS is executed, no checksum is used. If users need a checksum, they can use COMRS and another instruction available.

3. If the value in $\mathbf{D_2}$ is 2, 3, 5, 6 or 9, it is suggested that users should set a timeout period. After a timeout period is set, the sending of data will be retried if a stop character is not received.
4. The instruction does not automatically clear the value in $\mathbf{D_1}$~$\mathbf{D_1}$+n whenever the instruction is just executed or the PLC begins to receive new communication data. Only after a completion flag is switched from OFF to ON can users know whether data is received and how much data the PLC receives. If the users want to clear the values in $\mathbf{D_1}$~$\mathbf{D_1}$+n, they can use the instruction ZRST.
5. If the value in $\mathbf{S_1}$ is out of range, the instruction will not be executed.
6. If the number of devices starting from $\mathbf{S_2}$ is not equal to the value in $\mathbf{S_3}$, the instruction will not be executed, SM0 will be ON, and the error code in SR0 will be 16#2003.
7. If the value in $\mathbf{D_2}$ is not in the range of 0 to 9, the instruction will not be executed, SM0 will be ON, and the error code in SR0 will be 16#200B.
8. If the value in $\mathbf{D_2}$ is 6, 8 or 9, and the number of devices starting from $\mathbf{D_1}$ is not equal to the value in $\mathbf{D_2}$+1, the instruction will not be executed, SM0 will be ON, and the error code in SR0 will be 16#2003.
9. The quantity of data received is greater than the number of devices starting from $\mathbf{D_1}$, the data which can not be stored will be ignored.
10. If a completion flag is ON, the PLC will stop receiving data. If a communication port receives data when a completion flag is ON, the data will not be received.
11. If the setting value in $\mathbf{S_3}$ is less than 0 or greater than 256, the instruction will not be executed, SM0 will be ON, and the error code in SR0 will be 16#200B.
12. When the mode of $\mathbf{D_2}$ is 6 or 8, the length of $\mathbf{D_2}$+1 is less than 1 or greater than 256, the instruction will not be executed. SM0 will be ON, and the error code in SR0 will be 16#200B.

API	Instruction code			Operand	Function
1813		COMDF	P	S_1 · S_2 · S_3 · S_4 · S_5 · D	Setting the communication format for a serial communication port

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1													○	○		
S_2													○	○		
S_3													○	○		
S_4													○	○		
S_5													○	○		
D													○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													
S_2													
S_3													
S_4													
S_5													
D													

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

6

Symbol:

S_1 : Baud Rate (Unit:100 bps)

S_2 : Number of data bits

S_3 : Parity bit

S_4 : Number of end bits

S_5 : MODBUS format selection

D : Communication port number

Explanation:

1. The instruction only provides the way of directly setting the parameter values, instead of variable declaration.

2. S_1 sets the baud rate with the unit 100bps. For example, the input value 96 indicates 9600bps.

3. S_2 sets the number of data bits. The input value 7 indicates 7 data bits. 8 means 8 data bits. If the value is not 7 or 8 in S_2, the instruction is executed with the default value.

4. **S_3** sets the parity bit. The value 0 indicates None (no parity bit). 1 corresponds to Odd bit checking. The setting value 2 corresponds to Even bit checking. If the value is not 0, 1 or 2 in S_3, the default value is filled automatically.

5. **S_4** sets the number of end bits. The value 1 (preset) indicates 1 bit. 2 means 2 bits. If the value is not 1 or 2 in S_4, the instruction is executed with the default value.

6. **S_5** sets the communication mode of the Modbus communication protocol. The value 0 indicates ASCII (default value) and 1 means RTU. If the value is not 0 or 1 in S_5, the instruction is executed with the default value.

7. **D** sets communication port number. The serial number of COM1 is 1, COM2 is 2, Card1 is 11 and Card2 is 12. If the setting value is out of the valid range, the instruction will not perform any communication port format setting.

8. Users can also directly set the communication port through ISPSoft－＞HWCONFIG－＞COM Port or the special registers. (For the setting in HWCONFIG, refer to ISPSoft user manual. Refer to section 6.19.3 for the setting in the communication-related SR and SM registers.)

9. The communication at the actual communication port will change immediately after the setting of the instruction is used. If some communication is being carried out at the moment, it will be forced to cancel. Additionally, the corresponding setting value in SM/SR will change accordingly. For details on SM/SR, refer to section 6.19.3.

10. The instruction will not make any setting for the actual communication port when the communication format setting is the same as the previous setting.

Example:

1. Take PLC COM1 for example here. Other PLC communication ports are similar to it in the communication setting.

2. The contact of the start condition is X0.0.

3. The (RS485) communication format of PLC COM1 is set to 115200, 8, E and 1.

4. The (RS485) communication mode of PLC COM1 is set to ASCII.

5. Explanation of the COMDF instruction:

Operand	Description		Content value
S_1	Baud Rate	115200 bps	1152
S_2	The number of data bits	8	8
S_3	Parity bit	E	2
S_4	The number of end bits	1	1
S_5	MODBUS format selection	ASCII	0
D	Communication port number	PLC COM1	1

Network 1
X0.0
COMDF
En
1152
S1
8
S2
2
S3
1
S4
0
S5
1
D

API	Instruction code			Operand	Function
1814		VFDRW		S_1 · S_2 · S_3 · S	Serial communication instruction exclusive for Delta AC motor drive

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●		○	○	○	○		
S_2								●	●		○	○	○	○		
S_3								●	●		○	○	○	○		
S								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Communication port number

S_2 : VFD station address

S_3 : Function code

S : Source and received data

Explanation:

1. S_1 sets a communication port number. The serial number of COM1 is 1, COM2 is 2, Card1 is 11 and Card2 is 12. If the value exceeds the valid range, the instruction will not execute the receiving of any communication data.

2. S_2 sets the station address of VFD AC motor drive. If the station address is 0, it indicates that the broadcast mode is adopted. The range is 0~254. The instruction will not be executed if the value is out of the range.

3. S_3 the communication function code and **S** the source or received data are explained in the following table.

S_3 function code	S_3 function name	**S** source and received data	Remark
0	Reset due to abnormality	Unused	Any value can be input in **S**.

S_3 function code	S_3 function name	**S** source and received data	Remark
1	Clockwise running command	Velocity value	Refer to AC motor drive user manual for the setting value and the unit.
2	Counterclockwise running command	Velocity value	
3	Stop	Unused	Any value can be input in **S**.
4	Jog clockwise running command	Unused	For the setting of the jog velocity, refer to AC motor drive user manual.
5	Jog counterclockwise running command	Unused	
6	Reading the state	Received state values	Refer to AC motor drive user manual for the meaning of the state values of the 5 bit addresses H2100 ~ H2104 of VFD.

4. Timing chart for the communication of sending and receiving data:

Description:

① → Users start/stop the execution of the instruction.

② → Transmitting data begins after the instruction is started. During the time, the communication timeout time is not measured.

③ → The reception flag is set. From the moment when the first character is received to the moment when the next

character is received, the period of time will be measured. Whenever a character is received, the time measured is cleared. The communication timeout flag will be generated if the time measured is greater than the communication timeout setting value.

④ → When the receiving of data is completed, the completion flag is set. Users must clear the flag themselves.

⑤ → Before the communication instruction is sent out once again, the instruction need be stopped for one cycle after the completion flag is watched. And then the instruction is started in the next cycle.

5. There is no limit to the number of times to use the instruction. One communication port can only be used for the output and execution of one communication instruction every time. If the receiving and sending of data is completed, the instruction need be disabled to release the communication control right.

Example of setting communication protocol:

1. Set PLC COM1 (RS485) port with the station address 2 and the communication format: ASCII, 115200, 7, N, 2 through HWCONFIG.

2. Make a basic setting using the panel of Delta C2000 AC motor drive according to the following steps.

   A. Setting 09-00 to 1, the station address of the AC motor drive is set to 1.

   B. Setting 09-01 to 115.2, RS485 baud rate of the AC motor drive is 115200.

   C. Setting 09-04 to 1, RS485 communication format of the AC motor drive is 7, N, 2.

   D. Setting 09-20 to 1, the frequency instruction is input through RS485.

   E. Setting 09-21 to 2, the running instruction is input through RS485.

Example:

By using VFDRW instruction to control the velocity, make VFD run forward at the frequency of 120Hz, run reversely at the frequency of 180Hz and then stop running.

1. Connect AS COU to VFD.

   Set D202=12000 at first. When M1 is ON, VFD starts to accelerate after receiving the command of clockwise running and then runs clockwise at the frequency of 120Hz.

2. Set D204=18000 at first. When M2 is ON, VFD starts to decelerate till it stops after receiving the command of counterclockwise running and then it runs counterclockwise at the frequency of 180Hz.

3. When M3 is ON (at the moment, the value in D206 is ineffective), VFD decelerates to stop after receiving the stop command.

4. When M4 is ON, the values of H2100~H2104 of VFD are read to D220~224.

Device	D220	D221	D222	D223	D224
Content	Error code	VFD state	Frequency command	Output frequency	Output current

The state of VFD:

As Bit2 =1, VFD executes the jog command. As Bit4~3= 11B, VFD runs counterclockwise. As the frequency command is 18000, it indicates that VFD starts to run at the frequency of 180Hz.

(For the definitions of parameter addresses in the communication protocol, refer to Delta AC motor drive user manual.)

5. The reception completion flag SM100 is ON, the values of M1~M5 are cleared so as to avoid influencing the execution of the next communication command.

   After the receiving of the data that VFD sends back is completed, the format of the data sent back from VFD will be checked. If the data format is correct, SM100 is ON. Otherwise, SM102 is ON.

API	Instruction code			Operand	Function
1815		ASDRW		S_1 , S_2 , S_3 , S	Serial communication instruction exclusive for Delta servo drive

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●		○	○	○	○		
S_2								●	●		○	○	○	○		
S_3								●	●		○	○	○	○		
S								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Communication port number

S_2 : Station address of the servo

S_3 : Function code

S : Source and received data

Explanation:

1. S_1 sets a communication port number. The serial number of COM1 is 1, COM2 is 2, Card1 is 11 and Card2 is 12. If the value exceeds the valid range, the instruction will not execute the receiving of any communication data.

2. S_2 sets the station address of the servo. If the station address is 0, it indicates that the broadcast mode is adopted. The range of the value is 0~254. The instruction will not be executed if the value is out of the valid range.

3. For details on servo parameters, refer to Delta servo operation manual.

4. S_3 the communication function code and **S** the source or received data are explained in the following table.

Applied to A, AB, A+, B series			
S_3 function code	**S_3** function name	**S** source or received data	Remark
0	Reading the servo state value	Occupies 5 consecutive devices **S**~**S**+4	Reading the state value from P0-04~ P0-08
1	Reading the servo register value	Occupies 8 consecutive devices **S**~**S**+7	Reading the value in the registers of P0-09~P0-16
2	Writing the servo register value	Occupies 8 consecutive devices **S**~**S**+7	Writing the data in the registers of P0-09~P0-16
3	Jog velocity input, clockwise running, counterclockwise running and stop	The range of velocity: 1~3000; 4999 (Clockwise running); 4998 (Counterclockwise running) ; 5000 (Stop)	Writing the data into the register of P4-05
4	Servo ON/OFF	1: Servo On; Other value: Servo OFF	Writing the data into the register of P2-30
5	Velocity command	The valid range: -5000~5000	Writing the data into the registers of P1-09 ~P1-11
6	Torque command	The valid range: -300~300	Writing the data into the registers of P1-12 ~ P1-14

Applied to A2 series			
S_3 function code	**S_3** function name	**S** source or received data	Remark
16	Reading the servo state value	Occupies 10 consecutive devices **S**~**S**+9	Reading the state value from P0-09 ~ P0-13 (32-bit value)
17	Writing data into the servo register	Occupies 8 consecutive devices **S**~**S**+7	Writing the data into the registers of P0-17 ~ P0-20 (32-bit value)
18	Writing the mapping parameter value	Occupies 8 consecutive devices **S**~**S**+7	Writing the data into the registers of P0-25 ~ P0-28 (32-bit value)
19	Jog velocity input, clockwise running, counterclockwise	The range of velocity: 1~5000; 4999 (Clockwise running);	Writing data into the register of P4-05

	running, stop	4998 (Counterclockwise running) ; 0 (Stop)	
20	Servo ON/OFF	1: Servo On; Other value: Servo OFF	Writing the data into the register of P2-30
21	Velocity command (3 sets)	Occupies 6 consecutive devices with the range of the setting value: -60000~60000	Writing data into the registers of P1-09 ~ P1-11 (32-bit value)）
22	Torque command (3 sets)	Occupies 6 consecutive devices with the range of the setting value: -300~300	Writing the data into the registers of P1-12 ~ P1-14 (32-bit value)
23	Setting the mapping targets of servo parameters	Occupies 8 consecutive devices **S**~**S**+7	Writing the data into the registers of P0-35 ~ P0-38 (32-bit value)

5. Timing chart for the communication of sending and receiving data:

Description:

①→ Users start/stop the execution of the instruction.

②→Transmitting data begins after the instruction is started. During the period, the communication timeout time is not measured.

③→ The reception flag is set. From the moment when the first character is received to the moment when the next character is received, the period of time will be measured. Whenever a character is received, the time measured is cleared. The communication timeout flag will be generated if the time measured is greater than

the communication timeout setting value.

④ → When the receiving of data is completed, the completion flag is set. Users must clear the flag themselves.

⑤ → Before one communication instruction is sent out once again, the instruction need be disabled for one cycle after the completion flag is watched. And then it is started in the next cycle.

6. There is no limit to the number of times to use the instruction. One communication port can only be used for the output and execution of one communication instruction every time. If the receiving and sending of data is completed, the instruction need be disabled to release the communication control right.

Example of setting communication protocol:

1. Set PLC COM1 (RS485) port to ASCII, 115200, 8, E, 1 as the communication format through HWCONFIG.

2. Make a basic setting using the panel of Delta ASDA-A2 servo according to the following steps.

   A. By setting P2-08 to 10, the factory setting is restored.

   B. Repower the servo after power off.

   C. Set P1-01 to 0001 (PR mode).

   D. By setting P3-00 to 2, the station address of the servo is set to 2.

   E. By Setting P3-01 to 0205, the RS485 baud rate of the servo is set to 115200.

   F. By setting P3-02 to 0004, the RS485 communication format of the servo is 8, E, 1.

   G. Power the servo on again after the setting above is completed.

Example:

By using ASDRW instruction to control the velocity, make the servo run to the relative position 5000000PUU by accelerating for 400ms from the speed of 3000.0r/min and then decelerating for 200ms.

6. Connect AS CPU to ASDA-A2.

   Set D100=1 at first. When M1 is ON, ASDA-A2 is Servo ON.

7. Set the values in D200~D207 as the following table shows, which will be written to P0-35~P0-38 of ASDA-A2. 8 consecutive devices are occupied.

Device	D200	D201	D202	D203	D204	D205	D206	D207
Setting value（16#）	05140515		06020602		06030603		053C0507	

When M2 is ON, the values in D200~D207 are written to P0-35~P0-38 of ASDA-A2.

The setting values of P0-35~P0-38 are used for setting the mapping target of P0-25~P0-28. Users can set the mapping target by referring to Delta servo operation manual.

ASDA-A2	P0-35	P0-36	P0-37	P0-38
Setting value（16#）	0514 0515	0602 0602	0603 0603	053C 0507

When the value of P0-38 is set to 053C 0507, it indicates that the mapping parameter target of P0-28 is P5-60 (16 bits) and P5-07 (16 bits).

ASDA-A2	P0-25	P0-26	P0-27	P0-28
Mapping parameter target	P5-20, P5-21	P6-02	P6-03	P5-60, P5-07
Parameter name	Acceleration/deceleration time1 Acceleration/deceleration time2	Path type1	Path1 data	Target velocity and PR command trigger

8. Setting values in D220~D227 are shown below.

Device	D220	D221	D222	D223	D224	D225	D226	D227
Setting value（16#）	0190 00C8		0000 1083		4C4B40		7530 0001	

When M3 is ON, the values in D220~D227 are written into P0-25~28 of ASDA-A2.

ASDA-A2	P0-25	P0-26	P0-27	P0-28
Setting value（16#）	0190 00C8	0000 1083	4C4B40	7530 0001

The servo starts running after acceleration time=0190 (400ms), deceleration time=00C8 (200ms), path type=1083, position command=4C4B40 (5000000PUU), target velocity=7530 (3000.0rpm) and PR command trigger =1 are set.

9. Set the values in D240~D247 as the following table shows, which will be written into P0-17~ P0-20 of ASDA-A2. 8 consecutive devices are occupied.

Device	D240	D241	D242	D243	D244	D245	D246	D247
Setting value（10#）	41		0		0		0	

When M4 is ON, the values in D240~D247 are written into P0-17~20 of ASDA-A2.

The setting values of P0-17~20 are used for setting the contents of P0-09~12.

(Users can set the contents to be displayed by referring to Delta servo operation manual.)

ASDA-A2	P0-17	P0-18	P0-19	P0-20
Setting value（10#）	41	0	0	0

When P0-17 is set to 41, it indicates that the content of P0-09 is the drive state.

ASDA-A2	P0-09	P0-10	P0-11	P0-12
Content	Drive state	Number of motor feedback pulses	Number of motor feedback pulses	Number of motor feedback pulses

10. When M5 is set to ON, the values of P0-17~ P0-20 of ASDA-A2 are read to D260~D267.

Device	D260	D262	D264	D266
Content	Drive state	Number of motor feedback pulses	Number of motor feedback pulses	Number of motor feedback pulses

When the drive state bit, Bit 4 is1, it indicates that the target position is reached.

(Refer to Delta servo operation manual for the explanation of P0-46.)

11. When the reception completion flag SM100 is set to ON, the values of M1~M5 are cleared to avoid influencing the execution of the next communication command.

After the receiving of the data that ASDA-A2 sends back is completed, the format of the data sent back from ASDA-A2 will be checked. If the data format is correct, SM100 is ON. Whereas, SM102 is ON if the data format is incorrect.

API	Instruction code			Operand	Function
1816		CCONF	P	S_1~S_{11}	Setting the parameters in the data exchange table of a communication port

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6								●	●				○	○		
S_7			●					●								
S_8								●	●				○	○		
S_9								●	●				○	○		
S_{10}								●	●				○	○		
S_{11}			●					●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1~S_{11}	Refer to the explanation of the instruction.												

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1~S_{11} : Refer to the explanation of the instruction.

Explanation:

1. The names and descriptions of **S_1~S_{11}** are listed in the following table.

Device	Name	Description	Data type	Remark
S_1	Communication port number	1=COM1, 2=COM2. If other value is filled, the instruction will not execute the modification of the operand.	WORD	
S_2	Item number in the data exchange table	The valid range: 1~32. If the value is out the range, the instruction will not execute the modification of the operand.	WORD	
S_3	The station address of the remote equipment	The valid range: 0 and 1~240. If the value is greater than 240, the instruction will take 240 as the value automatically. If the value is 0 or less than 0, the originally set station address will not be changed.	WORD	A negative number will be seen as 0.
S_4	Function code for reading data	Supports 16#01, 02, 03, 04 and 17. If the value exceeds the range, the instruction will not execute the modification of the operand. If **S_4**=16#17, the value of **S_8** will be seen as 16#17.	WORD	
S_5	Reading the remote communication address	16#0000~16#FFFF	WORD	
S_6	Reading data length	Bit function code: Supports 0~256 bits; Word function code: Supports 0~100 words. The value 0 indicates not to read data. If the value is greater than the maximum, the maximum value will be taken for execution.	WORD	
S_7	The local register for storing the data received	Bit function code: only M device is selectable. Word function code: only D device is selectable.	BOOL WORD	
S_8	Function code for writing data	Supports 16#05, 06, 0F, 10. If the value is out of the range above, the	WORD	

Device	Name	Description	Data type	Remark
		instruction will not execute the modification of the operand.		
S_9	Writing the remote communication address	16#0000~16#FFFF	WORD	
S_{10}	Writing data length	Bit function code: supports 256 bits. Word function code: supports 100 words. The value 0 indicates not to write data. If the value is greater than the maximum, the maximum value will be taken for execution.	WORD	
S_{11}	The local register into which the remote data are written	Bit function code: only M is selectable. Word function code: only D is selectable.	BOOL WORD	

2. It is suggested that the instruction be set as the pulse instruction when the instruction is used.

3. See the details about the Modbus function codes in S_4 and S_8 as below.

   The command that AS reads the data from several bit devices (which are not discrete input devices) is 1 (16#01).

   The command that AS reads the data from several bit devices (which are only discrete input devices) is 2 (16#02).

   The command that AS reads the data from several word devices (which are not input registers) is 3 (16#03).

   The command that AS reads the data from several word devices (which are only input registers) is 4 (16#04).

   The command that AS writes the state into a bit device is 5 (16#05).

   The command that AS writes the state into a word device is 6 (16#06).

   The command that AS writes the state into several bit devices is 15 (16#0F).

   The command that AS writes the data into several word devices is 16 (16#10).

   The command that AS synchronously reads from and writes the data into several word devices is 23 (16#17).

   Only the function codes mentioned above are supported. For other function code setting value, such as 0, it will be seen as invalid (including communication address, length and the start register) and the instruction will execute the data exchange function based on the original communication parameter settings.

4. When users select16#17 (for reading and writing synchronously) in **S_4** (the function code for reading), the operand **S_8** (the function code for writing) is seen as invalid and 16#17 is automatically seen for writing data.

5. When users select16#05 or 16#06 in **S_8** (the function code for writing), the operand **S_{10}** (for writing the data length) is seen as invalid and 16#05 or 16#06 is automatically seen for writing a piece of data.

6. The parameter values specified by the instruction are valid only while PLC is running. After PLC is repowered, the data in the data exchange table set through HWCONFIG are taken as default values. If there are the values of parameters to be modified in the execution, the instruction will be needed for modifying them.

7. The instruction is used to immediately set the parameters for communication connection when the data exchange function is not started. The new communication parameter will not be able to execute till the next cycle if the data exchange function is being executed and meanwhile the communication parameters of the connection number are just changed.

   For example, while the data exchange function is executing the parameters of connection 3, the instruction specifies the parameters of connection 3 to change. So the newly changed communication parameters will not be executed till the next cycle when connection 3 is reached.

8. The instruction only provides the function of revising the communication parameters. Refer to the following flags for starting and closing of the communication connection function if PLC program is used for starting or closing of the connection number.

   When users set the automatic scan function through the editing software, the start/stop flag of the connection number will automatically update the start/stop state once after the data exchange function finishes executing the scan.

   See the following table for the details on SM and the explanation.

SM No.	Attribute	Explanation of COM1 data exchange parameters
SM750	R/W	The flag for enabling the data exchange
SM752 ~ SM783	R/W	The flags for enabling the data exchange connection 1~32
SM784 ~ SM815	R	The reading success flags of the data exchange connection 1~32
SM816 ~ SM847	R	The error flags of the data exchange connection 1~32
SM No.	Attribute	Explanation of COM2 data exchange parameters
SM862	R/W	The flag for enabling the data exchange
SM864 ~ SM895	R/W	The flags for enabling the data exchange connection 1~32
SM896 ~ SM927	R	The reading success flags of the data exchange connection 1~32
SM928 ~ SM959	R	The error flags of the data exchange connection 1~32

PLC will set it to ON when the reading success flag mentioned above indicates the data receiving is completed and the data checked are correct. If an error occurs in receiving of the data or communication timeout happens, the

error flag will be set to ON. (Users can refer to error codes.) Since the reading success flag and error flag of every connection number are not ON simultaneously, PLC will not reset any of them to OFF in the data exchange.

9. See the following table for (only-read registers) SR in the data exchange function and the explanation.

SR No.	Description
SR1335	The cycle of actual connection 1~32 in the COM1 data exchange
SR1336	The cyclic connection number in the COM1 data exchange currently
SR1340 ~ SR1371	The error codes of connection 1~32 in the COM1 data exchange
SR1375	The cycle of actual connection 1~32 in the COM2 data exchange
SR1376	The cyclic connection number in the COM2 data exchange currently
SR1380 ~ SR1411	The error codes of connection 1~32 in the COM2 data exchange

10. The data exchange function does not provide the writing success flag. It is suggested that the connection number in execution can be referred to for judging whether the writing of data succeeds or not if the writing success flag need be used.

    For example, when the executed connection number is 3 in SR1336, the communication actions which are taken successively are to read communication data first and then to write communication data after reading is finished. Finally the connection number is modified into 4 after writing is completed.

11. If the length value in $\mathbf{S_6}$ and the initial device of $\mathbf{S_7}$ are out of the range of D or M device, the length value in $\mathbf{S_6}$ will be automatically revised to the value within the valid range. For example, if the length value in $\mathbf{S_6}$ is100 and the initial device of $\mathbf{S_7}$ is M8182, the value in $\mathbf{S_6}$ will be revised into 10 automatically.

12. In the following cases, the instruction will not be executed and the parameter settings in HWCONFIG will not be changed. And SM0 will be set to ON and the error code will be 16#200B in SR0.

    - When the setting values in $\mathbf{S_1}$, $\mathbf{S_2}$, $\mathbf{S_4}$ and $\mathbf{S_8}$ are out of the specified range, it is seen as an input error.
    - When $\mathbf{S_4}$ or $\mathbf{S_8}$ function code selects the bit type for reading or writing data, the local device for storing data $\mathbf{S_7}$ or $\mathbf{S_{11}}$ must select M device. If the selection is not M, it is seen as an input error.
    - When $\mathbf{S_4}$ or $\mathbf{S_8}$ function code selects the word type for reading or writing data, the local device for storing data $\mathbf{S_7}$ or $\mathbf{S_{11}}$ must select D device. If the selection is not D, it is seen as an input error.

Example: AS COM1 (RS485):

1. For the data exchange between AS CPU and DVP-ES2 CPU, the COM1 data exchange table in ISPSoft HWConfig is shown as follows.

Item No.	Detection method	Remote station address	Local device		Remote device	Length	Function code	How to start
1	Specified connection	2	D50	←	D50	50	H03	Program control
			D100	→	D100	50	H10	
2	Specified connection	2	D150	←	D150	50	H03	Program control
			D200	→	D200	50	H10	

2. Before the data exchange is started, suppose that the corresponding data between AS CPU and DVP-ES2 CPU are listed in the following table.

AS PLC (Master)	Content value	ES2 PLC (Slave)	Content value
D50~D99	0	D50~D99	1~50
D100~D149	100~149	D100~D149	0
D150~D199	0	D150~D199	150~199
D200~D249	200~249	D200~D249	0

3. The function of the data exchange between AS CPU and DVP-ES2 CPU is started as X0.1 is ON.

4. After the data exchange is started, the corresponding data between AS CPU and DVP-ES2 CPU are changed as shown in the following table.

AS PLC (Master)	Content value	ES2 PLC (Slave)	Content value
D50~D99	1~50	D50~D99	1~50
D100~D149	100~149	D100~D149	100~149
D150~D199	150~199	D150~D199	150~199
D200~D249	200~249	D200~D249	200~249

5. As X0.0 is ON, the COM1 data exchange table parameters of AS CPU are modified as below.

Item No.	Detection method	Remote station address	Local device		Remote device	Length	Function code	How to start
1	Specified connection	3	D5000	←	D500	1	H04	Program control
			D6000	→	D600	1	H06	
2	Specified	3	M100	←	M0	10	H01	Program

Item No.	Detection method	Remote station address	Local device		Remote device	Length	Function code	How to start
	connection		M110	➔	M50	10	H0F	control

6. Due to the change of COM1 data exchange table parameters, the corresponding data between AS CPU and DVP-ES2 CPU are modified as below.

AS PLC (Master)	Content value	ES2 PLC (Slave)	Content value
D5000	3000	D500	3000
D6000	4000	D600	4000
M100~M109	ON	M0~M9	ON
M110~M119	OFF	M50~M59	OFF

API	Instruction code	Operand	Function
1817	MODRWE	S_1 · S_2 · S_3 · S_4 · S · n · D	Reading and writing Modbus data without any flag used

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S								●								
n								●	●				○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
S	●	●			●	●							
n		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Communication port number

S_2 : Unit address

S_3 : Function code

S_4 : Device address

S : Register involved in the reading and writing of the data

n : Data length

D : The flag for setting the completion of the reading and writing the data

Explanation:

1. **S_1** sets the serial number of a communication port. The serial number of COM1 is 1, COM2 is 2, Card1 is 11 and Card2 is 12. If the value exceeds the valid range, the instruction will not execute the receiving of any communication data.

2. Refer to the explanation of API1808 MODRW instruction for the meaning of **S_2, S_3, S_4, S and n**.

3. **D** sets the communication state flags when the communication instruction completes the communication. 3 consecutive devices are occupied by the flags. Users must reset them to OFF by themselves. The explanations of the flag states are shown in the following table.

Operand	Description
D	The Receiving of the data succeeds.
D +1	An error occurs in the receiving of the data.
D +2	Reception timeout flag

NOTE: Only one flag will be set to ON among the three state flags and meanwhile the corresponding special flags (SM) will be also set to ON every time the communication is completed. Refer to section 6.19.3 for the use of the special flag.

4. The timing for sending the instruction is when the instruction is started. Users must disable the instruction for a scan cycle after the communication is completed. And then the next communication instruction can be sent just after the instruction is restarted.

5. The communication action and control sequence of the instruction are similar to API1808 MODRW instruction. The only difference between them is that the sending of the communication command can be done without users' control over the flag for sending the data.

Example:

The MODRW program and MODRWE program are compared by taking PLC COM1 and function code 03 to read 8 pieces of data from D20 of DVP-ES2 for example. The same is for other PLC communication ports. For the use of other function codes, refer to API1808 MODRW instruction and the following example.

1. The device address of D20 in DVP-ES2 CPU, 16#1020 and the content values in D20~D27 are shown in the table below.

Device	D20	D21	D22	D23	D24	D25	D26	D27
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

2. AS reads the content values from D20~D27 in DVP-ES2 CPU through the communication.

Method 1: Using MODRW instruction

The data in D20~D27 of DVP-ES2 are read when SM96 is ON and X0.0 is ON.

Explanation of MODRW operands

Operand	Description	Device
S_1	Unit address	16#0001
S_2	Function code	16#0003
S_3	Reading the device address of D20	16#1020
S	The initial register for storing the data read	D10
n	Reading the data length	8

The communication response between AS CPU and DVP-ES2

ASCII mode: (The ASCII codes do not need to be converted intentionally and they are all expressed in the 16# values.)

- AS sends the communication command: “01 03 10 20 00 08 C4 CR LF”
- AS receives the communication command: “01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC AA CR LF”

RTU mode:

- AS sends the communication command: “ 01 03 10 20 00 08 41 06”
- AS receives the communication command: “01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC 90 FE”

SM100 is ON due to no error in the data.

After the receiving of the data sent back from DVP-ES2 is completed, the data format sent back from DVP-ES2 will be confirmed and judged whether it is correct. If no error occurs in the format, SM100 is ON. Whereas, SM102 is ON if the data format sent back is incorrect. SM104 is ON if there is no response.。

Method 2: Using MODRWE instruction

The data in D20~D27 of DVP-ES2 are read when X0.0 is ON.

Explanation of MODRWE operands

Operand	Description	Device
S1	Communication port number	16#0001
S2	Unit address	16#0001
S3	Function code	16#0003
S4	Reading the device address of D20	16#1020
S	The initial register for storing the data read	D10
n	Reading the data length	8
D	The flag for setting the completion of the reading and writing data	M0

The communication response between AS CPU and DVP-ES2

ASCII mode: (The ASCII codes do not need to be converted intentionally and they are all expressed in the 16# values.)

- AS sends the communication command: “01 03 10 20 00 08 C4 CR LF”
- AS receives the communication command: “01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC AA CR LF”

RTU mode:

- AS sends the communication command: “01 03 10 20 00 08 41 06”
- AS receives the communication command: “01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC 90 FE”

M0 is ON due to no error in the data.

After the receiving of the data sent back from DVP-ES2 is completed, the data format sent back from DVP-ES2 will be confirmed and judged whether it is correct. If no error occurs in the format, M0 is ON. Whereas, M1 is ON if the data format sent back is incorrect. M2 is ON if there is no response. And meanwhile the corresponding special flags SM100, SM102 and SM104 are ON as well.

3. The content values in D10~D17 of AS CPU:

Device	D10	D11	D12	D13	D14	D15	D16	D17
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

Additional remarks:

1. The data type is ARRAY [3] of BOOL if D operand is declared via ISPSoft.
2. If **D**+2 exeeds the device range, the instruction will not be executed, SM0 will be ON and the error code will be 16#2003 in SR0.

6.19.3 Descriptions on the Communication-related Flags and Registers

Communication-related flags (SM)

Flag				Description	Action
COM1	COM2	Card 1	Card 2		
SM96	SM97	SM76	SM77	Data sending request flag If users want to use the instruction to send and receive the data, they have to use the pulse instruction to set the flag to ON. When the instruction is executed, the PLC sends and receives the data. After the sending of the data is complete, the system automatically resets the flag to OFF.	Users set the flag to ON, and the system automatically resets it to OFF.
SM98	SM99	SM78	SM79	When the flag is ON, the PLC is waiting to receive the data.	The system automatically sets the flag to ON and resets it to OFF.
SM100	SM101	SM80	SM81	Reception complete flag After the receiving of the data is complete, the system automatically sets the flag to ON. When the flag is ON, the data received can be processed. After the processing of the data received is complete, users have to reset flags to OFF.	The system automatically sets the flag to ON, and users reset it to OFF.
SM102	SM103	SM82	SM83	Data receiving error flag An error occurs during the reception of the data by using the data receiving instruction.	The system automatically sets the flag to ON, and users reset it to OFF.
SM104	SM105	SM84	SM85	Communication timeout error flag If users set the communication timeout (in SR) and no data is received after the timeout period, the flag is ON. After the problem is solved, users have to reset the flag to OFF.	The system automatically sets the flag to ON, and users reset it to OFF.
SM106	SM107	SM86	SM87	The choice between the 8-bit processing mode and the 16-bit processing mode ON: The 8-bit processing mode OFF: The 16-bit processing mode	Users set the flag to ON and reset it to OFF.
SM210	SM212	-	-	Communication mode ON: RTU mode OFF: ASCII mode This setup can be done in HWCONFIG of ISPSoft, not available for Card1 and Card 2.	Users set the flag to ON and reset it to OFF.
SM209	SM211	SM90	SM91	Communication protocol changed flag The communication protocol changes in accordance with the setting values in SR. If the flag is set to ON, the communication protocol changes in accordance with the setting values in SR and then the system automatically resets the flag to OFF. NOTE: this change will not affect the parameters set in HWCONFIG. When the PLC is powered-on again, the PLC will operate according to the communication protocol set in the HWCONFIG.	Users set the flag to ON and reset it to OFF.

NOTE: the above flags are non-latching types.

Communication-related registers (SR)

Special data register				Description
COM1	**COM2**	**Card 1**	**Card 2**	
SR201	SR202	SR176	SR178	The communication port address
SR209	SR212	SR177	SR179	Communication protocol For details, please refer to the following table to set up the communication formatfor a serial communication port.
SR210	SR213	SR182	SR183	Communication timeout, unit: ms Suppose the setting value is larger than 0. When the communication instruction is executed and the PLC is in the receiving state, but no data is received after the timeout period or the intervening time of the two characters exceeds the setting value, the timeout flag will be ON. The register can be set to 0 and the communication timeout monitoring will be disabled. The MODRW instruction should be set between 100~32767 (ms).

Setting the communication format for a serial communication port

b0	Data length			7 (value=0)	8 (value=1)
b2~b1	Parity bits			00 : None	
				01 : Odd	
				10 : Even	
b3	stop bits			1 bit (value=0)	2 bits (value=1)
b7~b4	0001	(16#1)	:	4800	
	0010	(16#2)	:	9600	
	0011	(16#3)	:	19200	
	0100	(16#4)	:	38400	
	0101	(16#5)	:	57600	
	0110	(16#6)	:	115200	
	0111	(16#7)	:	230400	Not available for RS-232
	1000	(16#8)	:	500000	Not available for RS-232
	1001	(16#9)	:	921000	Not available for RS-232
	1111	(16#F)	:	Self-defined*1	
b8~b15	Undefined (reserved)				

*1: Users can set up the baud rate via the HWCONFIG of ISPSoft.

The data transmission speed is as follows.

Baud rate (bps)	RTU timeout timer (ms)	Baud rate (bps)	RTU timeout timer (ms)
4800	9	115200	1
9600	5	230400	1
19200	3	-	-
38400	2	-	-
57600	1	-	-

6.20 Other Instructions

6.20.1 List of Other Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1900	WDT	–	✓	Watchdog timer
1901	DELAY	–	✓	Delaying the execution of the program
1902	GPWM	–	–	General pulse width modulation
1904	EPUSH	–	✓	Storing the contents of the index registers
1905	EPOP	–	✓	Reading the data into the index registers

6.20.2 Explanation of Other Instructions

API	Instruction code			Operand	Function
1900		WDT	P	–	Watchdog timer

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	–

Symbol:

Explanation:

1. In the AS series PLC, there is a watchdog timer which is used to monitor the operation of the system.

2. The instruction WDT is used to reset the watchdog timer in the PLC. If the program scanning time exceeds 200 milliseconds, the error LED indicator is ON, and the PLC stops running.

3. The particular point when the watchdog timer acts:

   - The system is abnormal.

   - The execution of the program takes much time, and therefore the scam time is larger than the setting value of the watchdog timer. There are two way users can use to improve the situation.

     (a) Using the instruction WDT

     (b) Please refer to ISPSoft User Manual for more information about changing the setting value of the watchdog timer.

Example:

Suppose the program scanning time is 300 milliseconds. After the program is divided into two parts, and the instruction WDT is inserted between these two parts, the time it takes to scan either the first part of the program or the second part of the program is less than 200 milliseconds.

API	Instruction code			Operand	Function
1901		DELAY	P	**S**	Delaying the execution of the program

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	–

Symbol:

S : Delay

Explanation:

After the instruction DELAY is executed, the execution of the program following the DELAY is delayed for a period of time specified by users.

The unit of **S** is 0.1 milliseconds.

Example:

When X0.0 is ON, the instruction DELAY is executed. The execution of the program following DELAY is delayed for two milliseconds. That is, Y0.0 is ON and the states of Y0.0~Y0.15 are refreshed two milliseconds after the instruction DELAY is executed.

Additional remark:

1. If **S** I less than 0, there is no delay.

2. If **S** is larger than 1000, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. Users can adjust the delay according to the practical condition.

4. The delay will increase due to the communication or other influences.

API	Instruction code			Operand	Function
1902		GPWM		S_1 , S_2 , D	General pulse width modulation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●			○	○				
S_2		●			●	●		●								
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
–	AS	–

Symbol:

S_1 : Pulse width

S_2 : Pulse cycle

D : Output device

Explanation:

1. When the instruction GPWM is executed, every pulse with a width specified by S_1 and with a cycle specified by S_2 is output from the device specified by **D**.

2. The pulse width specified by S_1 is t. t should be within the range between 0 and 3276 milliseconds.

3. The pulse cycle specified by S_2 is T. T should be within the range between 1 and 32767 milliseconds, and S_1 should be less than S_2.

4. S_2+1 and S_2+2 are parameters for system use. Please do not occupy them.

5. If S_1 is less than 0, there is no pulse output. If S_1 is larger than S_2, the output device keeps ON.

6. S_1 and S_2 can be altered during the execution of the instruction GPWM.

7. If the conditional contact is not enabled, there is no pulse output.

8. When the on-line editing is used, please reset the conditional contact to initialize the instruction.

Example:

When the program is executed, the values in D0 and D2 are 1000 and 2000 respectively. When X0.0 is ON, the pulses illustrated below are output from Y0.0. When X0.0 is OFF, Y0.0 is OFF.

Additional remark:

1. The instruction counts by the scan cycle. Therefore, the maximum error is one scan cycle. Besides, $\mathbf{S_1}$, $\mathbf{S_2}$, and ($\mathbf{S_2}$-$\mathbf{S_1}$) should be larger than the scan cycle. Otherwise, an error occurs when the instruction GPWM is executed.

2. If the instruction is used in the function block or the interrupt task, the inaccurate pulse output will occur.

3. If users declare the operand $\mathbf{S_2}$ in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

API	Instruction code			Operand	Function
1904		EPUSH	P	**D**	Storing the contents of the index registers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	–

Symbol:

D : Device in which the value in the index register is stored

Explanation:

1. The values in E0~E9 are stored in the devices specified by the value in **D**. The allowed value for **D** is in the range of 0~99. Values exceed the range will not be executed.

2. The execution of the instruction involves sixteen devices, and the last six devices are for system use. If the instruction is executed and the number of times the data is stored is n, which is the value in **D**, the data in E0~E9 is stored in **D**+(16*n+1)~**D**+(16*n+16), and the value in **D** becomes n+1.

3. The range of device D should be 16x100+1.

4. This instruction uses pulse instruction to interact with the stack, pushing a value onto the stack. Therefore, users need to reset the contact upon the next operation.

5. When the instruction is used with the instruction EPOP, the value which is stored last in the device specified by the value in **D** is read first, following the LIFO (last in first out) principle.

Example:

Suppose the value in D0 is 0.

When X0.0 is ON for the first time, the data in E0~E9 is transmitted to D1~D10 and the value in D0 becomes 1.

When X0.0 is switched from OFF to ON for the second time, the data in E0~E9 is transmitted to D17~D26, and the value in D0 becomes 2.

When X0.0 is switched from OFF to ON for the nth time, the data in E0~E9 is transmitted to (n*16)+1~(n*16)+10.

Additional remark:

1. If the value in **D** is less than 0 or greater than 99, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the range of device is not sufficient for the **D**+((the value in **D**)+1)*16-1), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1905		EPOP	P	**D**	Reading the data into the index registers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	–

Symbol:

D : Device from which the value is read

Explanation:

1. The values in the devices specified by the value in **D** are read into E0~E9, and the value in **D** decreases by one. The allowed value for **D** is in the range of 1~100. Values exceed the range will not be executed.

2. The execution of the instruction involves sixteen devices, and the last six devices are for system use. If the instruction is executed and the number of times the data is stored is n, which is the value in **D**, the data in E0~E9 is stored in **D**+16*(n-1)+1~**D**+16*(n-1)+10, and the value in **D** becomes n-1.

3. This instruction uses pulse instruction to interact with the stack, taking the TOP VALUE from the stack and assigns it to the specified variable. Therefore, users need to reset the contact upon the next operation.

4. The value which is stored last in the device specified by the value in **D** is read first, following the LIFO (last in first out) principle.

Example:

When X0.0 is ON, the value in D0 is set to 0, and the values in E0~E9 are transmitted to D1~D10. After the execution of FB0 is complete, the values in D1~D9 are read into E0~E9 via the instruction EPOP.

Additional remark:

1. If the value in **D** is less than 0 or greater than 100, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the range of device is not sufficient for the **D**+((the value in **D**) *16-1), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

6.21 String Processing Instructions

6.21.1 List of String Processing Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2100	BINDA	DBINDA	✓	Converting the singed decimal number into the ASCII code
2101	BINHA	DBINHA	✓	Converting the binary hexadecimal number into the hexadecimal ASCII code
2102	BCDDA	DBCDDA	✓	Converting the binary-coded decimal number into the ASCII code
2103	DABIN	DDABIN	✓	Converting the signed decimal ASCII code into the signed decimal binary number
2104	HABIN	DHABIN	✓	Converting the hexadecimal ASCII code into the hexadecimal binary number
2105	DABCD	DDABCD	✓	Converting the ASCII code into the binary-coded decimal number
2106	$LEN	–	✓	Calculating the length of the string
2109	$FSTR	–	✓	Converting the floating-point number into the string
2110	$FVAL	–	✓	Converting the string into the floating-point number
2111	$RIGHT	–	✓	The retrieve of the characters in the string begins from the right.
2112	$LEFT	–	✓	The retrieve of the characters in the string begins from the left.
2113	$MIDR	–	✓	Retrieving a part of the string
2115	$SER	–	✓	Searching the string
2116	$RPLC	–	✓	Replacing the characters in the string
2117	$DEL	–	✓	Deleting the characters in the string
2118	$CLR	–	✓	Clearing the string
2119	$INS	–	✓	Inserting the string

6.21.2 Explanation of String Processing Instructions

API	Instruction code			Operand	Function
2100	D	BINDA	P	**S · D**	Converting the singed decimal number into the ASCII code

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source value

D : Device in which the conversion result is stored

Explanation:

1. The signed decimal binary number in **S** is converted into the ASCII code, and the conversion result is stored in **D**.

2. The instruction supports SM690, which controls the ending character.

3. The value in **S** used in the 16-bit instruction should be a binary number and should be within the range between -32768 and 32767. The operand **D** occupies four word devices. The data is converted as follows.

b15 b0		b15 b8	b7 b0	
1234	➔	16#32	16#31	**D** + 0
		16#34	16#33	**D** + 1
		16#00	16#00	**D** + 2

If SM690 is OFF, the ending character 16#0000 is stored in **D**+2. If SM690 is ON, the value in **D**+2 is unchanged without the ending character.

Besides, if the value in S is a positive value, only value will be inputted but not the sign character in D. If the value in S is a negative value, the sign character in D "-" will be stored in 16#2D. For example, if the value in S is -12345 and SM690 is OFF, the conversion result is as follows.

b15 b0		b15 b8	b7 b0	
-1234	➔	16#31	16#2D (-)	**D** + 0
		16#33	16#32	**D** + 1
		16#00	16#34	**D** + 2

4. The value in S used in the 32-bit instruction should be a binary number and should be within the range between -2147483648 and 2147483647. The operand D occupies six word devices. The data is converted as follows.

b31 b0		b15 b8	b7 b0	
12345678	➔	16#32	16#31	**D** + 0
		16#34	16#33	**D** + 1
		16#36	16#35	**D** + 2
		16#38	16#37	**D** + 3
		16#00	16#00	**D** + 4

If SM690 is OFF, the ending character 16#0000 is stored in D+4. If SM690 is ON, the value in D+4 is unchanged without the ending character.

Besides, if the value in S is a positive value, only value will be inputted but not the sign character in D as the previous example shown. If the value in S is a negative value, the sign character in D "-" will be stored in 16#2D. For example, if the value in S is -12345678, and SM690 is OFF, the conversion result is as follows.

b31 b0		b15 b8	b7 b0	
-12345678	➔	16#31	16#2D	**D** + 0
		16#33	16#32	**D** + 1
		16#35	16#34	**D** + 2
		16#37	16#36	**D** + 3
		16#00	16#38	**D** + 4

Example 1:

Suppose the value in L0 is 5126 and SM690 is OFF. When the PLC runs, the values are D0=16#3135, D1=16#3632, D2=16#0000.

D10		b15 b8	b7 b0	
5126	➔	16#31	16#35	D0
		16#36	16#32	D1
		16#00	16#00	D2

Example 2:

Suppose the value in D10 is -3842563 and SM690 is OFF. When the PLC runs, the values are D0=16#332D, D1=16#3438, D2=16#3532, D3=16#3336, D4=16#0000.

D10		b15 b8	b7 b0	
-3842563	➔	16#33	16#2D	D0
		16#34	16#38	D1
		16#35	16#32	D2
		16#33	16#36	D3
		16#00	16#00	D4

Additional remark:

1. If value in the device **D** is not sufficient for conversion, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the operand **D** used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [4] of WORD/INT.
3. If the operand **D** used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [6] of WORD/INT.

API	Instruction code			Operand	Function
2101	D	BINHA	P	**S · D**	Converting the binary hexadecimal number into the hexadecimal ASCII code

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source value

D : Device in which the conversion result is stored

Explanation:

6

1. The hexadecimal binary number in **S** is converted into the ASCII code, and the conversion result is stored in **D**.

2. The instruction supports SM690, which controls the ending character.

3. The value in **S** used in the 16-bit instruction should be within the range between 16#0000 and 16#FFFF, and should be a four-digit binary number. The operand **D** occupies three word devices. The data is converted as follows.

b15 b0		b15 b8	b7 b0	
02A6	➔	16#32	16#30	**D** + 0
		16#36	16#41	**D** + 1
		16#00	16#00	**D** + 2

If SM690 is OFF, 16#0000 is stored in **D**+2. If SM690 is ON, the value in **D**+2 is unchanged. For example, if the value in **S** is 16#02A6 and SM690 is OFF, the conversion result is as follows.

4. The value in **S** used in the 32-bit instruction should be within the range between 16#00000000 and 16#FFFFFFFF, and should be an eight-digit binary number. The operand **D** occupies five word devices. The data is converted as follows.

b31 b0		b15 b8	b7 b0	
03AC 6251	➔	16#33	16#30	**D** + 0
		16#43	16#41	**D** + 1

16#32	16#36	**D** + 2
16#31	16#35	**D** + 3
16#00	16#00	**D** + 4

If SM690 is OFF, 16#0000 is stored in D+4. If SM690 is ON, the value in D+4 is unchanged. For example, if the value in S is 16#03AC625E and SM690 is OFF, the conversion result is as follows.

Example 1:

Suppose the value in D10 is 16#9C06 and SM690 is OFF. When PLC runs, the values are D0=16#4339, D1=16#3630, D2=16#0000.

D10		b15 b8	b7 b0	
16#9C06	➔	16#43	16#39	D0
		16#36	16#30	D1
		16#00	16#00	D2

Example 2:

Suppose the value in D10 is 16#7B3C581F and SM690 is OFF. When PLC runs, the values are D0=16#4237, D1=16#4333, D2=16#3835, D3=16#4631, D4=16#0000.

D10		b15 b8	b7 b0	
16#7B3C 581F	➔	16#42	16#37	D0
		16#43	16#33	D1
		16#38	16#35	D2
		16#46	16#31	D3
		16#00	16#00	D4

Additional remark:

1. If **D**+2 used in the 16-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
2. If **D**+4 used in the 32-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the operand **D** used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [3] of WORD/INT.
4. If the operand **D** used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [5] of WORD/INT.

API	Instruction code			Operand	Function
2102	D	BCDDA	P	**S · D**	Converting the binary-coded decimal number into the ASCII code

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○	○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source value

D : Device in which the conversion result is stored

Explanation:

1. The binary-coded decimal number in **S** is converted into the ASCII code, and the conversion result is stored in **D**.

2. The instruction supports SM690, which controls the ending character.

3. The binary-coded decimal value in **S** used in the 16-bit instruction should be within the range between 0 and 9999, and should be a four-digit binary-coded decimal value. The operand **D** occupies three word devices. The data is converted as follows.

b15~12 b11~8		b7~4 b3~0			b15 b8	b7 b0	
0	8	7	6	➔	16#38	16#30	**D** + 0
Thousands Hundreds		T ens	U nits		16#36	16#37	**D** + 1
					16#00	16#00	**D** + 2

If SM690 is OFF, the ending character 16#0000 is stored in **D**+2. If SM690 is ON, the value in **D**+2 is unchanged without the ending character.

4. The binary-coded decimal value in S used in the 32-bit instruction should be within the range between 0 and 99999999, and should be an eight-digit binary-coded decimal value. The operand D occupies five word devices. The data is converted as follows.

b31 b0		b15 b8	b7 b0	
0876 5432	➔	16#38	16#30	**D** + 0
		16#36	16#37	**D** + 1
		16#34	16#35	**D** + 2
		16#32	16#33	**D** + 3
		16#00	16#00	**D** + 4

5. If SM690 is OFF, the ending character 16#0000 is stored in D+5. If SM690 is ON, the value in D+5 is unchanged without the ending character.

6. Even if the first digit of binary-coded decimal value in S is 0, it will be converted into the ASCII code 0 (16#30).

Example 1:

Suppose the binary-coded decimal value in D10 is 1295 and SM690 is OFF. When PLC runs, the values are D0=16#3231, D1=16#3539, D2=16#0000.

D10		b15 b8	b7 b0	
1295	➔	16#32	16#31	D0
		16#35	16#39	D1
		16#00	16#00	D2

Example 2:

Suppose the binary-coded decimal value in D10 is 00578352 and SM690 is OFF. When PLC runs, the values are D0=16#3030, D1=16#3735, D2=16#3338, D3=16#3235, D4=16#0000.

D10		b15 b8	b7 b0	
0057 8352	➔	16#30	16#30	D0
		16#37	16#35	D1
		16#33	16#38	D2
		16#32	16#35	D3
		16#00	16#00	D4

Additional remark:

1. If the value in S used in the 16-bit instruction is not within the range between 0 and 9999, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200D. (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.)

2. If the value in S used in the 32-bit instruction is not within the range between 0 and 99999999, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200D. (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.)

3. If D+2 used in the 16-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.

4. If D+4 used in the 32-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.

5. If the operand D used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

6. If the operand D used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [5] of WORD/INT.

API	Instruction code			Operand	Function
2103	D	DABIN	P	**S · D**	Converting the signed decimal ASCII code into the signed decimal binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S		●			●	●		●	●						○	
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source value

D : Device in which the conversion result is stored

Explanation:

1. The signed decimal ASCII code in **S** is converted into the signed decimal binary number, and the conversion result is stored in **D**.

2. The operand **S** used in the 16-bit instruction occupies three word devices, and the decimal ASCII code in **S** should be within the range between -32768 and 32767. If the value in **S** is a string and without the ending character 16#00, the conversion can be up to 5 digits (sign excluded).

	b15 b8	b7 b0
S + 0	" 2 "	" 1 "
S + 1	" 4 "	" 3 "
S + 2	16#00	16#00

➔

b15 b0	
1234	**D**

3. If the first character is " " (a space), the sign is a positive sign. If the first character is "-", the sign is a negative sign. Take the string "2345" for example.

	b15 b8	b7 b0
S + 0	" 2 "	" - "
S + 1	" 4 "	" 3 "
S + 2	16#00	" 5 "

➔

b15 b0	
-2345	**D**

4. The operand **S** used in the 32-bit instruction occupies six word devices, and the decimal ASCII code in **S** should be within the range between -2147483648 and 2147483647. If the value in **S** is a string and without the ending character 16#00, the conversion can be up to 10 digits (sign excluded).

	b15 b8	b7 b0
S + 0	" 2 "	" 1 "
S + 1	" 4 "	" 3 "
S + 2	" 6 "	" 5 "
S + 3	" 8 "	" 7 "
S + 4	" 0 "	" 9 "
S + 5	16#00	16#00

➔

b31 b0	
1234567890	**D**

5. If the first character is " " (a space), the sign is a positive sign. If the first character is "-", the sign is a negative sign. Take the string "-234567890" for example.

	b15 b8	b7 b0
S + 0	' 2 '	' - '
S + 1	' 4 '	' 3 '
S + 2	' 6 '	' 5 '
S + 3	' 8 '	' 7 '
S + 4	' 0 '	' 9 '
S + 5	16#00	16#00

➔

b31 b0	
-234567890	**D**

6. If the first digit of the string in the device S is blank (16#20) + sign (16#2B), it will be seen as 0. For the second digit, if the number is not 0~9, it will be seen as the ending of a string and no error message will be shown. For example if the word order is 16#20→16#31→16#32→16#2B, the conversion result is 12.

7. The string range in the device **S** for 16-bit instruction is 1~6 (positive/negative signs included) and for 32-bit instruction is 1~11 (positive/negative signs included).

8. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example 1:

Suppose the values are D20=16#302D, D21=16#3433, D22=16#0035. When PLC runs, the value is D0=-345.

	b15 b8	b7 b0
D20	16#30	16#2D
D21	16#34	16#33
D22	16#00	16#35

➔

b15 b0	
-345	D0

Example 2:

Suppose the values are D20=16#322D, D21=16#3433, D22=16#3635, D23=16#2037, D24=16#0000, D25=16#0000. When PLC runs, the value is D0=-234567.

	b15 b8	b7 b0		b31 b0	
D20	16#32	16#2D	➔	-234567	D0
D21	16#34	16#33			
D22	16#36	16#35			
D23	16#20	16#37			
D24	16#00	16#00			
D25	16#00	16#00			

6

Example 3:

Suppose the string value in **S** is 12. When PLC runs, the value is D0=12.

Additional remark:

1. If the value of the first word in **S** is an ending character (16#00), the instruction will see the value in **S** as 0 (16#30).

2. If the value of the first digit in **S** is 16#20 (blank) or 16#2B (+) or 16#2D (-) and the second digit is 16#00, the instruction will see the value in **S** as 0 (16#30).

3. Even if the first digit of binary-coded decimal value in **S** is 0, it will be converted into the ASCII code 0 (16#30).

4. The value of the first digit in **S** only supports ASCII codes, 16#30~16#39 (0~9), 16#200 (blank), 16#2D (negative

sign), 16#2B (positive sign), 16#00 (ending character). If the value of the first digit in **S** is not ASCII code, the instruction will not be executed, SM0 is ON, and the error code in SR0 is 16#2003.

5. Except the first digit, if the value of other digits in **S** is not ASCII codes, 16#30~16#39 or 16#00, it will be seen as 16#00.

6. If the value in **S** exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003. The instruction will not be executed.

7. If the operand **S** used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [3] of WORD/INT.

8. If the operand **S** used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [6] of WORD/INT.

API	Instruction code			Operand	Function
2104	D	HABIN	P	**S · D**	Converting the hexadecimal ASCII code into the hexadecimal binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S		●			●	●		●	●						○	
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source value

D : Device in which the conversion result is stored

Explanation:

1. The hexadecimal ASCII code in **S** is converted into the hexadecimal binary number, and the conversion result is stored in **D**.

2. The operand **S** used in the 16-bit instruction occupies two word devices. If the value in **S** is a string and without the ending character 16#00, the conversion can be up to 4 digits (sign excluded). The hexadecimal ASCII code in **S** should be within the range between 0000 and FFFF. If **S** is a string, the string should be within the range between “0” and “FFFF”.

3. The operand **S** used in the 32-bit instruction occupies four word devices. If the value in **S** is a string and without the ending character 16#00, the conversion can be up to 8 digits (sign excluded). The hexadecimal ASCII code in **S** should be within the range between 00000000 and FFFFFFFF. If **S** is a string, the string should be within the range between “0” and “FFFFFFFF”.

	b15 b8	b7 b0		b31 b0	
S + 0	" C "	" 5 "	➔	16#5CB8 07E1	**D**
S + 1	" 8 "	" B "			
S + 2	" 7 "	" 0 "			
S + 3	" 1 "	" E "			

4. The string range in the device **S** for 16-bit instruction is 1~4 and for 32-bit instruction is 1~8.

Example 1:

Suppose the values are D20=16#3641, D21=16#4633 (ASCII 16#A63F). When PLC runs, the value is D0=-22977.

	b15 b8	b7 b0		b15 b0	
D20	16#36	16#41	➔	16#A63F	D0
D21	16#46	16#33			

Example 2:

Suppose the values are D20=16#4634, D21=16#3244, D22=16#3738, D23=16#3035 (ASCII 16#4FD28750). When PLC runs, the value is (D1, D0)=16#4FD28750.

	b15 b8	b7 b0		b31 b0	
D20	16#46	16#34	➔	16#4FD2 8750	D0
D21	16#32	16#44			
D22	16#37	16#38			
D23	16#30	16#35			

Example 3:

Suppose the string value in **S** is A. When PLC runs, the value is D20=16#A=10.

Example 4:

Suppose the values are D20=16#3641, D21=16#4600 (ASCII 16#00A6). When PLC runs, the value is D0=166.

	b15 b8	b7 b0		b15 b0	
D20	16#36	16#41	➔	16#00A6	D0
D21	16#46	16#00			

Additional remark:

1. If the ASCII code in **S** is not within the range between 16#30~16#39 (“0”~”9”), or within the range between 16#41~16#46 (“A”~”F”), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the operand **S** used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

3. If the operand **S** used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [4] of WORD/INT.

API	Instruction code			Operand	Function
2105	D	DABCD	P	**S · D**	Converting the ASCII code into the binary-coded decimal number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S		●			●	●		●	●						○	
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	AS

Symbol:

S : Source value

D : Device in which the conversion result is stored

Explanation:

6

1. The ASCII code in **S** is converted into the binary-coded decimal number, and the conversion result is stored in **D**.

2. The operand **S** used in the 16-bit instruction occupies two word devices, and the ASCII code in **S** should be within the range between 0000 and 9999. If **S** is a string, the string should be within the range between “0” and “9999”.

3. The operand **S** used in the 32-bit instruction occupies four word devices, and the ASCII code in **S** should be within the range between 0000000 and 99999999. If **S** is a string, the string should be within the range between “0” and “99999999”.

	b15 b8	b7 b0		b15~b12	b11~b8	b7~b4	b3~b0	
S + 0	" 8 "	" 9 "	➔	5	4	3	2	**D**
S + 1	" 6 "	" 7 "		9	8	7	6	**D**+1
S + 2	" 4 "	" 5 "						
S + 3	" 2 "	" 3 "						

4. If the value in **S** is 16#20 the value is processed as 16#30. If the value in **S** is 16#00, the value is processed as an ending character..

5. If **S** used in the 16-bit instruction is a string, the number of characters contained in the string should be within the range between 1 and 4. If **S** used in the 32-bit instruction is a string, the number of characters contained in the string should be within the range between 1 and 8.

Example 1:

Suppose the values are D20=16#3820, D21=16#3637 (ASCII 876). When PLC runs, the value will be converted into Y0=16#876.

	b15 b8	b7 b0		b15~b12	b11~b8	b7~b4	b3~b0	
D20	16#38	16#20	➔	0	8	7	6	Y0
D21	16#36	16#37						

Example 2:

Suppose the values are D20=16#3738, D21=16#3536, D22=16#3334, D23=16#3132 (ASCII 87654321). When PLC runs, the value is (D11, D10)= 16#87654321.

	b15 b8	b7 b0		b31 b0	
D20	16#37	16#38	➔	8765 4321	D0
D21	16#35	16#36			
D22	16#33	16#34			
D23	16#31	16#32			

Example 3:

Suppose the string value in **S** is 1. When PLC runs, the value is D20=16#0001.

Example 4:

Suppose the string value in **S** is 1234. When PLC runs, the value is (D21, D20)= 16#00001234.

Additional remark:

1. If the ASCII code in **S** is not ASCII codes 16#30~16#39, 16#20, 16#00, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S** is a string and the number of characters contained in the string exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the operand **S** used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

4. If the operand **S** used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [4] of WORD/INT.

API	Instruction code			Operand	Function
2106		$LEN	P	**S · D**	Calculating the length of the string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●								
D		●			●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : String

D : Length of the string

Explanation:

1. The length of the string in **S** is calculated, exclusive of 16#00 with which the string ends. The length of the string is stored in **D**.

2. The value stored in D should be within the range between 0~32767. If exceeding the range, the value in D will be seen as 32767.

If the data in **S**~**S**+4 is ABCDEFGHI, the calculation result is as follows.

Example 1:

Suppose **S** is the string “DELTA”. When the PLC runs, the value in D0 is 5.

Example 2:

Suppose the data in D0~D2 is as follows. When the PLC runs, the value in L0 is 5.

D0	16#45 (E)	16#44 (D)
D1	16#54 (T)	16#4C (L)
D2	16#00 (Ending character)	16#41 (A)

Additional remark:

1. If the string does not end with 16#00, the instruction will execute to the maximum value 32767.

2. If the length of the value exceeds the device range, the last character will be seen as the ending character.

API	Instruction code			Operand	Function
2109		$FSTR	P	S_1 , S_2 , **D**	Converting the floating-point number into the string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●							○
S_2	●	●			●	●		●	●							
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Source value

S_2 : Initial device in which the format is stored

D : Initial device in which the conversion result is stored

Explanation:

6

1. The floating-point number in S_1 is converted into the string in accordance with the setting of S_2, and the conversion result is stored in **D**.

2. The floating-point number in S_1 is converted into the string and added with the ending character 16#00 and then store the conversion result in **D**.

(S_1+1 , S_1)

Floating-point number

	High byte	Low byte
D	Second ASCII code	First ASCII code
D + 1	Fourth ASCII code	Third ASCII code
D + 2	Sixth ASCII code	Fifth ASCII code
……	……	
D +n-1	m^{th} ASCII code	$m-1^{th}$ ASCII code
D +n		16#00 ending character

Floating-point string

3. The conversion result varies with the setting of S_2.

4. The value in S_2+1 should be within the range of 2≦S_2+1≦20; the signs (+,-), digits in the integer and decimal

number can be included but the point in the decimal number and the exponent number are not included. .

Operand	Description
S_2	0: Decimal format 1: Exponential
S_2+1	Number of character

5. If the value in **S_1** is a positive number, the first ASCII code can be inputted in D; if the value in **S_1** is a negative number, the sign 16#2D (-) will be inputted first and then the second ASCII can be inputted.

6. Decimal format (**S_2**=0)

After conversion, the floating-point string in the device D.

- The value in S2+1 should be within the range of 2≦S2+1≦20; the signs (+,-), digits in the integer and decimal number can be included but the point in the decimal number and the exponent number are not included..

- Example 1:

  Suppose the number of characters is 8. When the floating-point numbers are -1.2345678 and 123456. The calculation is as follows.

D number	Floating-point number -1.2345678		Floating-point number 123456	
	High byte	Low byte	High byte	Low byte
D	16#31 (1)	16#2D (-)	16#32 (2)	16#31 (1)
D + 1	16#32 (2)	16#2E (.)	16#34 (4)	16#33 (3)
D + 2	16#34 (4)	16#33 (3)	16#36 (6)	16#35 (5)
D + 3	16#36 (6)	16#35 (5)	16#00 ending character	
D + 4	16#00 ending character	16#38 (8)		

After the conversion, if the floating-point number can be shown in **S_2**+1 but the length exceed the value in **S_2**+1, the digits in decimal will be rounded off. If the floating-point string cannot fulfill all the number of characters, it is not required to fulfill.

- Example 2:

After the conversion, if the floating-point number can be shown in **S_2**+1, the instruction will use the exponential format to convert. For example the number of characters is 5 digits and the floating-point number is 1234567, the conversion result is1.2346E+06.

D number	Floating-point number 1234567	
D	16#2E (.)	16#31 (1)
D + 1	16#33 (3)	16#32 (2)
D + 2	16#36 (6)	16#34 (4)
D + 3	16#2B (+)	16#45 (E)
D + 4	16#32 (6)	16#30 (0)
D + 5	16#0000 ending character	

- Example 3:

After the conversion, if the floating-point number can be shown in **S_2**+1, the instruction will use the exponential format to convert. For example the number of characters is 2 digits and the floating-point number is 0.00012345, the conversion result is1.2E-04.

D number	Floating-point number 0.00012345	
D	16#2E (.)	16#31 (1)
D + 1	16#45 (E)	16#32 (2)
D + 2	16#30 (0)	16#2D (-)
D + 3	16#00 ending character	16#34 (4)

- Example 4:

After the conversion, if the absolute value of the floating-point number is <=10^{-5}, the instruction will use the exponential format to convert. For example the number of characters is 4 digits and the floating-point number is 0.00001234, the conversion result is1.234E-05.

D number	Floating-point number 0.00001234	
D	16#2E (.)	16#31 (1)
D + 1	16#33 (3)	16#32 (2)
D + 2	16#45 (E)	16#34 (4)
D + 3	16#30 (0)	16#2D (-)
D + 4	16#00 ending character	16#35 (5)

7. Exponential format (**S_2**=1)

After conversion,
the floating-point string
in the device D.

- The value in S2+1 should be within the range of 2≦S2+1≦20; the signs (+,-), digits in the integer and decimal number can be included but the point in the decimal number and the exponent number are not included. After calculation, the length will add the exponents (4 digits) and the point of the decimal number in.
- The number of character in the area for the integer is 1 digit.
- The number of character in the area for the exponent is 4 words.
  If the exponent is a positive number, 16#2B (+) will be added in the area for exponent in D. If the exponent is a negative number, 16#2D (-) will be added in the area for exponent in D. The number of character in the area for the exponent is 2 digits. If there is only 1 digit in the conversion result, the first digit of the area for the exponent will be added 16#30 (0).
- Example:

  Suppose the number of characters is 8. When the floating-point numbers are -123.456789 and 123456. The calculation is as follows.

D number	Floating-point number -123.456789		Floating-point number 12345	
	High byte	Low byte	High byte	Low byte
D	16#31 (1)	16#2D (-)	16#2E (.)	16#31 (1)
D + 1	16#32 (2)	16#2E (.)	16#33 (3)	16#32 (2)
D + 2	16#34 (4)	16#33 (3)	16#35 (5)	16#34 (4)

D + 3	16#36 (6)	16#35 (5)	16#2B (+)	16#45 (E)
D + 4	16#45 (E)	16#38 (8)	16#34 (4)	16#30 (0)
D + 5	16#30 (0)	16#2B (+)	16#00 ending character	
D + 6	16#00 ending character	16#32 (2)		

After the conversion, if the floating-point number can be shown in $\mathbf{S_2}$+1, the exceeding digits will be rounded off.

The conversion result of the string length for the floating number -123.456789 in a number of 8 characters is 13 (the ending character excluded).

The conversion result of the string length for the floating number 12345 in a number of 8 characters is 10 (the ending character excluded).

If the floating-point string cannot fulfill all the number of characters, it is not required to fulfill.

Example 1:

The floating-point number in (D1, D0= 12.3456) is converted into the decimal format of the string (D4=0, D5=8).

D10	16#32 (2)	16#31 (1)
D11	16#33 (3)	16#2E (.)
D12	16#35 (5)	16#34 (4)
D13	16#00 ending character	16#36 (6)

Example 2:

The floating-point number in (D1, D0 = 0.0012345678) is converted into the exponential format of the string (D4=1, D5=8).

D10	16#2E (.)	16#31 (1)
D11	16#33 (3)	16#30 (2)
D12	16#35 (5)	16#34 (4)
D13	16#37 (7)	16#36 (6)
D14	16#45 (E)	16#38 (8)
D15	16#30 (0)	16#2D (-)
D16	16#00 ending character	16#33 (3)

Additional remark:

1. If the value in $\mathbf{S_1}$ exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

2. If the value in $\mathbf{S_2}$ is neither 0 nor 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in $\mathbf{S_2}$+1 exceeds the range $2 \leqq \mathbf{S_2}+1 \leqq 20$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If users declare the operand $\mathbf{S_2}$ in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

API	Instruction code			Operand	Function
2110		$FVAL	P	**S · D**	Converting the string into the floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●						○	
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Source value

D : Device in which the conversion result is stored

Explanation:

1. The string in **S** is converted into the floating-point number, and the conversion result is stored in **D**.

	High byte	Low byte
S	Second ASCII code	First ASCII code
S + 1	Fourth ASCII code	Third ASCII code
S + 2	Sixth ASCII code	Fifth ASCII code
……	……	
S +n-1	mth ASCII code	m-1th ASCII code
S +n		16#00 ending character

Floating-point string

⇨

(**D+1 · D**)

Floating-point number

- Refer to the following sections for the ASCII code usage.
- For the decimal or exponential formats, the maximum length for the floating-point string (m) is 24 words (16#00 the ending character excluded) and as for n, the maximum length is 13 words.

2. The format of the value in **S** can be decimal or exponential. The PLC will determine the format according to the value in **S** automatically.

- Decimal format: the length for the floating-point string is 9; the ending character 16#00 is not included.

	High byte	Low byte		
S	16#31 (1)	16#32 (2)		
S + 1	16#32 (2)	16#2E (.)		**D**
S + 2	16#34 (4)	16#33 (3)	⇨	21.234567
S + 3	16#36 (6)	16#35 (5)		
S + 4	16#00 ending character	16#37 (7)		

OR

	High byte	Low byte		
S	16#31 (1)	16#2D (-)		
S + 1	16#32 (2)	16#2E (.)		**D**
S + 2	16#34 (4)	16#33 (3)	⇨	-1.234568
S + 3	16#36 (6)	16#35 (5)		
S + 4	16#00 ending character	16#38 (8)		

- Exponential format: the length for the floating-point string is 9; the ending character 16#00 is not included.

	High byte	Low byte		
S	16#31 (1)	16#2D (-)		
S + 1	16#32 (2)	16#2E (.)		**D**
S + 2	16#45 (E)	16#33 (3)	⇨	-1230
S + 3	16#30 (0)	16#2B (+)		
S + 4	16#00 ending character	16#32 (3)		

3. If the sign code in **S** is 16#20, 16#30, or 16#2B the conversion result is a positive value. If the sign code in $\mathbf{S_1}$ is 16#2D, the conversion result is a negative value.

Example 1:

D0	16#31 (1)	16#32 (2)
D1	16#32 (2)	16#2E (.)
D2	16#34 (4)	16#33 (3)
D3	16#36 (6)	16#35 (5)
D4	16#00 ending character	16#37 (7)

⇨ (D11 · D10) → 21.234567

Example 2:

D0	16#31 (1)	16#2D (-)
D1	16#32 (2)	16#2E (.)
D2	16#45 (E)	16#33 (3)
D3	16#30 (0)	16#2B (+)
D4	16#00 ending character	16#32 (3)

⇨ (D101 · D100) → -1230

Additional remark:

1. If the length of the string in **S** exceeds 25 bytes and does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#2003.

2. There are some rules for the value in S. if the value in S is not followed by the rules, the instruction will not be executed, SM0 is ON and the error code in SR0 is 16#2003.
   The first ASCII code: signs16#2B(+), 16#2D(-), blank16#20, and numbers 16#30(0)~16#39(9) are allowed.
   If the first ASCII code is a sign or a blank, the second ASCII code should be a number.
   The second ASCII code can be in a format of decimal or exponential.

- Decimal format:

Integer	OR	Integer	.	Decimal

The point“,”(16#2E), can only be inputted once and there should be numbers before and after the point.

- Exponential format:

Integer	Exponent	OR	Integer	.	Decimal	Exponent

The point“,”(16#2E), can only be inputted once and there should be numbers before and after the point.

There should be a number before the exponent.

Integer: Only numbers “0” (16#30)~“9”(16#39) are allowed.

Decimal: Only numbers “0” (16#30)~“9”(16#39) are allowed.

Exponent: The format for a 4 digit ASCII code is as below.

E OR e	+ OR-	First number	Second number

There should be an “E” (16#45) or ”e” (16#65) and can only be used once.

There should be a sign “+”(16#2B) or “-” (16#2D) and can only be used once.

There should be 2 digits; numbers “0” (16#30)~“9”(16#39).

3. If the string in **S** is out of range, the instruction is not executed, SM is ON, and the error code in SR0 is

- If the first character in the string is a number ”0”~”9” (16#30~16#39), the range for a floating-point string is 1~24. The minimum length for the string is 1.
- If the first character in the string is a blank (16#20) or a sign (“+”(16#2B) or “-” (16#2D)), the range for a floating-point string is 2~24. The minimum length for the string is “+1”.

4. If the conversion result exceeds the range of values which can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
2111		$RIGHT	P	**S · n · D**	The retrieve of the characters in the string begins from the right.

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●			●	●		●	●						○	
n	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : String

n : Number of characters which are retrieved

D : Device in which the characters retrieved are stored

Explanation:

1. The instruction is used to retrieve **n** characters in the string in **S** from the right, and the characters which are retrieved are stored in **D**. When **S** is a string device, the maximum length for the value in **S** is 31 words; when the **S** is not a string device, the maximum length for the value in **S** is 255 words.

2. If **n** is 0, the value in **D** is 0. The maximum length for n is 255 words. .

If the data in **S** is ABCDEF12345 and **n** is 5, five characters in the string in **S** are retrieved from the right. The conversion result is as follows.

Example:

Additional remark:

1. If the operand S is not a string ($) but a device with a string, the string in **S** can read up to 256 words (16#00 ending character included). If the string in **S** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If **n** is less than 0, it will be seen as 0. If n is greater than the length of the string in **S**, it will be seen as the length of the string is **S**.

3. If **D** is not sufficient to contain **n** characters, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2112		$LEFT	P	**S · n · D**	The retrieve of the characters in the string begins from the left.

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●						○	
n	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : String

n : Number of characters which are retrieved

D : Device in which the characters retrieved are stored

Explanation:

6

1. The instruction is used to retrieve **n** characters in the string in **S** from the left, and the characters which are retrieved are stored in **D**. When **S** is a string device, the maximum length for the value in **S** is 31 words; when the **S** is not a string device, the maximum length for the value in **S** is 255 words.

2. If **n** is 0, the value in **D** is 0.

If the data in **S** is ABCDEF12345 and **n** is 7, seven characters in the string in **S** are retrieved from the left. The conversion result is as follows.

Example:

When M0 is ON, the instruction $LEFT is executed. The six characters starting from the character in D100 are retrieved, and stored in D10.

Additional remark:

1. If the operand S is not a string ($) but a device with a string, the string in **S** can read up to 256 words (16#00 ending character included). If the string in **S** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If **n** is less than 0, it will be seen as 0. If n is greater than the length of the string in **S**, it will be seen as the length of the string is **S**.

3. If **D** is not sufficient to contain **n** characters, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2113		$MIDR	P	S_1 , S_2 , **D**	Retrieving a part of the string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1	●	●			●	●		●	●						○	
S_2	●	●			●	●		●	●		○					
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							●
S_2		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : String

S_2 : Part of the string which is retrieved

D : Device in which the characters retrieved are stored

Explanation:

1. Suppose the values in S_2 and S_2+1 are n and m respectively. The m characters starting from the nth character in the string in S_1 are retrieved, and stored in **D**.

2. If the data in S_1 is ABCDEFGHIJK, the value in S_2 is 3, and the value in S_2+1 is 7, the seven characters starting from the third characters in the string are retrieved from the left. The conversion result is as follows.

3. If the value of S_2 is S_2<=0, S_2+1<-1or S_2+1=0, the instruction is not executed.

4. If the value in S_2+1 is -1, the characters in S_1 starting from the character indicated by the value in S_2 to the last character in S_1 are retrieved.

5. If the data in S_1 is ABCDEFGHIJK, the value in S_2 is 5, and the value in S_2+1 is -1, the conversion result is as follows.

Example:

D20	3
D21	4

Additional remark:

1. If the operand S_1 is not a string ($) but a device with a string, the string in S_1 can read up to 256 words (16#00 ending character included). If the string in S_1 does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If the value of S_2 is S_2<=0 or S_2+1<-1, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in $\mathbf{S_2}$ or $\mathbf{S_2}$ + $\mathbf{S_2}$+1 is larger than the length of the string in $\mathbf{S_1}$, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the value in $\mathbf{S_2}$+1 is larger than the number of characters which can be retrieved from the string in $\mathbf{S_1}$, SM0 is ON, and the error code in SR0 is 16#2003.

5. If **D** is not sufficient to contain $\mathbf{S_2}$+1 characters, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

6. If the operand $\mathbf{S_2}$ used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

API	Instruction code			Operand	Function
2115		$SER	P	S_1 · S_2 · **n** · **D**	Searching the string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●						○	
S_2	●	●			●	●		●	●						○	
n	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							●
S_2		●			●	●							●
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : String which is searched

S_2 : String which is searched for

n : n^{th} character in S_2 from which the search begins

D : Search result

Explanation:

1. The instruction is used to search the string from the nth character in S_2 for the string which is the same as the string in S_1, and the search result is stored in **D**.

2. When the nth character exceeds the string in S_2, or S_1>S_2, D is 0.

3. Suppose the string in S_2 is "ABCDEFGHIJK", the string in S_1 is "EFGH", and **n** is 3. The search begins from the third character in S_2, and the value in **D** is 5.

Example:

Additional remark:

1. If the operand $\mathbf{S_1}$ is not a string ($) but a device with a string, the string in $\mathbf{S_1}$ can read up to 256 words (16#00 ending character included). If the string in $\mathbf{S_1}$ does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If the operand $\mathbf{S_1}$ is not a string ($) but a device with a string, the string in $\mathbf{S_2}$ can read up to 256 words (16#00 ending character included). If the string in $\mathbf{S_1}$ does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

3. If **n** is less than or equal to 0, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2116		$RPLC	P	S_1、S_2、S_3、S_4、**D**	Replacing the characters in the string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●							
S_2	●	●			●	●		●	●						○	
S_3	●	●			●	●		●	●		○	○	○	○		
S_4	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : String which is replaced

S_2 : New string

S_3 : Number of characters in S_1 which are replaced

S_4 : The characters in S_1 starting from the character indicated by the value in S_4 are replaced.

D : Device in which the execution result is stored

Explanation:

1. The characters in S_1 starting from the character indicated by the value in S_4 are replaced by the characters in S_2, the number of characters which are replaced is indicated by the value in S_3, and the result is stored in **D**.

2. The four characters starting from the sixth character in the string "12345**6789**0" are replaced by "MN", and the result is "12345**MN**0".

3. If the string in **S_2** is 16#00, the instruction has the function of deleting the characters.

4. If the value in **S_3** is larger than the number of characters which can be replaced in the string in **S_1**, the characters in **S_1** starting from the character indicated by the value in **S_4** to the last character in **S_1** are replaced.

5. If the value in **S_3** is equal to 0, the instruction is not executed.

Example:

When M0 is ON, the data in D0~D7 is “1234ABAB1234AB”, and the data in D10~D11 is “CDEF”. When the instruction $RPLC is executed, the characters in D0~D7 starting from the character indicated by the value in D51 are replaced by the characters in D10~D11. The number of characters which are replaced is indicated by the value in D50, and the result is stored in D20~D27.

If the values in D50 and D51 are 3 and 4 respectively, the execution result is as follows.

If the values in D50 and D51 are 4 and 4 respectively, the execution result is as follows.

If the values in D50 and D51 are 20 and 4 respectively, the execution result is as follows.

If the values in D50, D51, and D10 are 3, 4, and 16#00 respectively, the execution result is as follows. The three characters in D0~D7 starting from the fourth character are deleted.

Additional remark:

1. If the string in $\mathbf{S_1}$ does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If the string in $\mathbf{S_2}$ does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

3. If the value in $\mathbf{S_3}$ <0 or $\mathbf{S_4}$<=0 or the value in $\mathbf{S_4}$ is larger than the length of the string in $\mathbf{S_1}$, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the replaced value in the string (16#00 the ending character included) in **S1** is larger than 256, the instruction will not be executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2117		$DEL	P	S_1 · S_2 · S_3 · **D**	Deleting the characters in the string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●							
S_2	●	●			●	●		●	●		○	○	○	○		
S_3	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : String

S_2 : Number of characters which are deleted

S_3 : The characters in S_1 starting from the character indicated by the value in S_3 are deleted.

D : Device in which the execution result is stored

Explanation:

1. The characters in S_1 starting from the character indicated by the value in S_3 are deleted, the number of characters which are deleted is indicated by the value in S_2, and the result is stored in **D**.

2. The four characters starting from the third character in the string "12**3456**7890" in S_1 are deleted, and the result "127890" is stored in **D**.

3. If the value in **S_2** is larger than the number of characters which can be deleted in the string in **S_1**, the characters in **S_1** starting from the character indicated by the value in **S_3** to the last character in **S_1** are deleted, and 16#00 is stored in **D**.

4. If the value in **S_2** is equal to 0, the instruction is not executed.

Example:

When M0 is ON, the data in D0~D3 is “1234567”. When the instruction $DEL is executed, the characters in D0~D3 starting from the character indicated by the value in D11 are deleted. The number of characters which are deleted is indicated by the value in D10, and the result is stored in

If the values in D10 and D11 are 3 and 4 respectively, the execution result is as follows.

If the values in D10 and D11 are 5 and 4 respectively, the execution result is as follows. Owing to the fact that the number of characters which are deleted exceeds the range, the characters in D0~D3 starting from the fourth character to the last character are deleted.

If the values in D10 and D11 are 5 and 1 respectively, the execution result is as follows.

Additional remark:

1. If the string in $\mathbf{S_1}$ does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If the value in $\mathbf{S_2}$ is less than 0, the value in $\mathbf{S_3}$ is less than or equal to 0, or the value in $\mathbf{S_3}$ is larger than the length of the string in $\mathbf{S_1}$, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2118		$CLR	P	**S**	Clearing the string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : String which is cleared

Explanation:

1. The string in **S** is cleared. If the string in **S** does not end with 16#00, up to 255 characters will be cleared.

	b15 … b8	b7 … b0
S	16#31(1)	16#2D(-)
S +1	16#33(3)	16#32(2)
S +2	16#34(4)	16#35(5)
S +3	16#36(6)	16#2E(.)
S +4	16#38(8)	16#37(7)
S +5	16#00	

After the instruction is executed

S	16#0000
S +1	16#0000
S +2	16#0000
S +3	16#0000
S +4	16#0000
S +5	16#0000

Example:

The string in D0 is cleared, as illustrated below.

	b15 … b8	b7 … b0
D0	16#32(2)	16#31(1)
D1	16#34(4)	16#33(3)
D2	16#36(6)	16#35(5)
D3	16#00	16#37(7)

After the instruction is executed

16#0000	D0
16#0000	D1
16#0000	D2
16#0000	D3

API	Instruction code			Operand	Function
2119		$INS	P	S_1 · S_2 · S_3 · **D**	Inserting the string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●							
S_2	●	●			●	●		●	●						○	
S_3	●	●			●	●		●	●		○	○	○	○		
D		●			●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : String

S_2 : String which is inserted

S_3 : The string is inserted into S_1 after the character indicated by the value in S_3.

D : Device in which the execution result is stored

Explanation:

1. The string in S_2 is inserted into the string in S_1 after the character indicated by the value in S_3, and the result is stored in **D**.

2. If the string in either S_1 or S_2 is a null string, the other string which is not a null string is stored in **D**.

3. If the strings in S_1 and S_2 are null strings, 16#0000 is stored in **D**.

Example:

When M0 is ON, the data in D0~D3 is “1234567”, and the data in D10 is “AB”. When the instruction $INS is executed, “AB” is inserted into the string in D0~D3 after the character indicated by the value in D30. The result is stored in D20~D24.

If the value in D30 is 1, the execution result is as follows.

If the value in D30 is 0, the execution result is as follows.

Additional remark:

1. If the string in $\mathbf{S_1}$ does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If the string in $\mathbf{S_2}$ does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

3. If the value in $\mathbf{S_3}$ is less than 0, or if the value in $\mathbf{S_3}$ is larger than the length of the string in $\mathbf{S_1}$, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the inserted value in the string (16#00 the ending character included) in **S1** is larger than 256, the instruction will not be executed, SM0 is ON, and the error code in SR0 is 16#2003.

6.22 Ethernet Instructions

6.22.1 List of Ethernet Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2200	SOPEN	–	✓	Opening the socket
2201	SSEND	–	✓	Sending the data through the socket
2203	SCLOSE	–	✓	Closing the socket
2204	MSEND	–	✓	Sending the email
2206	INTOA	–	✓	Converting the IP address of the integer type into the IP address of the string type
2207	IATON	–	✓	Converting the IP address of the string type into the IP address of the integer type
2208	EIPRW	–	–	Reading and writing the Ethernet/IP data
2209	SCONF	–	✓	Setting TCP/UDP Socket parameters
2210	MCONF	–	✓	Reading/Writing the Modbus TCP data

6.22.2 Explanation of Ethernet Instructions

API	Instruction code			Operand	Function
2200		SOPEN	P	S_1, S_2, S_3	Opening the socket

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●			○	○	○		
S_2	●	●			●	●		●	●			○	○	○		
S_3	●	●			●	●		●	●			○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Socket mode

S_2 : Socket number

S_3 : Start mode

Explanation:

1. S_1 is 1 if users want to open the TCP socket, and S_1 is 0 if users want to open the UDP socket. S_2 is the socket number. The AS series PLC as the client sends the TCP connection request to the server if S_3 is 1, and the AS series PLC as the sever waits for the TCP connection request from the client if S_3 is 0. If users want to start the UDP connection, S_3 can be 0 or 1.

2. The operand S_1 should be either 0 or 1; the operand S_2 should be within the range between 1 and 4; the operand S_3 should be either 0 or 1.

3. Before using the instruction, users have to accomplish the following setting in HWCONFIG of ISPSoft.

   - PLC Parameter Setting→Ethernet-Basic→Setting the IP addres and the netmask address
   - PLC Parameter Setting→Ethernet-Advance→Socket→Enable Socket Function
   - PLC Parameter Setting→Ethernet-Advance→Socket→TCP/UDP Socket Connection and Setting the sockets which will be used.

4. Refer to API2209 SCONF instruction for the modification of the TCP and UDP Socket parameters

5. When the TCP Socket is opened, the settings of the Socket IP and communication ports are shown as below.

Start Mode	Remote IP	Local communication port	Remote communication port	Description
1	Specific IP address	0	0	Illegal
1	Specific IP address	0	Not equal to 0	Master mode, Specifies the IP address; but not specify the local communication port.
1	Specific IP address	Not equal to 0	0	Illegal
1	Specific IP address	Not equal to 0	Not equal to 0	Master mode, Specifies the IP address, local communication port and remote communication port
1	0.0.0.0	No limit to the value	No limit to the value	Illegal
0	Specific IP address	0	No limit to the value	Illegal
0	Specific IP address	Not equal to 0	0	Slave mode, Specifies the IP address; but not specify the remote communication port.
0	Specific IP address	Not equal to 0	Not equal to 0	Slave mode, Specifies the IP address and remote communication port.
0	0.0.0.0	0	No limit to the value	Illegal
0	0.0.0.0	Not equal to 0	0	Slave mode, Not specify the IP address and remote communication port
0	0.0.0.0	Not equal to 0	Not equal to 0	Slave mode, Not specify the IP address, but specifies the remote communication port.

6. If the data are transmitted through the TCP socket, and no error occurs after the instruction is executed, the socket starts to establish the connection with the remote device and the flag of being starting the connection is ON. If the connection is made successfully, the flag of successful connection is ON, and the flag of being starting the connection is OFF. If an error occurs, the error flag is ON.

TCP Socket Number	Successful connection	Having received the data	Having sent the data	Being starting the connection	Being closing the connection	Being sending the data	Error flag
1	SM1270	SM1271	SM1272	SM1273	SM1274	SM1275	SM1277
2	SM1278	SM1279	SM1280	SM1281	SM1282	SM1283	SM1285
3	SM1286	SM1287	SM1288	SM1289	SM1290	SM1291	SM1293
4	SM1294	SM1295	SM1296	SM1297	SM1298	SM1299	SM1301

7. If the data are transmitted through the UDP socket, and no error occurs after the instruction is executed, the flag of having started the connection is ON. If an error occurs, the error flag is ON.

UDP Socket Number	Having started the connection	Having received the data	Having sent the data	Being closing the connection	Error flag
1	SM1334	SM1335	SM1336	SM1337	SM1338
2	SM1339	SM1340	SM1341	SM1342	SM1343
3	SM1344	SM1345	SM1346	SM1347	SM1348
4	SM1349	SM1350	SM1351	SM1352	SM1353

8. Generally, the pulse instruction SOPENP is used.

Example 1:

1. The system framework below illustrates how to establish the TCP connection between a computer as Server and an AS series PLC as Client.

2. ISPSoft -> HWCONFIG（Ethernet - Basic）

Ethernet configuration	
IP Addressing Mode	Static
IP Address	192.168.1.111
Netmask address	255.255.255.0
Gateway address	192.168.1.1

3. a）ISPSoft -> HWCONFIG（Ethernet – Advance > Socket > TCP Socket Connection）

   b）SCONF instruction

TCP Socket Connection	
Remote IP	192.168.1.110
Remote port	3201
Local port	3200
Send Data Address	D1000
Send Data Length	200
Receive Data Address	D2000
Receive Data Length	200
Keep Alive Timer	10

4. The parameters related to TCP Socket1 are set first in HWCONFIG of the ISPSoft configuration software or in the SCONF instruction.

5. When M0 is ON, whether the socket is closed, has been connected, or is being connected is checked. If the socket is not closed, has not been connected, or is not being connected, the connection procedure is performed. After the socket has been connected, M0 will is switched to OFF.

6. When M1 is ON, whether the socket has been connected and no data are being sent is checked. If the socket has been connected, and no data are being sent, the data will be sent. If the socket has not been connected, the instruction will not be executed. After the sending of the data is completed, M1 will be switched to OFF.

7. When M2 is ON, whether the socket has been connected and the flag of having received the data is ON is checked. If the socket has been connected and the flag of having received the data is ON, it means that the data have already been received.

8. When M3 is ON, whether the socket has been connected is checked. If the socket has been connected, the connection will be closed. If the socket has not been connected, the instruction will not be executed. After the connection is closed, M2 and M3 will be switched to OFF.

Network 1

Network 2

Network 3

Network 4

Network 5

Example 2:

1. The example illustrates how to establish the UDP connection between a computer and an AS series PLC.

2. When M0 is ON, whether the socket has been connected is checked. If the socket has not been connected, the connection procedure is performed. After the socket has been connected, M0 will be switched to OFF.

3. When M1 is ON, the data are sent. After the sending of the data is completed, M1 will be switched to OFF.

4. When M2 is ON, whether the socket has been connected and the flag of having received the data is ON is checked. If the socket has been connected and the flag of having received the data is ON, it means that the data have been received.

5. When M3 is ON, whether the socket has been connected is checked. If the socket has been connected, the connection will be closed. If the socket has not been connected, the instruction will not be executed. After the connection is closed, M2 and M3 will be switched to OFF.

Network 1

Network 2

Network 3

Network 4

Network 5

Additional remark:

1. Errors in TCP connection are explained below.

Error code	Error flag	Description
16#2003	SM0	**S_1**, **S_2**, or **S_3** exceeds the range.
16#600C	SM1109	The local socket has been used.
16#600D	SM1100	Ethernet network is not connected.
16#6200	Error flag of TCP Socket	Illegal IP address
16#6201	Error flag of TCP Socket	Illegal TCP Socket communication mode setting
16#6202	Error flag of TCP Socket	Illegal TCP Socket mode setting
16#6203	Error flag of TCP Socket	Illegal address for sending data
16#6204	Error flag of TCP Socket	The length of sent data exceeds the range
16#6205	Error flag of TCP Socket	The device where data are sent exceeds the range
16#6206	Error flag of TCP Socket	Illegal address for receiving data
16#6207	Error flag of TCP Socket	The length of received data exceeds the range
16#6208	Error flag of TCP Socket	The device for receiving data exceeds the range
16#6212	Error flag of TCP Socket	TCP Socket communication timeout
16#6213	Error flag of TCP Socket	The size of data actually received is larger than the set data.
16#6214	Error flag of TCP Socket	TCP Socket connection is rejected by the remote equipment
16#6215	Error flag of TCP Socket	TCP Socket has not been connected
16#6217	Error flag of TCP Socket	TCP Socket connection has been triggered.
16#6218	Error flag of TCP Socket	The sending of data through TCP Socket has been triggered.
16#621A	Error flag of TCP Socket	Disabling the TCP Socket connection has been triggered.

2. Errors in UDP connection are explained below.

Error code	Error flag	Description
16#2003	SM0	**S_1**, **S_2**, or **S_3** exceeds the range.
16#600C	SM1109	The local socket has been used.
16#600D	SM1100	Ethernet network is not connected.
16#6209	Error flag of UDP Socket	Illegal IP address
16#620A	Error flag of UDP Socket	Illegal communication mode
16#620C	Error flag of UDP Socket	Illegal address for sending data

Error code	Error flag	Description
16#620D	Error flag of UDP Socket	The length of sent data exceeds the range
16#620E	Error flag of UDP Socket	The device where data are sent exceeds the range
16#620F	Error flag of UDP Socket	Illegal address for receiving data
16#6210	Error flag of UDP Socket	The length of data actually received exceeds the range.
16#6211	Error flag of UDP Socket	The device where data are received exceeds the range.
16#6213	Error flag of UDP Socket	The size of data actually received is larger than the set data.
16#6215	Error flag of UDP Socket	UDP Socket is not connected
16#6217	Error flag of UDP Socket	UDP Socket connection has been triggered.
16#6218	Error flag of UDP Socket	The sending of UDP Socket data has been triggered.
16#621A	Error flag of UDP Socket	Disabling the UDP Socket connection has been triggered.

3. When the client and server are both AS series PLC and the timeout time settings are same, the server will cut the connection automatically if the timeout occurs in the server first. Thus the error flag in the client will not be ON. Whereas, if the timeout occurs in the client first, the error flag in the client will be ON and the connection will be cut.

API	Instruction code			Operand	Function
2201		SSEND	P	S_1, S_2	Sending the data through the socket

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●			○	○	○		
S_2	●	●			●	●		●	●			○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Socket mode

S_2 : Socket number

Explanation:

1. S_1 is 1 if users want to open the TCP socket, and S_1 is 0 if users want to open the UDP socket. S_2 is the socket number.

2. The operand S_1 should be either 0 or 1, and the operand S_2 should be within the range between 1 and 4.

3. Before using this instruction, users need to use the instruction SOPEN to open the socket. If the flag of successful connection (of TCP socket) is ON, or the flag of having started the connection (of UDP socket) is ON, this instruction can be used.

4. If the data are sent through the TCP socket, and no error occurs after the instruction is executed, the data are sent, and the flag of being sending the data is ON. If the data are sent successfully, the flag of having sent the data is ON, and the flag of being sending the data is OFF. If an error occurs, the error flag is ON.

TCP Socket number	Being sending the data	Having sent the data	Error flag
1	SM1275	SM1272	SM1277
2	SM1283	SM1280	SM1285
3	SM1291	SM1288	SM1293
4	SM1299	SM1296	SM1301

5. If the data are sent through the UDP socket, and no error occurs after the instruction is executed, the flag of having sent the data is ON. If an error occurs, the error flag is ON.

UDP Socket number	Having sent the data	Error flag
1	SM1336	SM1338
2	SM1341	SM1343
3	SM1346	SM1348
4	SM1351	SM1353

6. Generally, the pulse instruction SSENDP is used.

Example:

Refer to the example in SOPEN instruction.

Additional remark:

1. Errors in TCP connection are explained below.

Error code	Error flag	Description
16#2003	SM0	S_1 or S_2 exceeds the range.
16#600D	SM1100	Ethernet network has not been connected.
16#6202	Error flag of TCP Socket	Illegal TCP Socket mode setting
16#6203	Error flag of TCP Socket	Illegal address for sending data
16#6204	Error flag of TCP Socket	The length of sent data exceeds the range.
16#6205	Error flag of TCP Socket	The device where data are sent exceeds the range.
16#6212	Error flag of TCP Socket	TCP Socket connection timeout
16#6214	Error flag of TCP Socket	TCP Socket TCP Socket connection is rejected by the remote equipment.
16#6215	Error flag of TCP Socket	TCP Socket has not been connected.
16#6218	Error flag of TCP Socket	The sending of data has been triggered.

2. Errors in UDP connection are explained below.

Error code	Error flag	Description
16#2003	SM0	S_1 or S_2 exceeds the range.
16#600D	SM1100	Ethernet network has not been connected.
16#620A	Error flag of UDP Socket	Illegal UDP Socket communication mode setting

16#620C	Error flag of UDP Socket	Illegal address for sending data
16#620D	Error flag of UDP Socket	The length of sent data exceeds the range
16#620E	Error flag of UDP Socket	The device for sending data exceeds the range
16#6218	Error flag of UDP Socket	The sending of data has been triggered.

API	Instruction code			Operand	Function
2203		SCLOSE	P	S_1, S_2	Closing the socket

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●			○	○	○		
S_2	●	●			●	●		●	●			○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Socket mode

S_2 : Socket number

Explanation:

1. S_1 is 1 if users want to close the TCP socket, and S_1 is 0 if users want to close the UDP socket. S_2 is the socket number.
2. The operand S_1 should be either 0 or 1, and the operand S_2 should be within the range between 1 and 4.
3. Before closing the socket, users need to make sure that the socket has been connected. Otherwise, the instruction is not executed.
4. If the TCP socket is closed by the client, the server continues to be connected to the local communication port (the flag of successful connection is ON). If the TCP socket is closed is by the server, the server will not be connected to the local communication port after the execution of the instruction is completed. After the TCP socket is closed, the corresponding flags will be OFF.
5. After the instruction is executed to close the UDP socket, the corresponding flag is OFF.
6. If the TCP socket is closed, and no error occurs after the instruction is executed, the connection with the remote device starts being closed and the flag of being closing the connection is ON. If the connection is closed successfully, the flag of being closing the connection is OFF. If an error occurs, the error flag is ON.

TCP Socket number	Being closing the connection	Error flag
1	SM1274	SM1277
2	SM1282	SM1285
3	SM1290	SM1293
4	SM1298	SM1301

7. If the UDP socket is closed, and no error occurs after the instruction is executed, the flag of having started the connection is OFF. If an error occurs, the error flag is ON.

UDP Socket number	Error flag
1	SM1338
2	SM1343
3	SM1348
4	SM1353

8. Generally, the pulse instruction SCLOSEP is used.

Example:

Please refer to the example of the execution of SOPEN.

Additional remark:

1. Errors in TCP connection are explained below.

Error code	Error flag	Description
16#2003	SM0	**S_1** or **S_2** exceeds the range.
16#600D	SM1100	Ethernet network has not been connected.
16#6212	Error flag of TCP Socket	TCP Socket communication timeout
16#6214	Error flag of TCP Socket	TCP Socket connection is rejected by the remote device
16#621A	Error flag of TCP Socket	Disabling TCP Socket connection has been triggered.

2. Errors in UDP connection are explained below.

Error code	Error flag	Description
16#2003	SM0	**S_1** or **S_2** exceeds the range.
16#600D	SM1100	Ethernet network has not been connected.
16#621A	Error flag of UDP Socket	Disabling UDP Socket connection has been triggered.

API	Instruction code			Operand	Function
2204		MSEND	P	S_1, S_2, S_3, D	Sending the email

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●			○	○	○		
S_2	●	●			●	●		●	●							
S_3	●	●			●	●		●	●							
D	●	●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1 : Remote email number

S_2 : Email subject

S_3 : Email body

D : Completion of the instruction

Explanation:

1. Users can send an email by setting S_1, S_2, and S_3.

2. Before using the instruction, users have to accomplish the following setting in ISPSoft.

   - PLC Parameter Setting→Ethernet-Basic→Setting the IP addres and the netmask address
   - PLC Parameter Setting→Ethernet-Advance→Email→Setting the SMTP server, the port, the local email address, and the SMTP subject
   - PLC Parameter Setting→Ethernet-Advance→Email and Trigger Configuration→Setting the email address
   - If the account identification is required, PLC Parameter Setting→Ethernet-Advance→Email→Setting the user name and the password

3. The email is set as follows.

Operand	Description	Setting range
S_1	Remote email number	The value in **S_1** should be within the range between 1 and 15. Bit0~bit3 set in ISPSoft indicate the remote email addresses. (Users can set 4 email addresses in ISPSoft.) Bit0 corresponds to remote email 1; bit1 corresponds to remote email 2 and so on. If users want to send an email, they have to set the corresponding bit in ISPSoft.
S_2	Email subject	The size of the email subject can be up to 16 words.
S_3	Email body	The size of the email body can be up to 64 words.
D	Completion of the instruction	After the execution of the instruction is completed, the bit is ON. If the execution of the instruction is abnormal, the next bit is ON.

4. Generally, the pulse instruction MSENDP is used.

Example:

Suppose the value in D0 is 00010100. When X0.0 is ON, the email is sent to remote email number 3 and remote email number 5. After the communication with the SMTP sever is completed, M20 is ON. If no error occurs during the communication, M21 is OFF and Y0.0 is ON.

Additional remark:

1. For the length of the string in **S_2** or **S_3**, the system will capture the data with the end of 16#00. If the length of the string is larger than the maximum value (with16#00 as the end), the length of the string in S_2 or S_3 will be counted equal to the maximum value.

2. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

3. Please reserve one word between **S_2** and **S_3** for the interrupt character.

4. Errors in the execution of the instruction are explained as below.

Error code	Error flag	Description
16#2003	SM0	1. **D+1** exceeds the device range 2. **S_1**<1 or **S_1**>15
16#600D	SM1100	Ethernet network has not been connected.
16#6100	D+1	Communication conflicts.
16#6107	D+1	E-mail communication timeout
16#6108	D+1	An error in the password authentication of the SMTP server account
16#6111	D+1	An invalid remote email address

API	Instruction code			Operand	Function
2206		INTOA	P	**S**，**D**	Converting the IP address of the integer type into the IP address of the string type

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S								●	●							
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S ： Source value

D ： Conversion result

Explanation:

1. The IP address in **S** and **S**+1 is converted into the IP address of the string type, and the conversion result is stored in **D**.

2. The operand **D** occupies eight consecutive devices.

Example:

D0	16#C0(192)	16#A8(168)
D1	16#00 (0)	16#01 (1)

After the instruction is executed

D10	"9"	"1"
D11	"."	"2"
D12	"6"	"1"
D13	"."	"8"
D14	"0"	"0"
D15	"."	"0"
D16	"0"	"0"
D17	16#00	"1"

"192.168.0.1"

Additional remark:

1. If users declare the operand **S** in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

2. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [8] of WORD/INT.

API	Instruction code			Operand	Function
2207		IATON	P	**S**，**D**	Converting the IP address of the string type into the IP address of the integer type

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S								●	●						○	
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S ： Source value

D ： Conversion result

Explanation:

1. The IP address of the string type in **S** is converted into the IP address of the integer type, and the conversion result is stored in **D** and **D**+1.

2. The operand **S** occupies eight consecutive devices.

"192.168.0.1"

S+0	"9"	"1"
S+1	"."	"2"
S+2	"6"	"1"
S+3	"."	"8"
S+4	"0"	"0"
S+5	"."	"0"
S+6	"0"	"0"
S+7	16#00	"1"

After the instruction is executed ⇨

D+0	16#C0(192)	16#A8(168)
D+1	16#00 (0)	16#01 (01)

3. There are 1~3 characters in every section of the IP address of the string type in **S**. These sections are separated by “.” (16#2E). For example, users can enter “192.168.0.1” instead of “192.168.000.001”

4. The value converted from the characters in every section of the IP address of the string type in **S** should be within the range of 0~255.

Example 1:

"192.168.0.1"

D10	"9"	"1"
D11	"."	"2"
D12	"6"	"1"
D13	"."	"8"
D14	"0"	"0"
D15	"."	"0"
D16	"0"	"0"
D17	16#00	"1"

After the instruction is executed ⇨

D0	16#C0(192)	16#A8(168)
D1	16#00 (0)	16#01 (1)

Example 2:

"192.168.0.1"

After the instruction is executed ⇨

D0	16#C0(192)	16#A8(168)
D1	16#00(0)	16#01(1)

Additional remark:

1. If the string in **S** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. In the string in **S**, except for the code representing the decimal point, the other binary codes have to be within the range between 16#30 and 16#39. If the other binary codes are not within the range between 16#30 and 16#39, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the number of decimal points "." in the string in **S** is not equal to 3, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the value converted from the characters in any section of the IP address of the string type in **S** is out of the range of 0~255, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

5. The number of characters in any section of the IP address of the string type in **S** is within the range of 1~3. If the

number of characters in any section of the IP address of the string type in **S** is larger than 3, the instruction is not executed, SM0 is ON and the error code in SR0 is 16#2003.

6. If users declare the operand **S** in ISPSoft, the data type will be ARRAY [8] of WORD/INT.

7. If users declare the operand **D** in ISPSoft, the data type will be ARRAY [2] of WORD/INT.

API	Instruction code			Operand	Function
2208		EIPRW		S_1~S_7, n, S, D_1, D_2	Reading and writing the Ethernet/IP data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●							
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6	●	●	●	●												
S_7								●	●				○	○		
n								●	●				○	○		
S								●								
D_1								●								
D_2								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1~D_2	Refer to Explanation in the instruction.												

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1~S_7, n, S : See Explanation below.

D_1~D_2 : See Explanation below.

6

Explanation:

1. The name and explanation of **S_1~S_7, n, S** and **D_1~D_2** are listed in the following table.

Operand	Name	Description	Data type	Remark
S_1	IP address	The first two sections of the IP address occupy the first word and the remaining two sections of the IP address occupy the second word. For example: If the IP address is 192.168.1.5, **S_1**=16#C0A8 and **S_1**+1=16#0105.	WORD[2]	Occupies two consecutive words
S_2	Connection mode	0: UCMM	WORD	
S_3	Function code (Service Code)	Range: 16#0000~00FF. If the function code exceeds the range, the instruction will not be executed.	WORD	
S_4	Class ID	Refer to Ethernet/IP protocol.	WORD	
S_5	Instance ID	Refer to Ethernet/IP protocol.	WORD	
S_6	Attribute ID switch	ON: Enable; OFF: Disable	BOOL	
S_7	Attribute ID	Refer to Ethernet/IP protocol.	WORD	
n	The length of read and written data	The size of the data which are written or read. Unit: byte; maximum: 200 bytes.	WORD	
S	The register involved with the read/ written data	The register where the sent data comes from or the register where the received data are stored	WORD[**n**]	
D_1	Communication status	0：Communication is not triggered to execute. 1：Communication is being performed. 2：Communication is completed and no error occurs. 3：A communication error occurs. 4：An error occurs in the parameter setting.	WORD	
D_2	Error code	The major error code and extended error code	WORD[2]	Occupies two consecutive words

2. When the instruction is enabled for the first time, it indicates that the communication command is sent. If the number of slaves connected reaches the upper limit, the communication status value in $\mathbf{D_1}$ is 0, which means the communication is not triggered to execute.

3. When the set parameter value exceeds the range, the instruction is not executed and the communication status value in $\mathbf{D_1}$ is 4

Example 1:

The vendor code of the device (at 192.168.1.5) are read and the read data are stored in D100. The EtherNet/IP Object parameters are set as below.

(1) Class ID = 1

(2) Instance ID = 1

(3) Attribute ID = 1

Network 1

Network 2

Example 2:

The maximum frequency (01-00) of the AC motor drive (at 192.168.1.6) is set to 60.00Hz. The EtherNet/IP Object parameters are set as below.

(1) Class ID = 16#0300

(2) Instance ID = 16#0001

(3) Attribute ID = 16#0000

Network 1

Network 2

Additional remark:

1. If users declare the operand $\mathbf{S_1}$ in ISPSoft, the data type will be ARRAY [2] of WORD.

2. If users declare the operand **S** in ISPSoft, the data type will be ARRAY [**n**] of WORD of which n is the size of the read/written data.

3. If users declare the operand **D_2** in ISPSoft, the data type will be ARRAY [2] of WORD.

4. Errors in the execution of the instruction are explained as below.

Error code	Error flag	Description
16#2003	SM0 / **D_1**	1. The value in **S_1**, **S_2**, **S_3** or **S_6** exceeds the range. 2. **S + n** is out of the range of the device address.
16#200B	SM0	The operand **n** is out of the range.
16#600D	SM1100	Ethernet network is not connected.
16#6301	**D_1**	The connection with the remote device is broken.
16#6302	**D_1**	The remote device response timeout
16#6303	**D_1**	An Illegal IP address
16#6304	**D_1**	An error in the response command service code
16#6305	**D_1**	An error in the response command length
16#6306	**D_1**	Communication conflicts

5. The status code in D2 [0] is explained as below.

Status code	Description	How to deal with
16#00	Connection succeeds	
16#01	Connection error	Check if the EDS file of the slave is correct.
16#02	Devices connected are unavailable	1. Check if the number of devices which the master is connected to exceeds the limit. 2. Check if the number of devices which the slave is connected to exceeds the limit.
16#03	An error in the parameter setting.	Check if the read/written data in S are correct.
16#04	Path error	Check if the settings of Class ID (S_4), Instance ID (S_5) and Attribute ID (S_7) are proper.
16#05	The path to the destination does not exist.	Check if the settings of Class ID (S_4), Instance ID (S_5) and Attribute ID (S_7) are proper.
16#07	The connection is broken.	1. Check if the Ethernet port of the slave device is connected properly. 2. Check if the setting of Keep alive timer for the slave device is proper.

Status code	Description	How to deal with
16#08	Service code is not supported	Check if the set function code (S_3) is proper.
16#09	Wrong attribute value	Check if the set registers involved in reading/writing data and the contents are proper.
16#0A	An error in the attribute list	Check if the slave device object attribute allows PLC to perform the Get_Attribute_List and Set_Attribute_List function.
16#0B	Transmission conflicts	Check if the service setting is repeated.
16#0C	Object status conflicts	Check if the Owner IO connection has been established.
16#0D	Object has existed.	Check if the slave has supported the set object. The service does not need to execute if the set object has been supported.
16#0E	Attribute is not writable.	Check if the object attribute supports the write function.
16#0F	No privilege to perform the service code	Check if the slave device is allowed to perform the service code.
16#10	The device can not perform the service currently.	Check if the Owner IO connection has been established.
16#11	The size of response data is too large.	Check if the length of data in object attribute exceeds the limit (100 words).
16#12	The data access sequence error occurs when Tag is accessed.	Check if the data length and data type are right.
16#13	The transmitted data are too short.	Check if the length (n) of read/written data is right.
16#14	Attribute value is unsupported.	Check if the set Attribute ID switch (S_6) and Attribute ID (S_7) are correct.
16#15	The transmitted data are too much.	Check if the read/written data length (n) is right.
16#16	The object is inexistent.	Check if the set Class ID (S_4) is right.
16#17	The data access sequence has an error when Tag is accessed.	1. Check if the Ethernet network connection is correct. 2. Check if some packet is lost in the Ethernet communication.
16#18		Check if a slave device state error occurs.

Status code	Description	How to deal with
	Attribute value is not stored.	
16#19	Attribute value storage error	Check if a slave device hardware error occurs.
16#1A	Router error. The length of the request packet exceeds the limit.	Check if the read/written data length (n) exceeds the limit of the router.
16#1B	Router error. The length of the response packet exceeds the limit.	Check if the read/written data length (n) exceeds the limit of the router.
16#1F	User-defined object access error	Refer to the definition of the slave device error.
16#20	Illegal parameter value	Check if the read/written value in S is correct.

API	Instruction code			Operand	Function
2209		SCONF	P	S_1~S_{10}	Setting TCP/UDP Socket parameters

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6								●	●				○	○		
S_7								●	●				○	○		
S_8								●	●				○	○		
S_9								●	●				○	○		
S_{10}								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1~S_{10}	Refer to Explanation in the instruction.												

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1~S_{10} : See Explanation below

Explanation:

1. The name and explanation of S_1~S_{10} are listed in the following table.

Operand	Name	Description	Data type	Remark
S_1	TCP/UDP selection	0: TCP, 1: UDP	WORD	

Operand	Name	Description	Data type	Remark
S_2	Socket number	Range: 1~4	WORD	
S_3	Remote IP address	The first two sections of the IP address occupy the first word and the remaining two sections of the IP address occupy the second word. For example: If the IP address is 192.168.1.5, **S_3**=16#C0A8 and **S_3**+1=16#0105.	WORD[2]	Occupies two consecutive words
S_4	Remote port	Range: 0~65535; 0 indicates any port.	WORD	
S_5	Local port	Range: 0~65535; 0 indicates any port.	WORD	
S_6	The register where the sent data comes from	Specifies the number of a D device. Range: 0~29999. For example: the setting value 100 means **S_7** bytes of data are sent from the registers starting from D100 (from the low byte to high byte).	WORD	
S_7	The size of the sent data	Maximum: 200 Bytes.	WORD	
S_8	The register where the received data are stored.	Specifies the number of a D device. Range: 0~29999. For example: the setting value 200 means **S_9** bytes of data are received and stored to the registers starting from D200 (from the low byte to high byte).	WORD	
S_9	The size of the received data	Maximum: 200 Bytes	WORD	
S_{10}	Connection time	Range: 1~30000, unit: second.	WORD	Applicable to TCP mode only

2. The pulse instruction is recommended for the setting when the instruction is used.

3. The Socket parameters take the setting values in HWCONFIG by default. When the parameter values need be changed randomly during the communication, the instruction will be needed for the modification.

4. If the parameters are set when the socket number is in communication, the setting values will not be effective until the communication is completed. It is suggested that the setting should be made after making sure that the socket is not used.

5. If any one of the setting value of parameters is out of the range, the instruction will not be executed, SM0 is ON and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2210		MCONF	P	S_1~S_{11}	Reading/Writing the Modbus TCP data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●							
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6								●	●				○	○		
S_7			●					●								
S_8								●	●				○	○		
S_9								●	●				○	○		
S_{10}								●	●				○	○		
S_{11}			●					●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1~S_{11}	Refer to Explanation in the instruction.												

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S_1~S_{11} : See Explanation below

Explanation:

1. The names and explanations of S_1~S_{11} are listed in the following table.

Operand	Name	Description	Data type	Remark
S_1	Data Exchange Table No.	MODBUS TCP data exchange table number, range: 1~32	WORD	

Operand	Name	Description	Data type	Remark
S_2	Remote IP address	The first two sections of the IP address occupy the first word and the remaining two sections of the IP address occupy the second word. For example: If the IP address is 192.168.1.5, **S_2**=16#C0A8 and **S_2**+1=16#0105.	WORD[2]	Occupies two consecutive words
S_3	Station address of the remote slave	Range: 0~255. If the setting value exceeds the range, the value in the low byte will be used automatically.	WORD	
S_4	The Read function code	Range: 16#01, 02, 03, 04 and 17. If the function code is out of the range, the modification of received parameters including the address and data length is not performed.	WORD	
S_5	Reading the remote communication address	16#0000~16#FFFF	WORD	
S_6	Reading the data length	The function code for reading Bit device supports max. 256 bits The function code for reading Word device supports max.100 words	WORD	
S_7	The local register where the received data are stored	The function code for reading data in Bit device supports M device only The function code for reading data in Word device supports D device only.	BOOL WORD	
S_8	The Write function code	Range: 16#05, 06, 0F and 10. If the function code is out of the range, the modification of sent parameters including address and data length is not performed.	WORD	
S_9	Writing the remote communication address	Range:16#0000~16#FFFF	WORD	
S_{10}	Writing the data length	Function code for writing data in Bit device: supports max. 256 bits	WORD	

Operand	Name	Description	Data type	Remark
		Function code for writing data in Word device: supports max. 100 words		
S_{11}	The local register where the remote data are written	The function code for writing data in Bit device supports M device only The function code for writing data in Word device supports D device only.	BOOL WORD	

2. It is suggested to use the instruction as the pulse instruction.

3. The Modbus communication function codes in **S_4** and **S_8** are explained as follows.

   When AS series PLC reads the data in multiple bit devices (not discrete input devices), the function code sent is 1 (16#01).

   When AS series PLC reads the data in multiple bit devices (not discrete input devices), the function code sent is 2 (16#02).

   When AS series PLC reads the data in multiple word devices (not input registers), the function code sent is 3 (16#03).

   When AS series PLC reads the data in multiple word devices (not input registers), the function code sent is 4 (16#04).

   When AS series PLC writes the status in one bit device, the function code sent is 5 (16#05).

   When AS series PLC writes the data in one word device, the function code sent is 6 (16#06).

   When AS series PLC writes the status in multiple bit devices, the function code sent is 15 (16#0F).

   When AS series PLC writes the data in multiple word devices, the function code sent is 16 (16#10).

   When AS series PLC writes the data in multiple word devices, the function code sent is 23 (16#17).

   At present, AS series PLC only supports the function codes above.

4. The parameter values set in the instruction are valid only while PLC is running. When PLC is repowered after power off, it will take the setting values in the data exchange table by default. If some parameter values need be revised during the data exchange, the instruction will be used for the modification of the parameter values.

5. If the specified number is in communication, the newly set parameter values will be effective when the next communication is performed after the current communication is completed.

6. When any one of the parameter values is not within the valid range, the instruction will not be executed; SM0 is ON and the error code in SR0 is 16#200B.

7. AS series PLC does not support Modbus TCP communication instructions. If the PLC program is used for the control, please enable the “Program control” mode in the data exchange table first and use the SM number in the following table.

SM number	Attribute	Explanation of the Ethernet data exchange parameters
SM1167	R/W	The flag for enabling the data exchange
SM1168 ~ SM1199	R/W	The flag for enabling the data exchange connection 1~32
SM1200 ~ SM1231	R	The success flag of the data exchange connection 1~32
SM1232 ~ SM1263	R	The error flag of the data exchange connection 1~32

When the receiving of data are OK, the success flag will be ON. When an error occurs in receiving data, the error flag will be ON. The success flag and error flag of one connection number are not switched to ON simultaneously.

8. The (Read-only) SR used in the data exchange is explained as follows.

SR number	Explanation
SR1120 ~ SR1151	Respectively indicate the actual communication time of connection 1~32.
SR1152 ~ SR1183	Respectively indicate the communication error codes of connection 1~32.

The error codes in SR1152 ~ SR1183 are defined as below.

Error code	Explanation
16#00XX	Remote device response error
16#F000	Ethernet network is not connected.
16#F001	Remote device response timeout
16#F003	TCP connection timeout
16#F007	Response command error
16#F009	The connection with the remote device is disconnected.

Item	Code	Description
File format	0	Binary value Default value
		The file name extension is .dmd.
		The unit of the value is the word.
	1	The values are separated by a comma.
		The unit of the value is the word.
		The file name extension is .cvs.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
	2	The values are separated by a comma.
		The unit of the value is the double word.
		The file name extension is .cvs.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
	3	The values are separated by a tab.
		The unit of the value is the word.
		The file name extension is .txt.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
	4	The values are separated by a tab.
		The unit of the value is the double word.
		The file name extension is .txt.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
	5	The values are not separated by any mark.
		The unit of the value is the word.
		The file name extension is .txt.
		The ASCII codes are adopted.

Item	Code	Description
		The value which is stored is a hexadecimal value.
	6	The values are not separated by any mark.
		The unit of the value is the double word.
		The file name extension is .txt.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
Function option	0	Appending The data which is written into the memory card is added after the last value in the file. Default value
		If the file does not exist, it is created automatically.
	1	Overwriting The data which is written into the memory card replaces the values in the file starting from the value indicated by the value in **S_4**.
		If the file does not exist, it is created automatically.
Reserved	-	The values of bit8~bit15 are 0.

- **S**: The data source
- **S_1**: The length of the data which is written into the file

  If the value in **S_1** is 0, the data is not written into the file.

Item	Description
Value unit	If the file format is 0, 1, 3, or 5, the unit of the value is the word. If the file format is 2, 4, or 6, the unit of the value is the double word.
Parameter unit	Double word
Length of the data	The devices in which the data is stored can not exceed the device range, and the size of the data which is written into the file can not be more than four gigabytes. (Please refer to chapter 2 for more information about the devices.)

- **S_2**: The line advance

  The value in **S_2** should be within the range between 0 and 256.

- **S_3~S_3**+4: **S_3** occupies five devices. Nine characters at most constitute a file name, including 16#00. If the string does not end with 16#00, the error occurs. If the ending character is read, the reading of the

characters stops, and whether the file name is legal is checked. The characters which can be used to constitute a file name are A~Z, a~z, and 0~9. Besides, the file name extension depends on the file format. The file which is created is in the default folder. If the file name is “Test1”, the characters are written into the devices as follows.

D	'e'	'T'	ASCII code →	D	16#65	16#54
D+1	't'	's'		D+1	16#74	16#73
D+2	NUL	'1'		D+2	16#00	16#31

- The default folder path:

Model name	Folder path
AS332T-A	PLC CARD\AS300\UserProg
AS332P-A	
AS324MT-A	

- **S_4**: The value in the file which is overwritten is indicated by the value in **S_4**.

Item	Description
Value unit	If the file format is 0, 1, 3, or 5, the unit of the value is the word. If the file format is 2, 4, or 6, the unit of the value is the double word.
Parameter unit	The parameter unit is the double word.
Usage	If the function option is 0, **S_4** is not used.
	If the function option is 1, the data which is written into the memory card replaces the values in the file starting from the value indicated by the value in **S_4**.
	The value in **S_4** should indicate the value in the file. If the value in **S_4** is 0, the first value in the file is overwritten.

2. The related flags:

Flag	Description
SM450	If the memory card is in the CPU module, the flag is ON.
SM451	The write protection switch on the memory card ON: The memory card is write protected. OFF: The memory card is not write protected.
SM452	The data is being written from the PLC to the memory card, or the data is being read from the memory card into the PLC.

Flag	Description
SM453	If an error occurs during the operation of the memory card, the flag is ON. If the flag is ON, users have to reset it to OFF. The error code is stored in SR453.

3. The related error codes (SR453):

Error code	Description
16#005E	An error occurs when the memory card is initialized.
16#005F	The path is incorrect, or the file does not exist.
16#0060	The default folder cannot be created.
16#0061	The memory space is insufficient.
16#0062	The memory card is write protected.
16#0063	An error occurs when the data is written into the file.
16#0064	The data cannot be read from the memory card.
16#0065	The file is a read-only file.

4. If the format of the file into which the data is written is 0, the format of the file from which the data is read is 0. Otherwise, the data cannot be read, and SM453 is ON. The same applies to the other file formats.

Example:

SM450 is ON when the memory card is inserted into the CPU module; SM452 is ON when MWRIT is executed; SM452 is OFF when the execution of MWRIT is complete. MWRITP the pulse instruction cannot be used continuously. If executing this pulse instruction to write data into the memory card continuously, it may exceed its written limitation and may lead to memory card broken.

Operand	Setting value	Description
D0	16#0011	The file into which the data is written
		The file format: The values are separated by a comma. The unit of the value is the word. The file name extension is .cvs. The ASCII codes are adopted.
D1	-	The data which is written into the file

Operand	Setting value	Description
D10, D11	16#00000030	The size of the data which is written into the file is 48 words.
D300	16#000A	Ten values are written into every line.
D20	D20=16#6554 D21=16#7473 D22=16#0031	The file name is “Test1”.
D30、D31	16#00000000	The data which is written into the memory card replaces the values in the file starting from the first value.

Additional remark:

1. If the value in **C** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in $\mathbf{S_1}$ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in $\mathbf{S_2}$ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the value in $\mathbf{S_3}$ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2301		MREAD	P	**C · S · S_1 · S_2 · S_3 · D**	Reading the data from the memory card into the PLC

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
C								●	●				○	○		
S								●	●							
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
C		●											
S													●
S_1			●										
S_2		●											
S_3			●										
D		●											

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

C : Control parameter

S : File name

S_1 : Data address in the file

S_2 : Reserved

S_3 : Data length

D : Data destination

Explanation:

1. The description of the operands:

- **C**: The control parameter

Item	Code	Description
File format	0	Binary value The default value is 0.
		The file name extension is .dmd.
		The unit of the value is the word.
	1	The values are separated by a comma.
		The unit of the value is the word.
		The file name extension is .cvs.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
File format	2	The values are separated by a comma.
		The unit of the value is the double word.
		The file name extension is .cvs.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
	3 3	The values are separated by a tab.
		The unit of the value is the word.
		The file name extension is .txt.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
	4	The values are separated by a tab.
		The unit of the value is the double word.
		The file name extension is .txt.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
	5	The values are not separated by any mark.
		The unit of the value is the word.
		The file name extension is .txt.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
	6	The values are not separated by any mark.

Item	Code	Description
		The unit of the value is the double word.
		The file name extension is .txt.
		The ASCII codes are adopted.
		The value which is stored is a hexadecimal value.
Function option	0	The values in the file starting from the value indicated by the value in **S_1**. are read. The default value is 0.
	1	The number of values is stored in **D** and **D**+1.
		If the file format is 0, 1, 3, or 5, the unit of the value is the word. If the file format is 2, 4, or 6, the unit of the value is the double word.
Reserved	-	The values of bit8~bit15 are 0.

- **S~S**+4: **S** occupies five devices. Nine characters at most constitute a file name, including 16#00. If the string does not end with 16#00, the error occurs. If the ending character is read, the reading of the characters stops, and whether the file name is legal is checked. The characters which can be used to constitute a file name are A~Z, a~z, and 0~9. Besides, the file name extension depends on the file format. The file which is created is in the default folder. If the file name is “Test1”, the characters are written into the devices as follows.

- The default folder path:

Model name	Folder path
AS332T-A	PLC CARD\AS300\UserProg
AS332P-A	
AS324MT-A	

- **S_1**: The value in the file which is read is indicated by the value in **S_1**.

Item	Description
Value unit	If the file format is 0, 1, 3, or 5, the unit of the value is the word. If the file format is 2, 4, or 6, the unit of the value is the double word.
Parameter unit	The parameter unit is the double word.
Usage	The value in **S_1** should indicate the value in the file. If the value in **S_1** is 0, the first value in the file is read.

- **S_3**: The length of the data which is read from the file

  The devices in which the data is stored cannot exceed the device range. If the value in **S_3** is larger than the number of values in the file, the length of the data read from the file is the number of values in the file. The unit **S_3** is the double word.

- **D**: The initial device in which the data is stored.

2. The related flags:

Flag	Description
SM450	If the memory card is in the CPU module, the flag is ON.
SM451	The write protection switch on the memory card ON: The memory card is write protected. OFF: The memory card is not write protected.
SM452	The data is being written from the PLC to the memory card, or the data is being read from the memory card into the PLC.
SM453	If an error occurs during the operation of the memory card, the flag is ON. If the flag is ON, users have to reset it to OFF. The error code is stored in SR453.

3. The related error codes (SR453):

Error code	Description
16#005E	An error occurs when the memory card is initialized.
16#005F	The path is incorrect, or the file does not exist.
16#0060	The default folder cannot be created.
16#0061	The memory space is insufficient.
16#0062	The memory card is write protected.
16#0063	An error occurs when the data is written into the file.
16#0064	The data cannot be read from the memory card.

4. If the format of the file into which the data is written is 0, the format of the file from which the data is read is 0. Otherwise, the data cannot be read, and SM453 is ON. The same applies to the other file formats.

Example:

SM450 is ON when the memory card is inserted into the CPU module; SM452 is ON when MREAD is executed; SM452 is OFF when the execution of MREAD is complete.

Operand	Setting value	Description
D0	16#0011	The file from which the data is read
		The file format: The values are separated by a comma. The unit of the value is the word. The file name extension is .csv. The ASCII codes are adopted.
D300	D300=16#6554 D301=16#7473 D302=16#0031	The file name is “Test1”.
D10, D11	16#00000000	The values in the file starting from the first value are read.
D1	16#000A	Ten values are read from every line.
D30、D31	16#00000020	The size of the data which is read from the file is 32 words.
D20	-	The data which is read is stored in D20.

Additional remark:

1. If the value in **C** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in $\mathbf{S_2}$ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in $\mathbf{S_3}$ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the value in **D** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2302		MTWRIT	P	**C**、**S**、**S_1**、**S_2**、**S_3**	Writing the string into the memory card

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
C								●	●				○	○		
S								●	●							
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●							

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
C		●											
S		●											
S_1		●											
S_2		●											
S_3													●

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

C : Control parameter

S : Data source

S_1 : Data length

S_2 : Separation mark

S_3 : File name

Explanation:

1. The description of the operands:

- **C**: The control parameter

Parameter value	Description
0	If the file exists, the data which is written into the memory card is added after the last byte in the file.

(Appending)	If the file does not exist, it is created automatically.
1 (Overwriting)	If the file exists, the new data which is written into the memory card replaces the old data in the file. The size of the file is the size of the new data.
	If the file does not exist, it is created automatically.

- **S**: The data source

  If the string which is written into the file is “12345”, the characters are stored in the devices as follows. Owing to the fact that a byte is taken as the basic unit, the first character is stored in the low byte in D300, the second character is stored in the high byte in D300. The same applies to the other characters. “16#00” is stored in the high byte in D300+2, and indicates the end of the string.

S300		S300+1		S300+2	
byte 2	byte 1	byte 4	byte 3	byte 6	byte 5
16#32	16#31	16#34	16#33	16#00	16#35

- **S_1**: The length of the data which is written into the memory card

  A byte is taken as the basic unit. The devices in which the data is stored cannot exceed the device range, and the length of the data which is written into the memory card cannot be more than 255 bytes.

- **S_2**: The separation mark

  If the value in **S_1** is N, the value in **S_2** is written into the memory card as follows.

S_2Operand		**Description**
High byte	**Low byte**	
16#00	16#00 or not 16#00	The N-byte data is written into the file.
Not 16#00	16#00	The N+1-byte data is written into the file. The value in the high byte in **S_2** is the value in the N+1st byte.
Not 16#00	Not 16#00	The N+2-byte data is written into the file. The value in the high byte in **S_2** is the value in the seventh byte, and the value in low byte in **S_2** is the value in the N+2th byte.

- **S_3~S_3+4**: **S_3** occupies five devices. Nine characters at most constitute a file name, including 16#00. If the string does not end with 16#00, the error occurs. If the ending character is read, the reading of the characters stops, and whether the file name is legal is checked. The characters which can be used to constitute a file name are A~Z, a~z, and 0~9. Besides, the file name extension depends on the file format. The file which is created is in the default folder. If the file name is “Test1”, the characters are written into the devices as follows.

S3	'e'	'T'	ASCII代碼	S3	16#65	16#54
S3 +1	't'	's'	⇨	S3 +1	16#74	16#73
S3+2	NUL	'1'		S3+2	16#00	16#31

- The default folder path

Model name	Folder path
AS332T-A	PLC CARD\AS300\UserProg
AS332P-A	
AS324MT-A	

2. The related flags:

Flag	Description
SM450	If the memory card is in the CPU module, the flag is ON.
SM451	The write protection switch on the memory card ON: The memory card is write protected. OFF: The memory card is not write protected.
SM452	The data is being written from the PLC to the memory card, or the data is being read from the memory card into the PLC.
SM453	If an error occurs during the operation of the memory card, the flag is ON. If the flag is ON, users have to reset it to OFF. The error code is stored in SR453.

3. The related error codes (SR453):

Error code	Description
16#005E	An error occurs when the memory card is initialized.
16#005F	The path is incorrect, or the file does not exist.
16#0060	The default folder can not be created.
16#0061	The memory space is insufficient.
16#0062	The memory card is write protected.
16#0063	An error occurs when the data is written into the file.
16#0064	The data can not be read from the memory card.
16#0065	The file is a read-only file.

Example:

SM450 is ON when the memory card is inserted into the CPU module; SM452 is ON when MTWRIT is executed; SM452 is OFF when the execution of MTWRIT is complete.

Operand	Setting value	Description
D0	16#0001	The file into which the data is written
		The file format: The unit of the character is the byte. The file name extension is .txt. The ASCII codes are adopted. The data in D300 is written into the file.
D300	-	The data which is written into the file
D10	16#000A	The size of the string which is written into the file is 10 bytes.
D1	16#0A00	After the data is written into the file, the separation mark is added after the last byte in the file.
D30	D30=16#6554 D31=16#7473 D32=16#0031	The file name is “Test1”.

Additional remark:

1. If the value in **C** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in $\mathbf{S_1}$ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in $\mathbf{S_3}$ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2303		MEMW	P	**S · D · n**	Writing the data into the file register

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●								
D									●							
n	●	●			●	●		●				●	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●											
D		●											
n		●											

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : The initial address of the data source

D : The initial address in which the data is stored

n : Data length

Explanation:

1. **S**: The initial address of the data source; it is suggested to declare an array type of variables.
2. **D**: The initial address in which the data is stored; it is suggested to assign an address for the file register and declare an array type of variables.
3. **n**: The length of the data which is written into the file register, ranging from 1 to 2048. When exceeding, the instruction will not be executed, SM0 is ON, and the error code in SR0 is 16#200B.
4. If the device **S** or **D** exceeds the allowed range, SM0 is ON, and the error code in SR0 is 16#2003.
5. Since it takes 60~120ms for the instruction to write, it is suggested to use this instruction when the PLC is idle. For example there is no external interrupt task, no high-speed outputting, or any immediate events for the PLC to process.
6. The instruction will only write when the contact is from OFF to ON and will only write once.
   NOTE: This file register can only be written into for 100,000 times.

6.24 Task Control Instructions

6.24.1 List of Task Control Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2400	TKON	–	✓	Enabling the cyclic task
2401	TKOFF	–	✓	Disabling the cyclic task

6.24.2 Explanation of Task Control Instructions

API	Instruction code			Operand	Function
2400		TKON	P	**S**	Enabling the cyclic task

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●						●	●		○	○	○			

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Task number

Explanation:

1. The cyclic task specified by **S** is enabled.
2. When the PLC runs, the execution of the cyclic tasks depends on the setting of the cyclic tasks in ISPSoft.
3. The description of the operands:
   - The operand **S** should be within the range between 0 and 31.
   - Please refer to ISPSoft User Manual for more information about creating and enabling the tasks.

Example:

When the PLC runs, cyclic task (0) is enabled. Since the instruction TKON in cyclic task (0) is executed, cyclic task (1) is enabled, and Y0.0 is ON.

The two cyclic tasks are created in ISPSoft. Cyclic task (0) is enabled when the PLC runs, and cyclic task (1) is not enabled when the PLC runs.

Cyclic task (1) is enabled by the execution of the instruction TKON in cyclic task (0).

Cyclic task (1) is executed.

Additional remark:

Please refer to ISPSoft User Manual for more information related to tasks.

API	Instruction code			Operand	Function
2401		TKOFF	P	**S**	Disabling the cyclic task

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S	●	●						●	●		○	○	○			

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
AS	AS	-

Symbol:

S : Task number

Explanation:

1. The cyclic task specified by **S** is disabled.

2. When the PLC runs, the execution of the cyclic tasks depends on the setting of the cyclic tasks in ISPSoft.

3. The description of the operands:

   - The operand **S** should be within the range between 0 and 31.

   - Please refer to ISPSoft User Manual for more information about creating and enabling the tasks.

Example:

When the PLC runs, cyclic task (0) and cyclic task (1) are enabled. Since the instruction TKOOFF in cyclic task (0) is executed, cyclic task (1) is disabled, and Y0.0 is OFF.

The two cyclic tasks are created in ISPSoft. Cyclic task (0) and cyclic task (1) are enabled when the PLC runs, and cyclic task (1) is disabled when the instruction TKOFF in cyclic task (0) is executed.

Cyclic task (1) is disabled by the execution of the instruction TKOFF in cyclic task (0).

Cyclic task (1) is not executed.

Additional remark:

Please refer to ISPSoft User Manual for more information related to tasks.

6.25 SFC Instructions

6.25.1 List of SFC Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2500	SFCRUN	–	–	SFC Run
2501	SFCPSE	–	–	SFC Pause
2502	SFCSTP	–	–	SFC Stop

6.25.2 Explanation of Task Control Instructions

API	Instruction code	Operand	Function
2500	SFCRUN	S_1 · S_2 · S_3	SFC Run

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1																
S_2								●					○	○		
S_3																

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													
S_2		●			●	●							
S_3													

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Name of the SFC POU

S_2 : Function code

S_3 : Device address

Explanation:

1. The designated SFC program of S_1 will be activated according to the setups of S_2.

2. When the instruction is executed, the SFC POU designated by S_1 will be activated only when the SFC POU is being scanned.

3. Operand:

   - S_1 defines the name of the SFC POU.
   - When the designated SFC POU of S_1 is executed, the parameters such as SFC/STEP/ACTION/TRANSITION of the SFC program will be cleared when S_2=0 or 1, and the execution will start according to the value specified in S_2.
   - S_2=0, the system will execute the SFC POU from the initial Step.
   - S_2=1, the system will execute the SFC POU from the designated Step of S_3.

- **S_2**=2, the status and the parameters such as SFC/STEP/ACTION/TRANSITION of the SFC will NOT be cleared and the system will start executing from where it pauses.
- **S_3** designates the step to be started in the SFC program of **S_1**.

4. The range of **S_2** is 0 to 2. When it is out of range, it will be seen as 0.
5. When the state of the SFC POU is RUN, executing this instruction is invalid.

Example:

Set up one LD(ladder) POU and specify its POU name as Main, and 2 SFC POUs with the names of TestSFC1 and TestSFC2.

1. When the program is executed (RUN), TestSFC1 and TestSFC2 will execute the SFCSTP, and 2 SFC POUs will stop executing.
2. When M0 is set from OFF to ON, TestSFC1/ TestSFC2 POU will execute the SFCRUN* instructions. Refer to the contents of TestSFC1 and TestSFC2 for execution details of the 2 POUs. When **S_2** is set to 0, the status and the parameters of the SFC will be cleared and will begin to execute from STEP 1. When **S_2** is set to 1, the status and the parameters will be cleared and will begin to execute from the designated STEP of **S_3**.
3. When M1 is set from OFF to ON, TestSFC1 POU will pause. When **S_2** is set to 1, all the executing actions and the outputs of the SFC will be cleared, and the system will run the final scan.
4. When M2 is set from OFF to ON, TestSFC1 POU will execute its actions. When **S_2** is set to 1, the status and the parameters will be cleared, and the system will begin to execute from STEP 2.
5. When M3 is set from OFF to ON, TestSFC2 POU will pause. When **S_2** is set to 0, all the executing actions of the SFC and the outputs will be kept, and the system will not run the final scan.
6. When M4 is set from OFF to ON, TestSFC1 POU will execute its actions. When **S_2** is set to 2, the status and the parameters will be kept and will begin to execute from where it pauses.
6. *SFCRUN will activate SPC POU at the next scan.

Main POU

Network 1
SM402
SFCSTP
En
TestSFC1
S1
SFCSTP
En
TestSFC2
S1
Network 2
M0
SFCRUN
En
TestSFC1
S1
0
S2
Step1
S3
SFCRUN
En
TestSFC2
S1
0
S2
Step1
S3
Network 3
M1
SFCPSE
En
TestSFC1
S1
1
S2
Network 4
M2
SFCRUN
En
TestSFC1
S1
1
S2
Step2
S3
Network 5
M3
SFCPSE
En
TestSFC2
S1
0
S2
Network 6
M4
SFCRUN
En
TestSFC2
S1
2
S2
Step2
S3

TestSFC1 POU

TestSFC2 POU

Additional Remark:

Please refer to ISPSoft User Manual for more information related to SFC.

API	Instruction code			Operand	Function
2501		SFCPSE		S_1 , S_2	SFC Pause

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S_1																
S_2								●					○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													
S_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S_1 : Name of the SFC POU

S_2 : Function code

Explanation:

1. The designated SFC POU of S_1 will pause according to the setups of S_2.
2. When the instruction is executed, the SFC POU designated by S_1 will be paused only when the SFC POU is being scanned.
3. When pausing, the status and the parameters such as SFC/STEP/ACTION/TRANSITION of the SFC will be kept.
4. Operand:
   - S_1 defines the name of the SFC POU.
   - When S_2=0, all the executing actions of the SFC and the outputs will be kept, and the system will not run the final scan.
   - When S_2=1, all the executing actions and the outputs of the SFC POU will be cleared, and the system will run the final scan..
5. The range of S_2 is 0 to 1. When it is out of range, it will be seen as 0.
6. When the state of the SFC POU is PAUSE/STOP, executing this instruction is invalid.

Example:

Please refer to the SFCRUN programing example for more information.

Additional Remark:

Please refer to ISPSoft User Manual for more information related to SFC.

API	Instruction code			Operand	Function
2502		SFCSTP		**S**	SFC Stop

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S																

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													

Pulse instruction	16-bit instruction	32-bit instruction
-	AS	-

Symbol:

S : Name of the SFC POU

Explanation:

1. The designated SFC POU of **S** will stop.
2. When the instruction is executed, the SFC POU designated by **S_1** will stop only when the SFC POU is being scanned.
3. When stopping, the status and the parameters of the SFC will be cleared, and the system will run the final scan.
4. When the state of the SFC POU is STOP, executing this instruction is invalid.

Example:

Please refer to the SFCRUN programing example for more information.

Additional Remark:

Please refer to ISPSoft User Manual for more information related to SFC.

6.26 High-speed Output Instructions

6.26.1 List of High-speed Output Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2700	–	DPLSY	–	High-speed pulse output (without ramp-up/down process)
2701	–	DPLSR	–	High-speed pulse output (with ramp-up/down process)
2702	PWM	DPWM	–	Pulse width modulation
2703	JOG	DJOG	–	JOG output
2704	–	DZRN	–	Zero return
2705	–	DPLSV	–	Adjustable pulse output
2706	–	DDRVI	–	Relative position control
2707	–	DDRVA	–	Absolute position control
2708	CSFO	–	–	Catch speed and proportional output
2709	–	DDRVM	–	Mark alignment positioning
2710	–	DPPMR	–	2-Axis relative-coordinate point-to-point synchronized motion
2711	–	DPPMA	–	2-Axis absolute-coordinate point-to-point synchronized motion
2712	–	DCICR	–	2-Axis relative-position clockwise arc interpolation
2713	–	DCICA	–	2-Axis absolute-position clockwise arc interpolation
2714	–	DCICCR	–	2-Axis relative-position counterclockwise arc interpolation
2715	–	DCICCA	–	2-Axis absolute-position counterclockwise arc interpolation
2716	–	DCCMR	–	The relative-position circle drawing
2717	–	DCCMA	–	The absolute-position circle drawing
2718	TPO	–	–	The position planning table controls the output
2719	–	DTPWS	✓	Setting single-axis output parameters in the position planning table

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2720	–	DTPWL	✓	Setting linear interpolation parameters in the position planning table
2721	–	DTPWC	✓	Setting arc interpolation parameters in the position planning table
2723		DPPGB	–	Point to point go back and forth

6.26.2 Explanation of High-speed Output Instructions

API	Instruction			Operand	Description
2700	D	PLSY		S_1, S_2, D	High-speed pulse output (without ramp-up/down process)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
D		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Pulse output frequency

S_2 : Number of output pulses

D : Pulse output device

Explanation:

1. S_1 specifies the pulse output frequency as 4MHz for the line driver output models and the value within the range of 0Hz~200kHz for the open collector output models. The unit of the output frequency is 1Hz.The tolerable error rate for 200kHz is about 0.02% and for 100kHz is about 0.01%. The error rate is decreasing with the decreasing frequency. For example, if the output frequency is set to 199990Hz, the actual output is 199960Hz. If the output frequency is set to 99999Hz, the actual output is 99990Hz. If the output frequency is out of the valid range, PLC will take the maximum or minimum frequency for output automatically.

2. The output frequency specified by S_1 can change during the execution of the instruction without the execution of ramp up/down process. The time to change the frequency is when the being executed instruction is scanned and the output of a full pulse which the instruction is outputting is completed.

3. S_2 specified the number of output pulses. The range: 0~2,147,483,647. When the number of output pulses is set to 0, it means the number of pulses is not restricted and pulses can be output constantly till the instruction is disabled. When the number of pulses specified is less than 0, no pulse is output.

4. After the output is started, the number of output pulses specified by S_2 will not be allowed to change again.

5. **D** only specifies Y0.0~Y0.11 as the output point. After the instruction is enabled, its output function becomes the high-speed output. The basic instruction output point control will be invalid. It is suggested that the general output function should not be used after the high-speed output function is used in the program.

6. The ratio of Duty-OFF Time and Duty-ON Time for the pulse output is 1:1.

7. There is no limit to how many times the instruction is used and repeated use of the same output points. But every one output point can only be started and output by one high-speed output instruction in one scan cycle and the instruction which is started first will occupy the output point first.

8. After every high-speed output instruction for every output point is enabled, other instruction occupying the same output point can not be started till the high-speed output instruction which is being executed is disabled.

9. When the high-speed output instruction is enabled in the interrupt program or is not edited in the main process, it is suggested that the instruction should be used with the auto-reset function together when output is completed and PLC will update the output state in the END instruction.

10. After the stop flag is set, PLC will stop the output and clear the Busy flag only after the start instruction is executed a second time and outputs a full pulse. PLC will continue to outpout pulses when the stop flag is reset and the previously remaining pulse output is not finished. The stop flag is set and reset by designers.

11. The high-speed output points and corresponding SM/SR are listed in the following table.

Output point number	Attribute#2	Y0.0	Y0.1	Y0.2	Y0.3	Y0.4	Y0.5
Busy flag	R	SM460	SM472	SM480	SM492	SM500	SM512
Completion flag#3	R/W	SM461	SM473	SM481	SM493	SM501	SM513
Stop flag	R/W	SM463	SM474	SM483	SM494	SM503	SM514
Output completion auto-reset#4	R/W	SM470	SM475	SM490	SM495	SM510	SM515
Present output position#1 （32-bit）	R/W	SR460 SR461	SR474 SR475	SR480 SR481	SR494 SR495	SR500 SR501	SR514 SR515
Output point number	Attribute #2	Y0.6	Y0.7	Y0.8	Y0.9	Y0.10	Y0.11
Busy flag	R	SM520	SM532	SM540	SM552	SM560	SM572
Completion flag #3	R/W	SM521	SM533	SM541	SM553	SM561	SM573
Stop flag	R/W	SM523	SM534	SM543	SM554	SM563	SM574
Output completion auto-reset #4	R/W	SM530	SM535	SM550	SM555	SM570	SM575

Output point number	Attribute#2	Y0.0	Y0.1	Y0.2	Y0.3	Y0.4	Y0.5
Present output position #1 （32-bit）	R/W	SR520 SR521	SR534 SR535	SR540 SR541	SR554 SR555	SR560 SR561	SR574 SR575

Note:

#1: All present output positions are latched when power is off.

#2: R means “Ready-only” registers and the data in the registers can not be modified. R/W means the data can be read and written in the registers.

#3: It is suggested that the completion flag should be cleared by designers. If the completion flag is not cleared, it will be cleared automatically when the high-speed output instruction is enabled next time.

#4: The output completion auto-reset flag is set by designers. PLC will clear the flag automatically after the output is completed.

Example 1:

1. When X0.0 is ON, Y0.0 outputs 200 pulses with the frequency of 1kHz. SM461 is ON when the pulse output is completed and then Y0.12 is ON.
2. When X0.0 is OFF, Y0.0 stops the output. The pulse output is restarted when X0.0 is ON again.

Example 2:

External interrupt input X0.0

External interrupt input X0.1

Explanation:

1. Y0.0 outputs 1000 pulses whenever X0.0 receives the external interrupt signal once. Y0.2 outputs 1000 pulses whenever X0.1 receives the external interrupt signal once.

2. When the external interrupt input X triggers the pulse output from Y, the interval time between the Y pulse output completion and the next external interrupt input X trigger must be one PLC scan cycle or above.

Example 3: (ST program)

```
0001 IF M0 THEN
0002 DPLSY(1000,1000,Y0.0);
0003 ELSIF SM461 THEN
0004 SM470:=TRUE;
0005 END_IF;
0006
0007
```

Explanation:

1. When M0 is ON, Y0.0 outputs 1000 pulses with the frequency of 1kHz.

2. When the pulse output is completed, SM461 is ON and then SM470 is ON.

3. When M0 changes from OFF to ON, the pulse output will be restarted.

API	Instruction			Operand	Description
2701	D	PLSR		S_1, S_2, S_3, D	High-speed pulse output (with ramp-up/down process)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
D		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Target output frequency

S_2 : Number of output pulses

S_3 : Ramp-up/ down time

D : Pulse output device

Explanation:

1. S_1 specifies the pulse output frequency as 4MHz for the line driver output models and the value within the range of 0Hz~200kHz for the open collector output models. The unit of the output frequency is 1Hz.The tolerable error rate for 200kHz is about 0.02% and for 100kHz is about 0.01%. The error rate is decreasing with the decreasing frequency. For example, if the output frequency is set to 199990Hz, the actual output is 199960Hz. If the output frequency is set to 99999Hz, the actual output is 99990Hz. If the output frequency is out of the valid range, PLC will take the maximum or minimum frequency for the pulse output automatically.

2. After the instruction is enabled, the target output frequency specified by S_1 can be changed and the ramp up/down process is performed according to the set ramp up/down time. The time to change the frequency is when the being executed instruction is scanned and the output of a full pulse which is being output is completed.

3. S_2 is the number of output pulses. The range: 0~2,147,483,647. When the number of pulses is set to 0, it means the number of pulses is not restricted and pulses can be output continuously till the instruction is disabled.

4. After the output is started, the number of output pulses which S_2 specifies will not be allowed to change again.

5. **S_3** sets the ramp-up/down time with the unit of 1ms. The value is effective when the instruction is enabled for the first time. If the target frequency specified by **S_1** is modified in the ramp-up process, the ramp up/down time will be reloaded for execution. But if the target frequency is modified when the output enters the ramp-down process, the changing will be invalid.

6. **D** only specifies the output device among Y0.0~Y0.11. After the instruction is enabled, its output function becomes the high-speed output. The general instruction output point control will be invalid. It is suggested that the general output function should not be used after the high-speed output function is used in the program.

7. The target output frequency **S_1**, number of output pulses **S_2** and Ramp up/down time **S_3** are illustrated in the following figure.

8. The ratio of Duty-OFF Time and Duty-ON Time for the pulse output is 1:1.

9. There is no limit to how many times the instruction is used and the repeated use of the same output points. But every output point can be started for output only by one high-speed output instruction in one scan cycle and the instruction which starts the output point first will occupy the output point first.

10. After the high-speed output instruction for one output point is enabled, other instruction which is specified to occupy the same output point can not be started till the high-speed output instruction in execution is disabled.

11. When the high-speed output instruction is enabled in the interrupt program or is not edited in the main program, it is suggested that the instruction should be used with the auto-reset function together when the output is completed and PLC will update the output state in the END instruction.

12. After the stop flag is set, PLC will perform the ramp-down stop or immediately stop the output only after the start instruction is executed a second time and a full pulse is output.

13. Refer to PLSY instruction for explanation of the high-speed output points and corresponding SM/SR.

API	Instruction			Operand	Description
2702	D	PWM		S_1, S_2, D	Pulse width modulation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○		○	○		
S_2	●	●			●	●	●	●	●		○		○	○		
D		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	AS	AS

Symbol:

S_1 : Pulse output width

S_2 : Pulse output cycle

D : Pulse output device

6

Explanation:

1. The 16-bit instruction PWM takes 100us as the output unit and the 32-bit instruction DPWM takes 1us as the output unit.

Output instruction	PWM	DPWM
Range of pulse output width S_1	0 ~ 30000	0 ~ 60000
Range of pulse output cycle S_2	0 ~ 30000	0 ~ 60000

2. **S_1** the pulse output width (Duty ON) is defined as t, **S_2** the pulse output cycle time (Cycle time) is T as below. **$S_1 \leqq S_2$** is recommended.

3. **D** can only use Y0.0~Y0.11 as the pulse output device.

4. There is no limit to how many times the instruction is used in the program. But only one of the output instructions which use the same output point can be executed in the same scan cycle. When several high-speed pulse output instructions start the same point in the program, PLC will first perform the output based on the instruction which is executed first.

5. If $S_1 \leqq 0$ or $S_2 \leqq 0$, it will be seen as 0 output (the pulse output is OFF). If $S_1 > S_2$, it will be seen as $S_1 = S_2$. When $S_1 = S_2$ and S_2 is not 0, the pulse output will be always ON.

6. The pulse output width S_1 and pulse output cycle S_2 can be modified when PWM is executed.

7. Relevant special registers SR are listed in the following table.

Output point number	Attribute	Y0.0	Y0.1	Y0.2	Y0.3	Y0.4	Y0.5
Present output position#1 （32-bit）	R/W	SR460 SR461	SR474 SR475	SR480 SR481	SR494 SR495	SR500 SR501	SR514 SR515
Output point number	Attribute	Y0.6	Y0.7	Y0.8	Y0.9	Y0.10	Y0.11
Present output position #1 （32-bit）	R/W	SR520 SR521	SR534 SR535	SR540 SR541	SR554 SR555	SR560 SR561	SR574 SR575

#1：All present output positions are latched when power is off.

Example:

When X0.0 is ON, Y0.1 outputs the following pulses. When X0.0 is OFF, Y0.1 output changes to OFF.

API	Instruction			Operand	Description
2703	D	JOG		S_1, S_2, S_3, D_1, D_2	JOG output

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○		○	○		
S_2					●	●	●	●	●		○		○	○		
S_3					●	●	●	●	●		○		○	○		
D_1		○														
D_2		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	AS	AS

Symbol:

S_1 : Ramp-up time

S_2 : Target output frequency

S_3 : Ramp-down time

D_1 : Pulse output device

D_2 : Auxiliary output device

Explanation:

1. **S_1** is the ramp-up time with the unit of 10ms. For example: the setting value 10 means the ramp-up time is 100ms. After the instruction is enabled, the output frequency can be divided into ten times and it speeds up once every 10ms. And it will speed up to JOG target frequency specified by **S_2** when **S_1** the ramp-up time is reached.

2. When the instruction is disabled, its output frequency will ramp down once every 10ms based on the set ramp-down time. The output will not stop till the ramp-down time is reached. If the ramp-down time is set to 0, the output will stop immediately.

3. The range of ramp-up time **S_1** and ramp-down time **S_3** is 0~3000 (that is 0~30 seconds). If the setting value is out of the range, PLC will take the minimum or maximum value for the output. The ramp-up time and ramp-down time will

be affected by the scan time. If the accurate time is required to switch the ramp-up and ramp-down process, the output instruction with the ramp up/down designed such as DDRVI instruction is recommended to use.

4. The range of the target output frequency **S_2** is -200K~200kHz. If the setting value is out of the range, PLC will take the minimum or maximum value for the output. If the output frequency is a positive number, it indicates the forward ouput. If the output frequency is a negative number, it indicates the reverse output.

5. The output point for **D_1** must be selected from Y0.0~Y0.11 as the following table shows. For the auxiliary output point for **D_2**, the output points in the following table can be referred to. If users choose other output point or M device, the value in SR indicating the output mode will be invalid and the "Pulse+direction" mode will be valid by default. **D_2** is a direction output point here.

D_1 selects the even output point number.

Output point for **D_1**	Y0.0	Y0.2	Y0.4	Y0.6	Y0.8	Y0.10
Direction output point for **D_2**	Y0.1	Y0.3	Y0.5	Y0.7	Y0.9	Y0.11
Busy flag	SM460	SM480	SM500	SM520	SM540	SM560
Output mode	SR462	SR482	SR502	SR522	SR542	SR562
Present output position	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521	SR540 SR541	SR560 SR561

D_1 selects the odd output point number.

Output point for **D_1**	Y0.1	Y0.3	Y0.5	Y0.7	Y0.9	Y0.11
Direction output point for **D_2**	Y1.0 or above or any M deivice (Bool)					
Busy flag	SM472	SM492	SM512	SM532	SM552	SM572
Present output position	SR474 SR475	SR494 SR495	SR514 SR515	SR534 SR535	SR554 SR555	SR574 SR575

6. If SR=0, it indicates the "Pulse+direction" output mode. When SR=1, it means the A/B phase output. (Note: The output mode can be selected only when **D_1** uses the even output point and **D_2** uses the recommended direction output point.)

7. When the output begins, the Busy flag SM is set to ON. When the output is completed, the Busy flag will be reset to OFF automatically and the Completion flag will not be ON.

8. The output timing diagram: (For the Busy flag in the following figure, refer to the Busy flags of axes.)

9. The target output frequency can be modified during the execution of the instruction. When the new frequency is greater than the previous one, the ramp-up slope is used. Whereas, the ramp-down slope is used if the new frequency is less than the previous one.

10. The ramp-up/down slope is set through conversion of the set time and target frequency when the instruction is enabled. Thus the slope will not change with the changing target frequency in the process of the the instruction output. For example, the original target frequency for the output is 1kHz and then it is modified as 2kHz. Thus the actual ramp-down time will be different from the time set originally. The following dotted line is the ramp-up/down timing diagram after the target frequency is modified.

API	Instruction			Operand	Description
2704	D	ZRN		S_1, S_2, S_3, S_4, D	Zero return

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
S_4	○															
D		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4	●												
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Target frequency for zero return

S_2 : JOG frequency for DOG

S_3 : Zero return mode

S_4 : Input device for DOG

D : Pulse output device

Explanation:

1. The range of the target frequency for zero return **S_1** is 1Hz~200kHz. The JOG frequency **S_2** should be less than the target frequency **S_1**. The JOG frequency **S_2** is the start frequency. If **S_1** is less than **S_2**, **S_1** will be revised automatically into the one equal to **S_2**.

2. The input point for **S_4** and output point for **D** must match for use. Do not change them during the execution of the instruction. The input point for **S_4** can only be one of the 16 high-speed input points X0.0 ~ X0.15. For the selection of **D** output point and direction output point, refer to the following table. If **D** is not the preset "Pulse+direction" output (default: 0), change the mode into A/B phase output by setting SR to 1.

Axis number	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
Input point for **S_4**	It can be any one of the input points X0.0 ~ X0.15. But the same input point can not be selected for different axis output. If DOG point shakes or the switch bounces, please set the input point filter time in HWCONFIG.					
Output point for **D**	Y0.0	Y0.2	Y0.4	Y0.6	Y0.8	Y0.10
Direction output point	Y0.1	Y0.3	Y0.5	Y0.7	Y0.9	Y0.11
Output mode	SR462	SR482	SR502	SR522	SR542	SR562
Busy flag	SM460	SM480	SM500	SM520	SM540	SM560
Completion flag	SM461	SM481	SM501	SM521	SM541	SM561
Present output position	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521	SR540 SR541	SR560 SR561

3. **S_3** is for selection of a zero return mode and the function code is set by two high and low16-bit parameters. See the details as below.

Functions	Code		Explanation
	High 16-bit	Low 16-bit	
Leaves the zero point in the negative direction and then stops (Mode 0)	0	0	When the instruction is enabled, the search for the zero point is toward the negative direction with the target frquency. When the zero point is ON (the zero point signal changes from OFF to ON), the frequency is decreased to the JOG speed and the the motion in the negative direction will continue and will not stop till the zero point signal changes from ON to OFF.
Leaves the zero point in the positive direction and then stops (Mode 1)	0	1	When the instruction is enabled, the search for the zero point is toward the negative direction with the target frquency. When the zero point is ON (the zero point signal changes from OFF to ON), the frequency is decreased to 0 immediately and then the the motion is toward the positive direction at the JOG speed and will not stop till the zero point signal changes from ON to OFF.
Mode 0 Moves again after returning to the zero point	Number of pulses for motion	2	The operation of returing to the zero point is the same as that for the low 16-bit code. After the zero point is ON, the motion will continue according to the number of pulses specified. When the high 16-bit code is a positive number, the search is toward the positive direction. Whereas, the negative value means that the search is toward the negative direction.
Mode 0 Searches Z phase	Number	3	The motion of returing to the zero point is the

after returning to the zero point (Z phase input point is set in HWCONFIG)	of Z phases		same as that for the low 16-bit code. After the zero point is returned to, the motion will continue according to the number of Z phases. When the high 16-bit code is a positive number, the search is toward the positive direction. Whereas, the negtive value indicates that the search is toward the negative direction. Suppose that the rising-edge trigger of X0.1 is specified as the condition for Z phase input by HWCONFIG, the counting will be performed once whenever the rising-edge trigger of X0.1 occurs.
Mode 0 Outputs the clear signal after returning to the zero point. (Output clear point is set in HWCONFIG)	Number of pulses for moition or number of Z phases	4+0=4 4+1=5 4+2=6 4+3=7 (bit2=ON)	Choosing the value 4~7 means selecting the functions of code 0~3 respectively and the specified output point will send an ON signal which is about 20ms wide when the function execution is completed. The range of the output point is Y0.12~Y0.15 and Y1.0 ~ Y1.15. For example, if Y1.12 is specified as the output point in HWCONFIG, it indicates Y1.12 is for the output of clear signals.
Leaves the zero point in the positive direction and then stops (Mode 1)	0	8+0=8 8+1=9 (bit3=ON)	The operation for zero point return is the same as that for code 1 (Mode 1)
Mode 1 Outputs the number of pulses after the zero point is returned to	Number of pulses for moition	8+2=10 (bit3=ON)	The operation for zero point return is the same as that for low 16-bit code 1. After the zero point is returned to, the motion will continue in accordance with the number of pulses specified. When the value of the high 16-bit code is a positive number, the motion will go toward the positive direction. Whereas, the motion will be in the negative direction if the value is a negative number.
Mode 1 Searches for Z phase after returning to the zero point (Z phase input point is set in HWCONFIG)	Number of Z phases	8+3=11 (bit3=ON)	The operation for zero point return is the same as that for low 16-bit code 1. After the zero point is returned to, the motion will continue in accordance with the number of Z phases to seek. When the value of the high 16-bit code is a positive number, the motion will go toward the positive direction. Whereas, the motion will be in the negative direction if the value is a negative number. If the rising edge trigger of X0.1 is the condition for Z phase input, the counting will be performed once whenever the rising-edge trigger of X0.1 occurs.
Mode 1 Outputs the clear signal after returning to the zero point (Output clear point is set in HWCONFIG)	0 or number of pulses or number of Z phases	12 ~ 15 (bit3=bit2 =ON)	After the zero point in mode 1 is returned to, the 20ms-width clear signal is output.
DOG point is B point		+16 (bit4=ON)	When in the low 16-bit code, bit 4 is ON, it means the zero point is ON as DOG point changes from

			ON to OFF and the zero point is left as DOG point changes from OFF to ON.

4. The execution sequence is as follows based on the value of the low 16-bit code in the table above:

   The DOG signal is determined by the value of bit4 → Mode 0 or mode 1 for the zero point return is selected according to the value of bit3. → The operation of the zero point return is performed according to the values of bit 1 and bit 0 → The operation of signal clear specifief by bit2 is performed.

5. Please set the input point and the rising edge trigger condition in HWCONFIG, when the position control system needs the positive and negative limit input points. Note that the limit input points must not be the same as the zero point and Z phase input points.

6. The Completion flag will be set to ON after the instruction finishes performing the specified function. For example, selecting the function of code 6, PLC will set the Completion flag to ON only when the Z phase seeking is completed.

7. After DZRN instruction is enabled, the interrupt service program will not be executed till DZRN instruction is disabled if the specified input point for zero point is the same as that for the external input interrupt in the program.

8. When the limit switch is started by HWCONFIG and the external input interrupt service program exists, the interrupt program will be executed together.

9. Steps for performing the functions:

Function code 0:

① The DZRN function is enabled and the search for the zero point is toward the negative direction with the target frequency.

② After the DOG signal is received, the output frequency will decrease into JOG frequency. The output will continue toward the negative direction and will not stop till the zero point signal changes from ON to OFF.

③ The output will stop when the signal changes from ON to OFF after DOG signal is left.

Function code 1:

① The DZRN function is enabled and the search for the zero point is toward the negative direction with the target frequency.

② After the DOG signal is received, the output is toward the positive direction with the JOG frequency after the output frequency is decreased and the motion direction is reversed. The output will not stop till the zero point signal changes from ON to OFF.

③ DOG signal is left and PLC will stop when the signal changes from ON to OFF.

Function code 2:

① The DZRN function is enabled and the search for the zero point is toward the negative direction with the target frequency.

②After DOG signal is received, the output will decrease the frequency to JOG frequency and will continue toward the negative direction.

③ When DOG signal is left and the signal changes from ON to OFF, the number of pulses specified will be output.

④ The first pulse output starts.

⑤ When the 100th pulse output is completed, PLC will stop and the Completion flag will be ON.

Function code 3:

① DZRN function is enabled and the search for the zero point is toward the negative direction with the target frequency.

② After the DOG signal is received, the output frequency will decrease to JOG frequency and the motion will continue toward the negative direction.

③ The motion will go on according to the number of Z phases when the signal changes from ON to OFF after DOG signal is left.

④ The first Z phase pulse

⑤ The motion will stop after the 2nd Z phase is completed and the Completion flag will be ON.

Function code 4:

① DZRN function is enabled and the search for the zero point is toward the negative direction with the target frequency.

② After the DOG signal is received, the output frequency will decrease to JOG frequency and will continue toward the negative direction. The output will not stop till the zero point signal changes from ON to OFF.

③ After DOG signal is left, the output will stop when the signal changes from ON to OFF and the clear signal will be ON for about 20 milliseconds.

Function code 0+ the negative limit function enabling:

The negative limit input point is set in HWCONFIG and then the setting is downloaded to PLC. (PLC will automatically judge the negative limit function when the instruction is enabled.)□

① DZRN function is enabled and the search for the zero point is toward the negative direction with the target frequency.

② After the negative limit switch is ON, the motion will stop and then will go toward the positive direction after the direction is reversed.

③ The motion will continue to go toward the positive direction after leaving the negative limit switch.

④ The output frequency will ramp down after DOG signal is received. The reverse output will be performed with the JOG frequency after the direction is reversed.

⑤ The output will stop when the signal changes from ON to OFF after the DOG signal is left.

Example 1:

When M0 is ON, the zero return is started by outputting the pulse from Y0.0 with the frequency of 20kHz.

When DOG signal is received and X0.4 is ON, Y0.0 outputs the pulse with the JOG frequency of 1kHz. And the output will not stop till X0.4 changes from ON to OFF.

Example 2:

When M0 is ON, the zero return is performed by outputting the pulse from Y0.0 with the frequency of 20kHz.

When DOG signal is received and X0.4 is ON, PLC decreases the frequency to the JOG frequency of 1kHz and the output is conducted with the JOG frequency of 1kHz. When X0.4 is OFF, PLC starts to seek the Z phase pulse. When X0.5 receives two pulses, PLC stops and Y1.4 outputs a 20ms-width pulse.

Explanation:

1. If the rising-edge trigger at X0.5 is specified as the condition for Z phase input in HWCONFIG, the counting will be performed once whenever the rising-edge trigger at X0.5 occurs.

2. Y1.4 is specified as the output point for outputting the clear signal in HWCONFIG.

3. S_3 is set as 16#00020007. The value of the high 16-bit code, 02 means to search Z phase twice. The value of the low 16-bit code, 07 means to output the clear signal (20ms wide) after the Z phase is found.

API	Instruction			Operand	Description
2705	D	PLSV		**S，D_1，D_2**	Adjustable pulse output

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S							●	●	●		○		○	○		
D_1		○														
D_2		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S : Pulse output frequency

D_1 : Pulse output device

D_2 : Pulse direction output device

6

Explanation:

1. **S_1** specifies the pulse output frequency. The range is -4MHz ~ +4MkHz for the line driver output models and -200kHz ~ +200kHz for the open collector output models. The minus sign and plus sign indicate the positive direction and negative direction. The pulse output frequency can be changed during the pulse output. But if the new frequency is different from the previous frequency in direction, the instruction will stop the output and one cycle later it will output the target frequency again.

2. Refer to the following table for selection of pulse output devices for **D_1** and **D_2**. When the output mode for **D_1** and **D_2** is not the default "Pulse+direction" output (0 is the default value), please modify the mode by setting SR to 1. And the mode is changed into A/B phase output.

Axis number	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
Output point for **D_1**	Y0.0	Y0.2	Y0.4	Y0.6	Y0.8	Y0.10
Direction output point for **D_2**	Y0.1	Y0.3	Y0.5	Y0.7	Y0.9	Y0.11

Output mode	SR462	SR482	SR502	SR522	SR542	SR562
Busy flag	SM460	SM480	SM500	SM520	SM540	SM560
Output present register	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521	SR540 SR541	SR560 SR561

3. The output point for the pulse direction output device **D_2** will be affected by the scan cycle if it is not the output point which is recommended to use.

4. The pulse direction output device **D_2** will change its own state according to the minus sign and plus sign of **S**. **D_2** is OFF if **S** is plus (+) and **D_2** is ON if **S** is minus (-).

5. There is no ramp-up/down setting in the instruction. Thus the ramp-up operation at the beginning and ramp-down stop operation can not be performed. If the function of ramp-up/down must be achieved, please use API 0703 DRAMP instrunction for increasing and decreasing of the pulse output frequency.

6. While the instruction is executing the pulse output, the output will immediately stop if the drive condition changes to OFF.

Example:

When M10 is ON, Y0.0 outputs the pulse with the frequency of 20kHz. Y0.1 = OFF means the positive direction for the pulse output.

API	Instruction			Operand	Description
2706	D	DRVI		S_1, S_2, D_1, D_2	Relative Position control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
D_1		○														
D_2		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Number of output pulses (Relative positioning)

S_2 : Pulse output frequency

D_1 : Pulse output device

D_2 : Pulse direction output device

Explanation:

1. **S_1** is the number of output pulses (Relative positioning). The available range is -2,147,483,648 ~ +2,147,483,647 of which the "+/-"signs indicate the positive and negative directions.

2. **S_2** is the pulse output frequency. Available range: 1Hz ~ +200k Hz for open collector output models and1Hz ~ +4Mk Hz for line driver output models.

3. Refer to the following table for selection of pulse output devices for **D_1** and **D_2**. When the output mode for **D_1** and **D_2** is not the default "Pulse+direction" output (0 is the default value), please modify the mode by setting SR to 1. And the mode is changed into A/B phase output.

Axis number	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
Output point for **D_1**	Y0.0	Y0.2	Y0.4	Y0.6	Y0.8	Y0.10
Direction output point for **D_2**	Y0.1	Y0.3	Y0.5	Y0.7	Y0.9	Y0.11
Output mode	SR462	SR482	SR502	SR522	SR542	SR562

DDRVI and DDRVA support the output devices for **D_1** which can be the odd points as shown in the following table. Only the "Pulse+direction" output mode is supported. The input points Y0.0, Y0.2…Y0.10 are not recommended to use as the high-speed output points.

Axis number	Axis 7	Axis 8	Axis 9	Axis 10	Axis 11	Axis 12
Output point for **D_1**	Y0.1	Y0.3	Y0.5	Y0.7	Y0.9	Y0.11
Direction output point for **D_2**	Y1.0 or above or any M deivice (Bool)					

4. If the pulse direction output device for **D_2** is not the output point recommended in the table above, the output will be affected by the scan time.

5. When **D_2** uses the output mode "pulse+direction" and **S_1** has the minus sign, **D_2** is ON. When **S_1** has the plus sign, **D_2** is OFF. **D_2** will not be OFF immediately after the pulse output in the negative direction is completed. **D_2** is switched to OFF when the contact for the execution of the instruction is OFF.

6. For the instruction execution, the parameters such as the start/end frequency and ramp-up/down time need be used. For SM/SR which axes correspond to, refer to the following table.

When **D_1** selects the even output point as the output device

Output point for **D_1**	Y0.0	Y0.2	Y0.4	Y0.6	Y0.8	Y0.10
Busy flag	SM460	SM480	SM500	SM520	SM540	SM560
Completion flag	SM461	SM481	SM501	SM521	SM541	SM561
Reversing the output direction	SM462	SM482	SM502	SM522	SM542	SM562
Stop flag	SM463	SM483	SM503	SM523	SM543	SM563
Enabling S curve ramp-up/down	SM468	SM488	SM508	SM528	SM548	SM568
Enabling fixed slope ramp-up/down	SM469	SM489	SM509	SM529	SM549	SM569
Output completion auto-reset	SM470	SM490	SM510	SM530	SM550	SM570
Executing an interrupt program when pulse output ends	SM471	SM491	SM511	SM531	SM551	SM571
Corresponding interrupt I number	I500	I501	I502	I503	I504	I505

The output immediately stops when the instruction is disabled or stops	SM476	SM496	SM516	SM536	SM556	SM576
Present output position	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521	SR540 SR541	SR560 SR561
Start/end frequency(Hz）	SR463	SR483	SR503	SR523	SR543	SR563
Ramp-up time（ms）	SR464	SR484	SR504	SR524	SR544	SR564
Ramp-down time（ms）	SR465	SR485	SR505	SR525	SR545	SR565
Target frequency of the fixed slope	SR472 SR473	SR492 SR493	SR512 SR513	SR532 SR533	SR552 SR553	SR572 SR573
S curve mode	SR604	SR605	SR606	SR607	SR608	SR609
Present output frequency	SR610 SR611	SR612 SR613	SR614 SR615	SR616 SR617	SR618 SR619	SR620 SR621

When **D_1** selects the odd output point as the output device

Output point for **D_1**	Y0.1	Y0.3	Y0.5	Y0.7	Y0.9	Y0.11
Busy flag	SM472	SM492	SM512	SM532	SM552	SM572
Completion flag	SM473	SM493	SM513	SM533	SM553	SM573
Output stop flag	SM474	SM494	SM514	SM534	SM554	SM574
Output completion auto-reset	SM475	SM495	SM515	SM535	SM555	SM575
The output immediately stops when the instruction is disabled or stops	SM477	SM497	SM517	SM537	SM557	SM577
Present output position	SR474 SR475	SR494 SR495	SR514 SR515	SR534 SR535	SR554 SR555	SR574 SR575
Start/end frequency(Hz）	SR476	SR496	SR516	SR536	SR556	SR576
Ramp-up/down time（ms）	SR477	SR497	SR517	SR537	SR557	SR577

The flags such as S curve ramp-up/down, fixed slope ramp-up/down and executing an interrupt program when pulse output ends are not supported by the odd output points.

If the output is ongoing, the preset output is the ramp-down stop when the instruction is disabled or stops temporarily. If the output has to be stopped immediately, users should set the immediately stop flag to ON.

7. Refer to the following table for Attribute, Factory default and Latched of SM/SR that axes correspond to.

Item	Attribute	Stop→Run	Run→Stop	Power ON	Factory default	Latched
Busy flag	R	OFF	OFF	OFF	OFF	N
Completion flag	R / W	OFF	OFF	OFF	OFF	N
Stop flag	R / W	OFF	OFF	OFF	OFF	N
Output completion auto-reset	R / W	OFF	OFF	OFF	OFF	N
Enabling S curve ramp-up/down	R / W	OFF	OFF	OFF	OFF	N
Enabling fixed slope ramp-up/down	R / W	OFF	OFF	OFF	OFF	N
Executing an interrupt program when pulse output ends	R / W	OFF	OFF	OFF	OFF	N
The output immediately stops when the instruction is disabled or stops	R / W	OFF	OFF	OFF	OFF	N
Present output position	R / Wd	No change	No change	No change	0	Y
Start/end frequency(Hz）	R / Wd	No change	No change	No change	200	Y
Ramp-up time（ms）	R / Wd	No change	No change	No change	200	Y
Ramp-down time（ms）	R / Wd	No change	No change	No change	200	Y
Target frequency of the fixed slope	R / Wd	No change	No change	No change	0	Y
S curve mode	R / Wd	No change	No change	0	0	N

Present output frequency	R	0	0	0	0	N

Note: R mean the device is readable. W means the device is writable. Wd means the device is writable any time execpt when the high-speed output is performed.

8. After the function of output completion auto-reset is enabled, the flag will be reset to OFF automatically when the output is completed. Thus the flag need be set to ON again if the auto-reset function is needed for the next output operation. It is suggested that the rising-edge or falling-edge should be used to trigger the flag every time. The function is usually used in the case that the output instruction DDRVI can not be scanned and executed by PLC program after DDRVI is enabled. For example, the instruction is enabled in the interrupt program.

9. If the Completion flag is used for setting the I interrupt function, the flag will automatically be reset to OFF when the output is completed and the interrupt program is entered. Thus the flag need be set to ON again for the next output in which the interrupt occurs. It is suggested that the rising-edge or falling-edge should be used the trigger the flag every time.

Example 1:

When M10 is ON, Y0.0 outputs 20,000 pulses with the frequency of 2kHz (Relative positioning). Y0.1 = OFF means the positive direction.

Additional remark:

1. Explanation of relative positioning: specify the travel distance from current position with the “+/-” signs.

2. Setting ramp up/down and the items for relative positioning

a) Y0.0 output curve diagram

b) Several DRVI instructions can be edited using the same output point in the user program. Only one instruction can be started whenever the PLC program is executed. For example, when one instruction is used to start Y0.0 for output, then other instructions which also use Y0.0 for output will not be executed. The instruction which starts the output point first will use the point first.

c) After the instruction is enabled, the modification of other all parameters will not be accepted till the instruction is disabled.

3. The ramp-up and ramp-down for fixed slopes

a) Y0.0 output curve diagram

b) Start the fixed slope flag SM469 and write the target frequency of the fixed slope into SR472.

c) The new slope will take the place of the orginal slope and the positioning will be performed by DDRVI as the figure above shows.

d) The general slope is determined by the start/end frequency, target frequency and ramp up/down time.

The fixed slope is determined by the start/end frequency, target frequency of the fixed slope and ramp up/down time.

e) When the target frequency of the fixed slope is less than the target frequency, the fixed slope function will not be started.

4. The stop flag and immediately stop flag

a) The output diagram based on example 1

b) Ramp-down stop

① When M10 is ON, DDRVI instruction is enabled and starts to output pulses. And SM460, the Busy flag is ON.

② When the instruction is disabled (M10 is OFF) or the stop flag SM463 is ON, the state of the immediately stop flag SM476 will be confirmed.

③ If the immediately stop flag SM476 is OFF, the output will ramp down till stop and then SM460 the Busy flag is OFF.

④ If M10 is not OFF and the stop flag SM463 is not ON after the instruction is enabled, Y0.0 will stop the output after outputting 20,000 pulses.

c) Immediately stop

② If the instruction is disabled (M10 is OFF) or the stop flag SM463 is ON, the state of the immediately stop flag SM476 will be confirmed.

③ If the immediately stop flag SM476 is ON, the output will stop immediately and SM460 the Busy flag will be OFF.

5. Hardware limit function

a) The output curve diagram of axis 1 (Y0.0/Y0.1)

b) Operation of the positive limit

⓪ Positive limit input point X0.6 is set in HWCONFIG and then the setting is downloaded to PLC. When SM464 is set to ON, it means that the positive limit function is started.

① DDRVI outputs 20,000 pulses in the positive direction. (**Note:** the instruction will not output in the positive direction if the positive limit point is ON when the function is started)

② When the limit input point X0.6 is triggered by the external mechanism and is ON, it indicates that the condition for the hardware positive limit function is met.

③ DDRVI instruction will stop the output immediately and the positive limit alarm SM465 is ON.

④ If the condition for the hardware limit function is not met, Y0.0 will stop the output after outputting 20,000 pulses.

c) Operation of the negative limit

⓪ Negative limit input point X0.7 is set in HWCONFIG and then the setting is downloaded to PLC. When SM466 is set to ON, it means that the negative limit function is started.

① DDRVI outputs 20,000 pulses in the negative direction. (**Note:** the instruction will not output in the negative direction if the negative limit point is ON when the function is started)

② When the limit input point X0.7 is triggered by the external mechanism and is ON, it indicates that the condition for the hardware negative limit function is met.

③ DDRVI instruction will stop the output immediately and the negative limit alarm SM467 is ON.

④ If the condition for the hardware limit function is not met, Y0.0 will stop the output after outputting -20,000 pulses.

d) SM flags related to the limit function

Axis number	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
Enabling the hardware positive limit	SM464	SM486	SM506	SM526	SM546	SM566
Positive limit alarm flag	SM465	SM487	SM507	SM527	SM547	SM567
Enabling the hardware negative limit	SM466	SM488	SM508	SM528	SM548	SM568
Negative limit alarm flag	SM467	SM489	SM509	SM529	SM549	SM569

Note: The limit point alarm flag is an only-read flag and is set or reset automatically by PLC according to the state of the start flag.

6. Software limit function

a) The output curve diagram of axis 1 (Y0.0/Y0.1)

b) Operation of the positive limit

⓪ Positive software limit position is set in HWCONFIG (by taking SR580=11,000 for example). After the setting is downloaded to PLC, it means the positive limit function is started.

① DDRVI starts to output 20,000 pulses in the positive direction. (**Note:** the instruction will not output in the positive direction if the positive limit is exceeded when the function is started)

② When the current position SR460 > the limit position SR580, it means that the condition for the software limit function is met.

③ DDRVI instruction will stop the output immediately and the positive limit alarm SM465 is ON.

④ If the condition for the limit function is not met, Y0.0 will stop the output after outputting 20,000 pulses.

a) Operation of the negative limit

⓪ Negative software limit position is set in HWCONFIG (by taking SR582=-11,000 for example). After the setting is downloaded to PLC, it means the negative limit function is started.

① DDRVI starts to output 20,000 pulses in the negative direction. (**Note:** the instruction will not output in the negative direction if the negative limit is exceeded when the function is started)

② When the current position SR460 < the limit position SR582, it means that the condition for the software negative limit function is met.

③ DDRVI instruction will stop the output immediately and the negative limit alarm SM467 is ON.

④ If the condition for the software limit function is not met, Y0.0 will stop the output after outputting -20,000 pulses.

b) SR related to the limit function

Axis number	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
Positive software limit position	SR580 SR581	SR584 SR585	SR588 SR589	SR592 SR593	SR596 SR597	SR600 SR601
Negative software limit position	SR582 SR583	SR586 SR587	SR590 SR591	SR594 SR595	SR598 SR599	SR602 SR603

Positive output limit: The output will stop immediately if the current position is greater than the positive limit position.

Negative output limit: The output will stop immediately if the current position is less than the negative limit position.

When the positive and negative limits are both 0, the software limit function can not be started. Since the

software limit function will check the output position when the output instruction is scanned, the stop of the output will be affected by the PLC scan. To quickly stop the output in real time, please use the external input point as the limit point.

7. **ST Example:**

```
0001 IF M0 THEN
0002 DDRVI(1000,1000,Y0.0,Y0.1);
0003 ELSIF SM461 THEN
0004 SM470:=TRUE;
0005 END_IF;
0006
0007
```

Explanation

1. When M0 is ON, Y0.0 outputs 1000 pulses with the frequency of 1kHz.
2. When the pulse output is completed, SM461 is ON and meanwhile SM470=ON is triggered.
3. The pulse output will restart when M0 changes from OFF to ON again.

API	Instruction			Operand	Description
2707	D	DRVA		S_1, S_2, D_1, D_2	Absolute position control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
D_1		○														
D_2		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Number of output pulses (Absolute positioning)

S_2 : Pulse output frequency

D_1 : Pulse output device

D_2 : Pulse direction output device

Explanation:

1. S_1 is the number of output pulses (Relative positioning). The available range is -2,147,483,648 ~ +2,147,483,647 among which the "+/-"signs indicate the positive and negative directions.

2. S_2 is the pulse output frequency. Available range: 1Hz ~ +200k Hz for open collector output models and1Hz ~ +4Mk Hz for line driver output models.

3. The absolute positioning is that the pulse output is conducted from current position till the specified target position is reached. For example, the number of output pulses at current position is 100 and the number of pulses at the target position S_1 is set to 1000. So the number of the actual output pulses is 1000-100=900.

4. Refer to DDRVI instruction for relevant explanation.

Example:

If the value of the present output position SR460 (32-bit) is 100 and M10 is ON, by means of DDRVA instruction, Y0.0 will output pulses with the frequency of 2kHz till the value in SR460 becomes 20,000 (Absolute positioning). Y0.1 = OFF means the positive direction.

Additional remark:

1. Absolute positioning: the way of specifying the distance from the center (zero point).

API	Instruction	Operand	Description
2708	CSFO	S_1, S_2, S_3, D	Catch speed and proportional output

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2								●								
S_3								●								
S_4								●								
S_5								●								
D_1		○														
D_2								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2			●				●						
S_3			●				●						
S_4		●											
S_5		●											
D_1	●												
D_2			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
—	AS	-

6

Symbol:

S_1 : Signal input device

S_2 : Number of input pulses

S_3 : Pulse input frequency

S_4 : Setting the denominator (output frequency) of the proportion of the input frequency and output frequency

S_5 : Setting the numerator (input frequency) of the proportion of the input frequency and output frequency

D_1 : Pulse output device

D_2 : Pulse output frequency

Explanation:

1. S_1 can only specify X0.0, X0.2, X0.4, X0.6, X0.8 and X0.10 as the input points and occupies two consecutive points for input. The instruction can not be executed if the input points are not among the points specified above. After the

input points are selected, the high-speed counter will be specified automatically. If there is a DCNT instruction with the same high-speed counter in the program, PLC will first execute the instruction which starts the counter first. The input points and corresponding high-speed counters are shown in the following table.

Group number	1	2	3	4	5	6
S_1+0 input point（Phase A）	X0.0	X0.2	X0.4	X0.6	X0.8	X0.10
S_1+1 input point（Phase B）	X0.1	X0.3	X0.5	X0.7	X0.9	X0.11
High-speed counter number	HC202	HC206	HC210	HC214	HC218	HC222
Flag of reversing input direction	SM270	SM271	SM272	SM273	SM274	SM275

2. If the high-speed counters for the instruction can only use the phase A/B input mode, please set the flag of reversing the input direction to ON when MPG is connected but has not rotated yet and the input point of PLC is ON.

3. **S_2** is the number of input pulses. Please use the 32-bit variable to declare the parameter.

4. **S_3** is the frequency of input pulses. Please use the 32-bit variable to declare the parameter with the unit of 1Hz.

5. **S_4** is the denominator (output frequency) of the proportion of the input frequency and output frequency. **S_5** is the numerator (input frequency) of the proportion of the input frequency and output frequency. The range of **S_4** and **S_5** is 1~255. If the setting value exceeds the range, the instruction will take the maximum or minimum value for calculation automatically. For example, if input frequency: output frequency= 5:3, the denominator output frequency is K3 and the numerator input frequency is K5. If the input frequency: output frequency=1:2, the denominator output frequency is K2 and the numerator input frequency is K1.

6. **D_1** can only specify Y0.0, Y0.2, Y0.4, Y0.6, Y0.8 and Y0.10 as the output points and occupies two consecutive points for output.

The output points and corresponding output mode SR are shown in the following table.

Output axis number	1	2	3	4	5	6
D_1+0 output point	Y0.0	Y0.2	Y0.4	Y0.6	Y0.8	Y0.10
D_1+1 output point	Y0.1	Y0.3	Y0.5	Y0.7	Y0.9	Y0.11
Output mode	SR462	SR482	SR502	SR522	SR542	SR562

7. **D_2** is the frequency of output pulses. Please use the 32-bit variable to declare the parameter with the unit of 1Hz.

8. There is no limit to how many times the instruction is used. But its high-speed input and output points can not be used by other instructions whenever the instruction is enabled. Otherwise, the instruction can not be executed.

Note:

1. The input pulse frequency is calculated on the base of the input pulse width (ON) in the positive half cycle. If pulse width for ON: pulse width for OFF is not 1:1, PLC takes the ON width as the standard for conversion by default.

2. The input pulse=ON means the input point LED is on. Please notice whether the input point LED is OFF and whether the flag of reversing the input direction need be activated if the MPG is used but it has not rotated yet.

Example 1:

X0.0/X0.1 input pulses for detection of the MPG. When M0 is ON, the setting values of S_4 and S_5 are both 1 (D4=1 and D5=1). When the input frquency is 10Hz (D0, D1=10) and the number of pulses is 10 (D2, D3=10), the output axis (Y0.0/Y0.1) will output 10 pulses (SR460, SR461=10) with the frequency of 10Hz (D10, D11=10).

Example 2:

When M0 is ON, the setting values of S4 and S5 are 3 and 5 (D4=3, D5=5) respectively. When the input frequency of MPG is 100Hz (D0, D1=100) and the number of pulses is 100 (D2, D3=100), the output axis (Y0.0/Y0.1) will output 60 pulses (SR460, SR461=60) with the frequency of 60Hz (D10, D11=60).

Network 1
M0
MOV
En
3
S
D
D4= 3
MOV
En
5
S
D
D5= 5
Network 2
M0
CSFO
En
X0.0
S1
D1
Y0.0
D0=100
S2
D2
D10= 60
D2=100
S3
D4=3
S4
D5=5
S5

API	Instruction			Operand	Description
2709	D	DRVM		S_1, S_2, S_3, S_4, S_5, S_6, D_1, D_2	Mark alignment positioning

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3	○		○													
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6								●	●				○	○		
D_1		○														
D_2		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3	●												
S_4			●				●						
S_5			●				●						
S_6			●				●						
D_1	●												
D_2	●												

6

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Target number of output pulses

S_2 : Target output frequency

S_3 : Input point number for receiving the external interrupt signal

S_4 : The number of output pulses at the end of the forepart mask section

S_5 : The number of output pulses at the beginning of the rear mask section

S_6 : The number of output pulses in the ramp-down process after the interrupt signal is received

D_1 : Pulse output device

D_2 : Direction pulse output device

Explanation:

1. When the value of **S_1** is 0, it indicates the output is a maximum 32-bit value (with a sign) and the output will not perform the ramp-down stop till the mark signal is received.

If the value of S_1 is not 0 and the external interrupt does not occur, the output pulses of the target number are output. The range of S_1 is -2,147,483,648 ~ +2,147,483,647 of which the "+/-" sign means the positive/negative direction.

2. When the value of S_2 is less than or equal to 0, the output is not enabled. If the output frequency is greater than the maximum frequency, PLC will take the maximum frequency for output.

3. S_3 can specifiy M or X device as the input point. The input source signal will be affected by the scan cycle if the selected input point is not the one among X0.0~X0.15 in PLC .

4. The instruction must be used by combination of the input point number and external interrupt program so as to achieve the real-time ramp-down output. For the operation of rising-edge or falling-edge trigger for the external interrupt, please make the selection of them during the editing of the external interrupt program.

   A. Using the external interrupt (I0xx, I1xx): when the external interrupt occurs, the ramp-down stop is performed after the mark signal is received. No operation is performed if the external interrupt is not enabled.

   B. Without using the external interrupt (I0xx, I1xx): When the instruction is executed, the ramp-down stop will be performed after the mark signal is received if the rising edge occurs at X point and it will be affected by the scan time.

5. The number of output pulses in the forepart mask section is between 1 and S_4. When the setting value for S_4 is 0, it means not to enable the forepart-section mask function. When the number of pulses to be masked exceeds the value of S_1, the instruction will take the target number of output pulses S_1 as the condition for the number to be masked. If the external input trigger occurs within the number of output pulses to be masked, the external input interrupt will be invalid automatically.

6. The number of output pulses in the rear mask section is between S_5 and S_1. When S_5 =0 or S_5 >=S_1, it means that the rear mask section function is not enabled. When the external input trigger occurs in the mask sections, the external input interrupt will be invalid automatically. When S_4 >S_5, it indicates the external input interrupt is invalid in the output process.

7. If the forepart and rear mask sections are both set and S_4 < S_5, it means the valid input interrupt occurs in the section between S_4+1 and S_5-1.

8. If S_6 is set to 0 after the mark signal is received, it means that the ramp-down stop should be performed based on the ramp-down time. If S_6 =-1 or < 0 after the mark signal is received, it means that the output will immediately stop. If the setting value of S_6 is not enough to achieve the ramp-down stop within the ramp-down time, the instruction will limit the target frquency and perform the ramp-down stop in accordance with the set ramp-down time. If S_6 is greater than the number of pulses output within the ramp-down time, the instruction will output the number of the redundant pulses with the output frequency of when the interrupt is triggered and then perform the ramp-down process.

9. If the output has entered the ramp-down process when the external input interrupt trigger occurs, the instruction will complete the output of the number of pulses specified by S_6.

10. See the following timing diagram about the output, interrupt trigger, mask and relavant flags.(For the completion flag, refer to the Completion flag SM of axes in DDRVI instruction.)

11. The target frequency **S_2** of the instuction can be modified in the process of the pulse output.Since the number of output pulses in the ramp-down section has been set when the instruction is enabled, the timing diagram for changing the frequency is shown as the following dot line.

12. Refer to DDRVI instruction for the selection of pulse output devices for **D_1** and **D_2**. If the output points are not the recommended ones, the direction output **D_2** will be affected by the scan cycle.

Example 1:

X0.4 external interrupt program

Note:

1. When M0 changes from OFF to ON, Y0.0 starts to output the pulses. After more than 30,000 pulses are output, the external interrupt is detected on X0.4. The value in D0 will increase by 1 and the pulse output stops after 20,000 pulses are output (S_6:20,000). If the interrupt does not occur, the output will not stop till 100,000 pulses are output.

2. If the number of output pulses is between 1~30,000, it belongs to the forepart mask section. The external interrupt occurs at X0.4 at the moment and the instruction will not perform the operation of the ramp-down stop.

3. If the number of output pulses is between 70,000~100,000, it belongs to the rear mask section. The external interrupt occurs at X0.4 at the moment and the instruction will still not perform the operation of the ramp-down stop.

4. When the pulse output is completed, SM461 is ON and M0 is reset.

Additional remark:

1. The alignment mark function and mask function:

   ① When DDRVM is enabled, the pulse output starts.

   ② When the interrupt occurs for the mark alignment, the ramp-down process is performed and the output stops after the output of the number of the pulses specified by $\mathbf{S_6}$ is completed.

   ③ When the interrupt for the mark alignment does not occur or takes no effect, DDRVM stops outputting the pulses after outputting the target number of pulses specified by $\mathbf{S_1}$.

S_4: The number of output pulses in the forepart mask section. If the interrupt occurs in the section, the interrupt will be ineffective for the mark alignment.

S_5: The number of output pulses in the rear mask section. If the interrupt occurs in the section, the interrupt will be ineffective for the mark alignment.

S_2: The target output frequency

2. The mark alignment function is enabled in the ramp-up and target frequency and ramp-down sections.

When the mark alignment function is not enabled:

The output will stop after 100,000 pulses are output when the mark alignment function is not enabled.

Case 1 of when the mark alignment occurs in the ramp-up section

The operation of the mark alignment is performed when the number of pulses is 3000. The target frequency of 10kHZ can not be reached if S_6 is 15,000.

Case 2 of when the mark alignment occurs in the ramp-up section

The operation of the mark alignment is performed when the number of pulses is 6000. The target frequency of 10kHZ can be reached if S_6 is 15,000.

When the mark alignment occurs in the target frequency section

The operation of the mark alignment is performed when the number of pulses is 50,000.

Case 1 of when the mark alignment occurs in the ramp-down section

The operation of the mark alignment is performed when the number of pulses is 95,000.

Case 2 of when the mark alignment occurs in the ramp-down section

The operation of the mark alignment is performed when the number of pulses is 95,000. After the mark alignment, the number of output pulses for $\mathbf{S}_6$ is -1. (The output stops immediately.)

API	Instruction			Operand	Description
2710	D	PPMR		S_1, S_2, S_3, D_1, D_2	2-Axis relative-coordinate point-to-point synchronized motion

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Number of output pulses of X axis

S_2 : Number of output pulses of Y axis

S_3 : Maximum point-to-point output frequency

D_1 : Pulse output device for X axis

D_2 : Pulse output device for Y axis

Explanation:

1. **S1** and **S2** specify the number of output pulses (relative positioning) of X axis and Y axis. Range: -2,147,483,648 ~ +2,147,483,647 (The "+/-" sign indicates the positive/negative direction).

2. S_3 specifies the maximum point to point output frequency. Range: 1Hz ~ +200k Hz.

3. D_1 and D_2 are the output devices for X axis and Y axis respectively. Users can designate the following 6 axes for output but the fixed direction output points can not be changed. The direction signal: OFF means the positive direction and ON is the negative direction. If the preset "Pulse+direction" output mode is not used (default: 0), change the output mode by setting SR to 1.

Axis number	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
Output points for **D_1** and **D_2**	Y0.0	Y0.2	Y0.4	Y0.6	Y0.8	Y0.10
Fixed direction output point	Y0.1	Y0.3	Y0.5	Y0.7	Y0.9	Y0.11
Output mode	SR462	SR482	SR502	SR522	SR542	SR562

4. PLC will assign the value of **S_3** to X axis or Y axis which output the largest number of pulses. But if the numbers of pulses which the two axes output are excessively different so that no proper output frequency can be calculated through the numbers of their output pulses, PLC will automatically decrease the maximum point to point frequency and no alarm information will appear.

5. The parameters such as the start/end frequency and ramp-up/down time of axes are used in the execution of the instruction. Instead of the originally configured parameters of axes, the X axis parameters are taken as the reference source. For example, When X axis selects axis1 and Y axis selects axis 3 for the output, axis 1 parameters will be taken as the parameter sources for the start frequency and ramp up/down time.

6. See the flags of axes and corresponding SM/SR in the following table.

Axis number	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
Busy flag	SM460	SM480	SM500	SM520	SM540	SM560
Completion flag	SM461	SM481	SM501	SM521	SM541	SM561
Present output position	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521	SR540 SR541	SR560 SR561
Start/end frequency	SR463	SR483	SR503	SR523	SR543	SR563
Ramp-up time	SR464	SR484	SR504	SR524	SR544	SR564
Ramp-down time	SR465	SR485	SR505	SR525	SR545	SR565

7. The instruction has no limit to how many times it is used. But when the instruction is enabled, the instruction can not execute the output till Y axis completes the output and is released if Y axis output is being used.

8. When the 2-axis synchronized pulse output is completed, the corresponding Completion flags of the two axes will be set. Since the same timing for the completion of the 2-axis synchronized pulse output can not be ensured every time, the execution of the next user program should go on after the corresponding Completion flags of the two axes are judged simultaneously.

Example:

1. Draw a rhombus as below.

2. Steps:

- Set the coordinates of four points (0, 0), (-27000, -27000), (0, -55000), (27000, -27000) (as the figure above). Calculate the relative coordinates of the four points and obtain (-27000, -27000), (27000, -28000), (27000, 28000), and (-27000, 27000). Place them in the 32-bit registers (D200, D202), (D204, D206), (D208, D210), (D212, D214).

- RUN the PLC. Set M0 to ON to start the 2-axis line drawing.

6

3. Operation:

When PLC runs and M0 is ON, PLC will start the first point-to-point motion with the frequency of 100KHz. D0 will increase by 1 whenever a point-to-point motion is completed and the second point-to-point motion will start to execute automatically. The operation pattern repeats until the fourth point-to-point motion is completed.

API	Instruction			Operand	Description
2711	D	PPMA		S_1, S_2, S_3, D_1, D_2	2-Axis absolute-coordinate point-to-point synchronized motion

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Number of output pulses of X axis

S_2 : Number of output pulses of Y axis

S_3 : Maximum point-to-point output frequency

D_1 : Pulse output device for X axis

D_2 : Pulse output device for Y axis

Explanation:

1. **S1** and **S2** specify the number of output pulses (absolute positioning) of X axis and Y axis.

2. For relevant explanation, refer to DPPMR instruction.

Example:

1. Draw a rhombus as the figure below.

2. Steps:

- Set the four coordinates (-27000, -27000), (0, -55000), (27000, -27000) and (0, 0) (as the figure above). Place them in the 32-bit registers (D200, D202), (D204, D206), (D208, D210), (D212, D214).
- RUN the PLC. Set M0 to ON to start the 2-axis line drawing.

3. Operation:

When PLC runs and M0 is ON, PLC will start the first point-to-point motion with the frequency of 100KHz. D0 will increase by 1 whenever a point-to-point motion is completed and the second point-to-point motion will start to execute automatically. The operation pattern repeats until the fourth point-to-point motion is completed.

API	Instruction			Operand	Description
2712	D	CICR		S_1, S_2, S_3, S_4, S_5, D_1, D_2	2-Axis relative-position clockwise arc interpolation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
S_4							●	●	●		○		○	○		
S_5							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						
S_5			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

6

Symbol:

S_1 : X axis target coordinate (Relative positioning)

S_2 : Y axis target coordinate (Relative positioning)

S_3 : The shift of the center

S_4 : Target reference frequency

S_5 : Function selection

D_1 : Pulse output device for X axis

D_2 : Pulse output device for Y axis

Explanation:

1. S_1, S_2 and S_3 respectively designate the X axis target coordinate, Y axis target coordinate (relative position) and the shift of the center of a circle. Refer to the following clockwise arc operation for details.

2. D_1 and D_2 are the pulse output devices for X axis and Y axis respectively. Refer to DPPMR instruction for the selection of output points and output mode of axes.

3. **S_4** sets the target frequency for reference. The target reference frequency will be used for the prior calculation when PLC plans the travel path for the arc after the instruction is enabled. But if the estimated calculation process can not achieve the arc travel path, the output frequency will be decreased automatically so as to fulfill the synchronized arc drawing function.

4. **S_5** is the setting value for function selection. When the setting value is 0, the arc resolution takes the 10° arc as the basic angle for motion. When the setting value is 1, the arc resolution takes the 5° arc as the basic angle for motion. When the setting value is not 1, 0 will be taken as the setting value for the execution of the instruction.

Operation of drawing the clockwise arc:

1. See the four quadrants (QI, QII, QIII and QIV) of a coordinate system as below. Point A indicates the current coordinates on X axis and Y axis.

2. Point B is the target coordinates specified by **S_1** and **S_2**. Point O is the center of a circle where point A and point B are in.

3. **S_3** is the distance that the center point O shifts by.

4. When the target point B is in QI and QIV of point A, the arc travel paths are performed as the solid lines in the figures below based on the setting values in **S_3**. (r: radius)

S_3>= 0

S_3< 0

5. When the target point B is in QII and QIII of point A, the arc travel paths are shown as the solid lines in the figures below based the setting values in **S_3**.

6. When of target point B, X axis coordinate = 0 and Y axis target coordinate >= 0, point B is defined as the point staying in QI of point A. Whereas, if Y axis target coordinate < 0, point B is defined as the point staying in QIII of point A.

Example:

1. Draw a DELTA LOGO as the figure below.

2. Steps: the logo can be divided into three parts.

- An ellipse= use DCICR instruction for the relative-position clockwise arc interpolation.
- The absolute coordinates of the ellipse: P1（39930，-9982）, P2（-9982，39930）, P3（73206，123119） and P4（123119，73206）.
- Taking （39930，-9982） as the start point, the obtained relative coordinates: （-49912，49912）, （83188，83189）, （49913，-49913） and （-83189，-83188）.

When PLC runs and M1 is ON, PLC will start the drawing of the first segment of the arc. D1 will increase by 1 whenever a segment of arc is completed and the second segment of the arc will start to

execute automatically. The operation pattern repeats until the fourth segment of arc is completed.

When D1=1, using DCICR, PLC will perform the drawing of the first-segment arc from P1 to P2 with the shift of the center of the circle: 18825 and arc resolution: 5.

When D1=2, using DCICR, PLC will perform the drawing of the second-segment arc from P2 to P3 with the shift of the center of the circle: 110295 and arc resolution: 5.

When D1=3, using DCICR, PLC will perform the drawing of the third-segment arc from P3 to P4 with the shift of the center of the circle: 18825 and arc resolution: 5.

When D1=4, using DCICR, PLC will perform the drawing of the fourth-segment arc from P4 to P1 with the shift of the center of the circle: 110295 and arc resolution: 5.

- A circle = use DCCMR instruction for the relative-position circle drawing.
- The absolute coordinates of the circle: P5（86213，15435） and P6（103048，37425）. Taking （86213，15435） as the start point, the relative coordinates of the center of the circle is （16835，21990）.

When PLC runs and M2 is ON, PLC will start to perform the relative-position circle drawing with the target reference frequency of 20kHz.

- A delta = use DPPMA for the absolute-position 2-axis synchronized motion.
- Absolute coordinates of the delta= start point P7 （-80360，23641）, P8（208552，-23641）and P9（62096，206884）.

When PLC runs and M0 is ON, PLC will start to perform the first segment of 2-axis synchronized motion with the frequency of 50kHz. D1 will increase by 1 whenever a segment of 2-axis synchronized motion is completed and the second segment of 2-axis synchronized motion will start to execute automatically. The operation pattern repeats until the third segment of 2-axis synchronized motion is completed.

When D1=1, PLC will start to draw the line from P7 to P8 with DPPMA.

When D1=2, PLC will start to draw the line from P8 to P9 with DPPMA.

When D1=3, PLC will start to draw the line from P9 to P7 with DPPMA.

API	Instruction			Operand	Description
2713	D	CICA		S_1, S_2, S_3, S_4, S_5, D_1, D_2	2-Axis absolute-position clockwise arc interpolation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
S_4							●	●	●		○		○	○		
S_5							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						
S_5			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

6

Symbol:

S_1 : X axis target coordinate (Absolute positioning)

S_2 : Y axis target coordinate (Absolute positioning)

S_3 : The shift of the center

S_4 : Target reference frequency

S_5 : Function

D_1 : Pulse output device for X axis

D_2 : Pulse output device for Y axis

Explanation:

1. S_1 and S_2 respectively designate the X axis target coordinate, Y axis target coordinate. For relevant explanation, refer to DCICR instruction.

API	Instruction			Operand	Description
2714	D	CICCR		S_1, S_2, S_3, S_4, S_5, D_1, D_2	2-Axis relative-position counterclockwise arc interpolation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
S_4							●	●	●		○		○	○		
S_5							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						
S_5			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

6

Symbol:

S_1 : X axis target coordinate (Relative positioning)

S_2 : Y axis target coordinate (Relative positioning

S_3 : The shift of the center

S_4 : Target reference frequency

S_5 : Function

D_1 : Pulse output device for X axis

D_2 : Pulse output device for Y axis

Explanation:

1. S_1, S_2 and S_3 respectively designate the target coordinate on X axis, target coordinate on Y axis and the shift of the center of a circle. Refer to the following operation of drawing the counterclockwise arc for details.

2. D_1 and D_2 are the output devices for X axis and Y axis respectively. Refer to DPPMR instruction for the selection of output points and output mode of axes.

3. **S_4** sets the target frequency for reference. The target reference frequency will be used for the prior calculation when PLC plans the travel path for the arc after the instruction is enabled. But if the estimated calculation process can not achieve the planned arc travel path, the output frequency will be decreased automatically so as to fulfill the synchronized arc drawing function.

4. **S_5** is the setting value for function selection. When the setting value is 0, the arc resolution takes the 10° arc as the basic angle for motion. When the setting value is 1, the arc resolution takes the 5° arc as the basic angle for motion. When the setting value is not 1, 0 will be taken as the setting value for the execution of the instruction.

Operation of drawing the counterclockwise arc:

1. See the four quadrants (QI, QII, QIII and QIV) of a coordinate system as below. Point A indicates the current coordinates on X axis and Y axis.

2. Point B is the target coordinates specified by **S_1** and **S_2**. Point O is the center of a circle where point A and point B are in.

3. **S_3** is the distance that the center point O shifts by.

4. When the target point B is in QI and QIV of point A, the arc travel paths are performed as the solid lines in the two figures below based on the setting values in **S_3**. (r: radius)

S_3>= 0

S_3< 0

5. When the target point B is in QII and QIII of point A, the arc travel paths are shown as the solid lines in the figures below based the setting values in **S_3**.

S_3>= 0

S_3< 0

6. When X axis coordinate of target point B = 0 and Y axis coordinate >= 0, point B is defined as the point staying in QI of point A. Whereas, if Y axis coordinate < 0, point B is defined as the point staying in QIII of point A.

Example:

1. Draw a DELTA LOGO as the figure below.

2. Steps: the logo is divided into three parts.

- An ellipse= use DCICR instruction for the relative-position counterclockwise arc interpolation.
- The absolute coordinates of the ellipse: P1（39930，-9982）, P4（123119，73206）, P3（73206，123119） and P2（-9982，39930）
- Taking （39930，-9982） as the start point, the obtained relative coordinates: （83189，83188）, （-49913，49913）, （-83188，-83189） and（49912，-49912）.

Network 1

Network 2

Network 3

Network 4

Network 5

Network 6

When PLC runs and M1 is ON, PLC will start the drawing of the first segment of clockwise arc with the frequency of 50kHz. D1 will increase by 1 whenever a segment of arc is completed and the second segment of arc will start to execute automatically. The operation pattern repeats until the fourth segment of arc is completed.

When D1=1, using DCICCR, PLC will perform the drawing of the first-segment arc from P1 to P4 with the shift of the center of the circle: -110295 and arc resolution: 5.

When D1=2, using DCICCR, PLC will perform the drawing of the second-segment arc from P4 to P3 with the shift of the center of the circle: -18825 and arc resolution: 5.

When D1=3, using DCICCR, PLC will perform the drawing of the third-segment arc from P3 to P2 with the shift of the center of the circle: -110295 and arc resolution: 5.

When D1=4, using DCICCR, PLC will perform the drawing of the fourth-segment arc from P2 to P1 with the shift of the center of the circle: -18825 and arc resolution: 5.

- A circle = use DCCMR instruction for the relative-position circle drawing. Refer to the example in API 2712 DCICR for detailed operation.
- A delta = use DPPMA for the absolute-position 2-axis synchronized motion. Refer to the example in API 2712 DCICR for detailed operation.

API	Instruction			Operand	Description
2715	D	CICCA		S_1, S_2, S_3, S_4, S_5, D_1, D_2	2-Axis absolute-position counterclockwise arc interpolation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
S_4							●	●	●		○		○	○		
S_5							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						
S_5			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

6

Symbol:

S_1 : X axis target coordinate (Absolute positioning)

S_2 : Y axis target coordinate (Absolute positioning)

S_3 : The shift of the center

S_4 : Target reference frequency

S_5 : Function

D_1 : Pulse output device for X axis

D_2 : Pulse output device for Y axis

Explanation:

1. S_1 and S_2 respectively designate the target coordinate on X axis, target coordinate on Y axis. For relevant explanation, refer to DCICCR instruction.

API	Instruction			Operand	Description
2716	D	CCMR		S_1, S_2, S_3, S_4, D_1, D_2	The relative-position circle drawing

Devie	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
S_4							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : X axis coordinate of the center (Relative positioning)

S_2 : Y axis coordinate of the center (Relative positioning)

S_3 : Function selection

S_4 : Target reference frequency

D_1 : Pulse output device for X axis

D_2 : Pulse output device for Y axis

Explanation:

1. S_1 and S_2 are respectively the X axis coordinate and Y axis coordinate of the center of a circle. (Relative position)

2. S_3 is the setting value for function selection. When the setting value is 0, the arc resolution takes a 10° arc as the basic angle for clockwise motion. When the setting value is 1, the arc resolution takes a 5° arc as the basic angle for clockwise motion. When the setting value is 2, the arc resolution takes a 10° arc as the basic angle for counterclockwise motion. When the setting value is 3, the arc resolution takes a 5° arc as the basic angle for counterclockwise motion.

3. **D_1** and **D_2** are the pulse output devices of X axis and Y axis respectively. Refer to DPPMR instruction for the selection of output points and output mode of axes.

4. **S_4** sets the target frequency for reference. The target reference frequency will be used for the prior calculation when PLC plans the travel path for the arc after the instruction is enabled. But if the calculation process which is estimated can not achieve the planned arc travel path, the output frequency will be decreased automatically so as to fulfill the synchronized arc drawing function.

   When the X axis coordinate of the center point O=0, point O is defined as the point staying in QI of point A if Y axis target coordinate >= 0. Whereas, if Y axis target coordinate < 0, point B is defined as the one staying in QIII of point A.

The clockwise circle drawing:

Point A is the present position and point O is the target center, r is the radius of the circle as the figures below.

Center O in QI and QIV

Center O in QII and QIII

The counterclockwise circle drawing:

Center O in QI and QIV

Center O in QII and QIII

Example:

1. Taking (40000, 50000) as the center O and (10000, 10000) as point A, draw a circle as below.

2. Steps:

- The obtained relative coordinates of point O: (30000, 40000)
- The coordinates of point A is the present position; SR460=10000 and SR480=10000
- S_3=1 and the clockwise circle drawing is performed with a 5° as the basic motion angle.
- When PLC runs and M0 is set to ON, the relative-position circle drawing starts.

3. Operation:

- When PLC runs and M0 is ON, PLC starts to perform the clockwise circle drawing with the frequency of 100kHz.
- When the circle drawing is completed, SM461 and SM481 are ON.

API	Instruction			Operand	Description
2717	D	CCMA		S_1, S_2, S_3, S_4, D_1, D_2	The absolute-position circle drawing

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
S_4							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : X axis coordinate of the center (Absolute positioning)

S_2 : Y axis coordinate of the center (Absolute positioning)

S_3 : Function selection

S_4 : Target frequency

D_1 : Pulse output device for X axis

D_2 : Pulse output device for Y axis

Explanation:

1. S_1 and S_2 are respectively the X axis coordinate and Y axis coordinate of the center of a circle. (Absolute positioning). For other explanation, refer to DCCMR instruction.

Example:

1. Taking (40000, 50000) as the center O and (10000, 10000) as point A, draw a circle as below.

2. Steps:

- Fill the absolute X coordinate and Y coordinate (40000, 50000) in the instruction
- Point A is the present position, SR460=10000 and SR480=10000
- S_3 is 1 and the clockwise circle drawing is performed with a 5° as the basic motion angle.
- When PLC runs and M0 is ON, the absolute-position circle drawing starts.

3. Operation:

- When PLC runs and M0 is ON, PLC starts to perform the clockwise circle drawing with the frequency of 100kHz.
- When the circle drawing is completed, SM461 and SM481 are ON.

API	Instruction	Operand	Description
2718	TPO	S，D_1，D_2	The position planning table controls the output

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S								●				○	○	○		
D_1								●								
D_2		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D_1		●			●	●							
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	AS	-

Symbol:

S : The first output number in the position planning table

D_1 : The output number which is being output

D_2 : Switch flag at the end of the consecutive-number output

6

Explanation:

1. **S** is the number listed in the position planning table. If the number does not exist in the table, the instruction will not be executed, SM0 will be ON and the error code SR0=16#2027 will be kept.

2. **S** is the first output number when PLC is enabled. Refer to the value in **D_1** if there are consecutive-number outputs during the output.

3. When one single number or the first one of consecutive numbers is output, the switch flag **D_2** will be set to OFF. When the consecutive-number output reaches the output number to be switched to, **D_2** will be ON for a scan cycle and will be ON again till the last number output is completed.

   The example and timing diagram for the flag state switching are shown as below.

4. The switch flag **D_2** will be affected by PLC scan time. If the time for switching to the next number of output is shorter than the scan time, the flag for switching to the number of output may not reset to OFF.

API	Instruction			Operand	Description
2719	D	TPWS	P	S_1, S_2, S_3	Setting single-axis output parameters in the position planning table

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S_1 : A number listed in the position planning table

S_2 : Target position

S_3 : Target speed

6

Explanation:

1. S_1 is the number listed in the position planning table. If the number does not exist in the table or the number is not under Single-axis point-to-point motion or Single-axis multi-segment motion, the instruction will not be executed, SM0 will be ON and the error code SR0=16#2027 will be kept.

2. S_2 is the target position which is a 32-bit integer. If the mechanical unit conversion is used during the editing of the position planning table in the software, please use the conversion instruction for modification first.

3. S_3 is the target speed. The range: 1Hz~200,000Hz. (Note: The maximum speed of the output axis Y0 and Y2 can be set to 4MHz for AS324MT.)

4. When the instruction is enabled to modify the parameters of the single axis which is outputting pulses, the modified parameters will be kept in the table and will not be effective till the next output is started.

5. The parameters modified by the instruction can only be modified while PLC is running. The last written parameter will not be kept when the power turns OFF. The table which is edited in the software and downloaded to PLC is seen as the default position planning table whenever the power is ON.

API	Instruction			Operand	Description
2720	D	TPWL	P	S_1, S_2, S_3, S_4	Setting linear interpolation parameters in the position planning table

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S_1 : A number listed in the position planning table

S_2 : Target position of X axis

S_3 : Target position of Y axis

S_4 : Target speed

Explanation:

1. S_1 is the number listed in the position planning table. If the number does not exist in the table or the output of the number does not belong to 2-axis linear interpolation, the instruction will not be executed, SM0 will be ON and the error code SR0=16#2027 will be kept.

2. S_2 and S_3 are respectively the target positions of X axis and Y axis which can only be 32-bit integers. If the mechanical unit conversion is used during the editing of the position planning table in the software, please use the conversion instruction for modification first.

3. **S_4** is the target speed. The range: 1Hz~200,000Hz.

4. When the linear interpolation is executed, the target frequency **S_4** will automatically correspond to the output of an axis which is farthest from its target position. If X axis and Y axis can not achieve the simultaneous reaching of the target positions, PLC will automatically decelerate the frequency so as to make the two axes reach the target positions simultaneously.

6. When the instruction is enabled to modify parameters of two axes any one of which is outputting, the modified parameters of the two axes will be kept in the table and will not be effective till the next 2-axis output is started.

5. The parameters modified by the instruction can be modified only while PLC is running. The last written parameter will not be kept when the power turns OFF. The table which is edited in the software and downloaded to PLC is seen as the default position planning table whenever the power is ON.

API	Instruction			Operand	Description
2721	D	TPWC	P	S_1, S_2, S_3, S_4, S_5	Setting arc interpolation parameters in the position planning table

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						
S_5			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
AS	-	AS

Symbol:

S_1 : A number listed in the position planning table

S_2 : X axis coordinate of the center

S_3 : Y axis coordinate of the center

S_4 : The shift of the center

S_5 : Target speed

Explanation:

1. S_1 is the number listed in the position planning table. If the number does not exist in the table or the output of the number does not belong to 2-axis arc interpolation, the instruction will not be executed, SM0 will be ON and the error code SR0=16#2027 will be kept.

2. **S_2** and **S_3** are respectively the coordinates of the center on X axis and Y axis. **S_4** is the shift of the center. The three parameters can only be 32-bit integers. If the mechanical unit conversion is used during the editing of the position planning table in the software, please use the conversion instruction for modification first.

3. **S_5** is the target speed. The range: 1Hz~200,000Hz. It is the referece speed for the actual output. PLC will automatically decelerate the speed if the 2-axis synchronized output can not be achieved.

4. When the instruction is enabled to modify parameters of two axes any of which is outputting, the modified parameters of the two axes will be kept in the table and will not be effective till the next 2-axis output is started.

5. The parameters modified by the instruction can be modified only while PLC is running. The last written parameter will not be kept when the power turns OFF. The table which is edited in the software and downloaded to PLC is seen as the default position planning table whenever the power is ON.

The instruction does not support the change of the clockwise and counterclockwise motion direction. For the direction change, please add or modify the output parameters for arc interpolation in the position planning table in the software.

API	Instruction			Operand	Description
2723	D	PPGB		S_1, S_2, S_3, S_4, D_1, D_2	Point-to-point go back and forth

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●								
S_2								●								
S_3								●								
S_4								●								
D_1		○														
D_2			●													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4									●				
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	-	AS

6

Symbol:

S_1 : Relative target position A

S_2 : Relative target position B

S_3 : Target speed

S_4 : Target speed ratio adjusted value (floating point value)

D_1 : Pulse output device

D_2 : Changeable target speed indicator

Explanation:

1. This high speed output instruction is used for a movement going back and forth between 2 target positions which are converted from the two relative target positions specified by S_1 and S_2, applicable for warping machines in the textile industry and winding & binding machines in the cable industry, and many more.

2. When the instruction starts the output, the relative positions S_1 and S_2 need be specified first so that the instruction can make the prior calculation for switching to the next output. After the instruction is enabled, users can modify the target positions which are to exectue but the outputting target positions can not be changed.

① DPPGB instruction is enabled. When **S_1** is 30,000 and **S_2** is -30,000, the motion will go toward the target position A1=30,000.

② If the pulse position >15,000, the relative positions are modified into **S_1**:20,000 and **S_2**:-20,000 and the target position is A1=30,000 at the moment. The target positions obtained through calculation are B1=10,000(A1+ **S_2**) and A2=30,000(B1+ **S_1**) since the outputting target positions can not be modified.

3. **S_3** is the target speed (32-bit integer), and **S_4** is the target speed ratio adjusted value (floating point). The actual speed is the result of multiplying the values of **S_3** and **S_4** and then making the calculation result rounded down to the 32-bit integer. The acceptable input speed range is 1~200kHz. When values are out of range, PLC will automatically take the minimum or the maximum of the range for operation. The ratio formula for the actual target speed is **S_3** x **S_4**. For instance, target speed is 1kHz, the adjusted floating point is 1.2345 and the actual will be 1234Hz.

4. When the outputting is ongoing, the target speed and the adjusted ratio can be modified, and the result will be updated to the actual output speed once the instruction is scanned. But it is suggested that the target speed should not change too greatly in case the calculated deceleration will be affected.

5. The output points for **D_1** can only be Y0.0, Y0.2, Y0.4, Y0.6, Y0.8 and Y0.10 and it occupies 2 consecutive points for output. The output sets and the output special register modes are listed below.

Output axis number	1	2	3	4	5	6
D_1+0 output points	Y0.0	Y0.2	Y0.4	Y0.6	Y0.8	Y0.10
D_1+1 output points	Y0.1	Y0.3	Y0.5	Y0.7	Y0.9	Y0.11
Output modes	SR462	SR482	SR502	SR522	SR542	SR562

6. **D_2** is the flag indicating the changeable target speed and the adjusted ratio. When this flag is ON, that means the executing target speed can be modified. When the flag is switching from ON to OFF, that means it is now decelerating and the current target speed will be seen as the target speed of the next output.

7. This instruction has no limitation on the execution times. But during execution, the appointed high speed axis cannot be occupied by other instructions. Otherwise it will not be executed.

Output timing diagram:

Example:

When M0 is ON, DPPGB instruction starts to perform the motion going back and forth between the target positions which are calculated from two relatives target positions specified by **S_1** and **S_2** at the actual target speed of 8500Hz(5000*1.7). Y0.0 is the output point. Y0.1=OFF means the positive direction.

Network 2
M0
DPPGB
En
D10 S1
D12 S2
D14 S3
D16 S4
D1 Y0.0
D2 M100
Network 3
Y0.1
INCP
En
D0 D
Network 4
=
Q
D0 S1
1 S2
>
Q
SR460 S1
15000 S2
DMOV
En
26000 S
D D10
DMOV
En
-28000 S
D D12
Network 5
=
Q
D0 S1
2 S2
>
Q
SR460 S1
15000 S2
DMOV
En
22000 S
D D10
DMOV
En
-24000 S
D D12
Network 6
=
Q
D0 S1
3 S2
>
Q
SR460 S1
15000 S2
DMOV
En
18000 S
D D10
DMOV
En
-20000 S
D D12
Network 7
=
Q
D0 S1
4 S2
>
Q
SR460 S1
15000 S2
DMOV
En
30000 S
D D10
DMOV
En
-24000 S
D D12
Network 8
=
Q
D0 S1
5 S2
MOV
En
1 S
D D0

Explanation:

1. After the instruction starts the output, modify the relative target positions A for $\mathbf{S_1}$ and B for $\mathbf{S_2}$.

a) The output curve diagram of Y0.0:

b) Explanation:

① When $\mathbf{S_1}$:30,000, $\mathbf{S_2}$:-30,000 and M0 has DPPGB instruction enabled, the motion goes toward the position A1=30,000.

② When D0=1 and the position: number of pulses in SR460 >15,000, the relative positions are changed into $\mathbf{S_1}$:26,000 and $\mathbf{S_2}$:-28,000.

At the moment, the target position is A1=30,000 since the outputting target position can not be modified. The target positions obtained through calculation are B1=2,000(A1-28,000), A2=28,000(B1+26,000).

③ When D0=2 and the position: number of pulses in SR460 >15,000, the relative positions are changed into $\mathbf{S_1}$:22,000 and $\mathbf{S_2}$:-24,000.

At the moment, the target position is A2=28,000 since the outputting target position can not be modified.

The target positions obtained through calculation are B2=4,000(A2-24,000), A3=26,000(B2+22,000).

④ When D0=3 and the position: number of pulses in SR460 >15,000, the relative positions are changed into $\mathbf{S_1}$:18,000 and $\mathbf{S_2}$:-20,000.

The target positions obtained through calculation are B3=6,000(A3-20,000)，A4=24,000(B3+18,000).

⑤ When D0=4 and the position: number of pulses in SR460 >15,000, the relative positions are changed into $\mathbf{S_1}$:30,000 and $\mathbf{S_2}$:-24,000.

The target positions obtained through calculation are B4=0(A4-24,000) and A1=30,000(B4+30,000).

Step	S_1: Relative position	S_2: Relative position	Target position B	Target position A
①	30,000	-30,000	-	A1=30,000
②	26,000	-28,000	B1=2,000	A2=28,000
③	22,000	-24,000	B2=4,000	A3=26,000
④	18,000	-20,000	B3=6,000	A4=24,000
⑤	30,000	-24,000	B4=0	A1=30,000

6.27 Delta CANopen Communication Instructions

6.27.1 List of CANopen Communication Instructions

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2800	INITC	–	–	Initializing the servos for the CANopen communication
2801	ASDON	–	–	Servo-ON and servo-OFF
2802	CASD	–	–	Setting the acceleration time and deceleration time of a servo
2803	–	DDRVIC	–	Servo relative position control
2804	–	DDRVAC	–	Servo absolute position control
2805	–	DPLSVC	–	Servo speed control
2806	ZRNC	–	–	Homing
2807	COPRW	–	–	Writing and reading the CANopen communication data

6.27.2 Explanation of CANopen Communication Instructions

API	Instruction	Operand	Description
2800	INITC	**S**	Initializing the servos for the CANopen communication

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S													○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●											

Pulse Instruction	16-bit instruction	32-bit instruction
—	AS	—

Symbol:

S : The number of servos for initialization

Explanation:

1. Before the instruction is executed, please make sure that "Delta ASD-A2 control" is selected as the working mode of the function card 2 in the hardware configuration software HWCONFIG.

2. The range of **S**: 1~8. When the input value is greater than 8, PLC will automatically take 8 as the value of **S** for the initialization. The set station addresses can only start from 1 and the middle numbers can not be skipped or reserved.

3. SM1681 will be cleared to OFF when the instruction is enabled. SM1681 will be set to ON when all slaves complete the initialization.

4. SM1682 will be set to ON when an error occurs during the execution of the communication. Besides, SR658 will keep the number of the axis in which an error occurs and SR659 will keep the error code.

Example: (The Communication with Delta servo ASD-A2M)

1. Before using Delta CANopen instructions, please mount AS-FCOPM card in AS series PLC (Slot 2 only) and make sure that "Delta ASD-A2 control" is selected as the working mode of AS-FCOPM card 2 in the configuration software HWCONFIG. The baud rate is set to 1000kbps or one of 125k, 250k, 500k and 1000kbps.

Figure 1 Function card 2 setting

Figure 2 Working mode selection

2. Connect AS series PLC to ASDA series with a CANopen communication cable as shown in the figure below. (The terminal resistor of120Ω in AS-FCOPM card need be switched to ON so as to enable the terminal resistor.)

3. Follow the steps below for the basic setting on the panel of ASD-A2.

   a. Set the servo parameter P2-08 to 10 to restore the factory setting.

   b. Repower the servo after power off.

   c. Set P1-01 to 0001 (PR mode).

   d. Set P3-01 to 0400 and the baud rate of the servo for CAN communication to 1.0Mbit/s. (The set baud rate should be the same as that for the PLC.)

   e. Set a station address for every servo based on the quantity of servos. P3-00 of each servo is set to 1, 2 and 3 in order. Maximum 8 servos can be set.

   f. Repower the servo after power off.

   g. The operation can begin after the basic setting is finished.

4. Download the example program and set M0 to ON. Then the instruction will make the servos of station addresses 1~3 initialized. When SM1681 is ON, it means the initialization is completed. (When the servo enters the CANopen mode successfully, CO-LD information will be displayed.)

5. The settings of the servo drive in the AS series initialization process are shown below.

   a. Set P2-30 (auxiliary function) to 5. It means that the servo does not need to store the setting values in EEPROM for permanent retention in order to prolong the servo life span.

   b. Reset P6-02 (PATH#1) to 0 and P6-06(PATH#3) to 0. (It indicates that PATH#1 & #3 in PR mode are cleared)

   c. Set P3-06 (SDI source) to 16#0100. (It means that DI1~DI8 are controlled by the hardware, EDI9 is controlled by the software and EDI10~EDI14 are controlled by the hardware.)

   d. Reset P4-07 (SDI status controlled manually) to 0.

   e. Set P2-36 (EDI9) to 16#0101. (It means that the function of EDI9 is selected as Servo ON.)

   f. Set P0-17 (CM1A) to 1. (It indicates that the mapping parameter is the pulse command output register CMD_O.)

   g. Set P0-18 (CM2A) to 64. (It indicates that the mapping parameter is the pulse command register CMD_E.)

h. Set P5-20~P5-35 (acceleration time) to 1. (It indicates that the acceleration time is 1ms.)

i. Set P5-60~P5-75 (target speed) to 1. (It indicates that the target speed is 0.1rpm)

j. Make PDO1 correspond to P5-07 (PR command), P0-01(Fault code) and P0-46 (state of DO point).

k. Make PDO2 correspond to P0-09 (CM1 state: CMD_O) and P0-10(CM2 state: CMD_E).

6. For the servo parameters set in the above six items a, b, f, g, j and k, do not use COPRW instruction to modify them.

7. When the absolute-type servo is used, users should use the COPRW communication instruction for writing 16#0100 to P3-12 so that relevant absolute-type servo parameters will be written to EEPROM at the moment when power is off.

8. To modify the hardware DI signal setting of ASD-A2, Uses can set relevant DI signal configuration parameters by manual or by using the COPRW instruction. COPRW can be used for the modification of the configuration after the execution of INITC instruction is completed and before the servo is enabled.

9. For more details about the servo parameters, refer to Delta servo operation manual.

API	Instruction	Operand	Description
2801	ASDON	S_1, S_2	Servo-ON and servo-OFF

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●		○	○	○	○		
S_2													○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
—	AS	—

Symbol:

S_1 : Station address of a servo

S_2 : Servo-ON and servo-OFF

Explanation:

1. The execution of INITC must be finished before the instruction is executed.
2. The range of S_1: K1~K8.
3. If S_2 is a non-zero value, the servo is enabled (Servo ON). If S_2 is 0, the servo is disabled (Servo OFF).
4. Every servo has a flag (SM1651~M1658) for displaying own state. The actual servo state can be known from the flag. When the flag is ON, the servo is Servo-ON. When the flag is OFF, the servo is Servo-OFF.
5. SM1682 will be set to ON if an error occurs during the communication. Besides, SR658 will keep the number of the axis in which an error occurs and SR659 will keep the error code.

Example:

1. When M0 changes from OFF to ON, the INITC instruction starts to make the servos of station addresses 1~3 initialized until SM1681 is ON.
2. When M1 changes from OFF to ON, the ASDON instruction starts to make the servo of station address 2 enabled. When SM1652 is ON, it indicates Servo-ON.

Network 1
M0
INITC
En
3
S
SM1681
M0
R
Network 2
M1
ASDON
En
2
S1
1
S2
SM1652
M1
R

API	Instruction		Operand	Description
2802		CASD		S_1 , S_2 , S_3	Setting the acceleration time and deceleration time of a servo

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●		○	○	○	○		
S_2	●	●			●	●		●	●		○	○	○	○		
S_3	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
—	AS	—

Symbol:

S_1 : Station address of a servo

S_2 : Acceleration time（ms）

S_3 : Deceleration time（ms）

Explanation:

1. The execution of INITC must be finished before the instruction is executed.
2. The range of S_1: 1~8.
3. The range of S_2 and S_3: 0~32767. If the values of S_2 and S_3 are out of the range, 0 will be seen as the input value. (Unit: ms)
4. The acceleration time is the period of time during which the servo speeds up from 0 to 3000.0 rpm. The deceleration time is the period of time during which the servo speeds down from 3000.0 rpm to 0.
5. When the instruction is enabled, SM1681 will be cleared to OFF. When the servo has responded to the received command, SM1681 will be ON.
6. SM1682 will be set to ON when an error occurs during the communication. Besides, SR658 will keep the number of the axis in which an error occurs and SR659 will keep the error code.

Example:

1. When M0 changes from OFF to ON, the INITC instruction makes the servos of station addresses 1~3 initialized till SM1681 is ON.

2. When M1 changes from OFF to ON and the target speed of the servo of station addresses 2 is 3000rpm, the CASD instruction sets the acceleration time of servo 2 to 3000ms and the deceleration time to 9000ms.

3. If the target speed of servo 2 is 1000 rpm, the acceleration time and deceleration time will be as follows.

   Acceleration time: [3000ms/3000rpm]*1000rpm=1000ms

   Deceleration time: [9000ms/3000rpm]*1000rpm=3000ms

API	Instruction			Operand	Description
2803	D	DRVIC		S_1 , S_2 , S_3	Servo relative position control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Station address of a servo

S_2 : Relative target position

S_3 : Target speed

Explanation:

The execution of INITC and ASDON (servo ON) instructions must be finished before the instruction is executed.

1. The range of S_1: 1~8.
2. The range of S_2: -2147483648 ~ 2147483647. The +/- sign means the forward / reverse direction. The target position is a relative position.
3. The unit of the value of S_3: 0.1rpm.The range: 1~60000. It indicates 0.1~6000.0 rpm.
4. When the target position is reached, the corresponding finish flags of axes (SM1631~SM1638) will be ON.
5. Refer to the following table for the corresponding SM and SR of axes.
6. SM1682 will be set to ON when an error occurs during the communication. Besides, SR658 will keep the number of the axis in which an error occurs and SR659 will keep the error code.

Example:

1. When M0 changes from OFF to ON, the INITC instruction starts to make the servos of station addresses 1~3 initialized till SM1681 is ON.

2. When M1 changes from OFF to ON, the ASDON instruction starts to make the servo of station address 2 enabled. When SM1652 is ON, it means that the servo is Servo-ON.

3. When M2 changes from OFF to ON, servo 2 moves to the relative position 100000 PUU at 100.0r/min.

   The finish flag SM1632 is ON when the target position is reached.

Explanation of special flags (SM) and registers (SR) related to ASD-A2 CAN communication instructions

Special flags (SM) and registers (SR) related to ASD-A2 CAN communication

Flag	R/W	ID. 1	ID. 2	ID. 3	ID. 4	ID. 5	ID. 6	ID. 7	ID. 8
Enabling the specific function	R/W	The flag is set in HWCONFIG and is set to ON when PLC changes from Stop to Run.							
Initialization and communication are completed	R	SM1681							
CANopen communication error flag	R	SM1682							
Positioning is completed.	R	SM1631	SM1632	SM1633	SM1634	SM1635	SM1636	SM1637	SM1638
Stop flag	R/W	SM1641	SM1642	SM1643	SM1644	SM1645	SM1646	SM1647	SM1648
Servo-ON flag	R	SM1651	SM1652	SM1653	SM1654	SM1655	SM1656	SM1657	SM1658
The function of going back and forth is enabled. Only DDRVAC is supported.	R/W	SM1661	SM1662	SM1663	SM1664	SM1665	SM1666	SM1667	SM1668
The go-back/go-forth direction indication flag. Only DDRVAC is supported.	R	SM1671	SM1672	SM1673	SM1674	SM1675	SM1676	SM1677	SM1678
The number of the axis which has a communication error	R	SR658							
Communication error code	R	SR659							

Servo parameters of axes correspond to special registers in the CAN communication as below.

Servo parameter name (Number)		ID. 1	ID. 2	ID. 3	ID. 4	ID. 5	ID. 6	ID. 7	ID. 8
PR command（P5_07）	→	SR661	SR662	SR663	SR664	SR665	SR666	SR667	SR668
Alarm code（P0_01） (hexadecimal)	→	SR671	SR672	SR673	SR674	SR675	SR676	SR677	SR678
DO state（P0_46）	→	SR681	SR682	SR683	SR684	SR685	SR686	SR687	SR688
Servo current position（P0_09）	→	SR691 SR692	SR693 SR694	SR695 SR696	SR697 SR698	SR699 SR700	SR701 SR702	SR703 SR704	SR705 SR706
Target command position（P0-10）	→	SR711 SR712	SR713 SR714	SR715 SR716	SR717 SR718	SR719 SR720	SR721 SR722	SR723 SR724	SR725 SR726

The table of ASD-A2 CAN error code is shown below.

Error code	Cause
0x0002	The slave has no response to the SDO message.
0x0003	An error occurs in the message the slave receives. The error often occurs in the setting of COPRW instruction so that the slave can not receive the complete message.
0x0004	The slave PDO message is not received.
0x0005	An error occurs in the using of the instruction operand.
0x0006	One of the servos is being used when the INITC instruction is executed.

API	Instruction			Operand	Description
2804	D	DRVAC		S_1 , S_2 , S_3	Servo absolute position control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Station address of a servo

S_2 : Absolute target position

S_3 : Target speed

Explanation:

1. Before the instruction is executed, the execution of INITC and ASDON (Servo ON) instructions must be finished
2. The range of S_1: 1~8.
3. The range of S_2: -2147483648 ~ 2147483647. The +/ - sign means the forward / reverse direction. The target position is an absolute position.
4. Refer to DRVIC instruction for the explanation of other content and example.

Example 1:

1. When M0 changes OFF to ON, the INITC instruction makes the servos of station addresses 1~3 initialized till SM1681 is ON.
2. When M1 changes from OFF to ON, the ASDON instruction starts to make the servo of station address 2 enabled. When SM1652 is ON, it means Servo-ON.

3. When M2 changes from OFF to ON, servo 2 moves from current position to the absolute position 100,000 PUU at 100.0r/min.

   The finish flag SM1632 is ON when the target position is reached.

Example 2:

1. Add one row of program to the program in example 1. When PLC runs and SM1662 is ON, the function of going back and forth of servo 2 is enabled.

2. As the figure shows below, the servo moves from current position 20,000 to the absolute target position 100,000 after M2 is ON and then it goes back and forth between the absolute position 100,000 and 0.

   The direction indication flag SM1672 is ON when the servo goes toward the target position for the first time after Servo-ON. After that, the flag will repeat the state changing from ON to OFF.

3. The target position can be modified any time in the motion. But the timing for making the new target position valid is the next cycle of when the servo goes back and forth.

M2
SM1672
PUU
100,000
20,000
.....
Time

API	Instruction			Operand	Description
2805	D	PLSVC		S_1 , S_2	Servo speed control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
—	—	AS

Symbol:

S_1 : Station address of a servo

S_2 Target speed

Explanation:

6

1. The execution of INITC and ASDON (Servo ON) instructions must be finished before the instruction is executed.
2. The range of S_1: 1~8.
3. The range of S_2: -60000 ~ 60000. The +/- sign means the forward / reverse direction. Unit: 0.1rpm.
4. For corresponding SM and SR of axes, refer to DRVIC instruction.
5. SM1682 will be set to ON when an error occurs during the communication. Besides, SR658 will keep the number of the axis in which an error occurs and SR659 will keep the error code.

Example:

1. When M0 changes from OFF to ON, the INITC instruction starts to make the servos of station addresses 1~3 initialized till SM1681 is ON.
2. When M1 changes from OFF to ON, the ASDON instruction starts to make the servo of station address 2 enabled. When SM1652 is ON, it means Servo-ON.

3. When M2 changes from OFF to ON, servo 2 moves at 100.0r/min until M2 is OFF.

API	Instruction	Operand	Description
2806	ZRNC	S_1 , S_2 , S_3	Homing

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
—	AS	—

Symbol:

S_1 : Station address of a servo

S_2 : The 1st – segment speed

S_3 : The 2nd – segment speed

6

Explanation:

1. The execution of INITC and ASDON (Servo ON) instructions must be finished before the instruction is executed.
2. The range of S_1: 1~8.
3. The range of S_2: 1 ~ 20000. Unit: 0.1 rpm
4. The range of S_3: 1 ~ 5000. Unit: 0.1 rpm
5. When the servo returns to the home point, the corresponding finish flag of axes will be ON. Refer to DRVIC instruction for the explanation of special flags (SM) and registers (SR).
6. SM1682 will be set to ON when an error occurs during the execution of the communication. Besides, SR658 will keep the number of the axis in which an error occurs and SR659 will keep the error code.

Example:

1. When M0 changes from OFF to ON, the INITC instruction starts to make servo 1 initialized till SM1681 is ON.
2. When M1 changes from OFF to ON, the acceleration time (10000) and deceleration time (20000) of servo 1 are set by using the CASD instruction. When SM1651 is ON, it means Servo-ON.

3. When M2 changes from OFF to ON,

   The 2-byte data are written to P5-04 of servo 1. When D100=5 and the writing of data is finished, M100 is ON.

   The 4-byte data are written to P6-00 of servo 1. When D102=0 and the writing of data is finished, M101 is ON.

   The 4-byte data are written to P6-01 of servo 1. When D104=0 and the writing of data is finished, M102 is ON.

   P5-04 is used for setting the homing mode.

   P6-00 and P6-01 are the homing definition.

4. When M3 changes from OFF to ON, the homing function is enabled for servo 1.

Network 4
M2
COPRW
En
1 S1
16#002B S2
16#2504 S3
16#0000 S4
D100 S5
D M100
M100
COPRW
En
1 S1
16#0023 S2
16#2600 S3
16#0000 S4
D102 S5
D M101
M101
COPRW
En
1 S1
16#0023 S2
16#2601 S3
16#0000 S4
D104 S5
D M102
M102
M2
R
ZRST
En
M100 D1
M102 D2

Network 5
M3
ZRNC
En
1 S1
5000 S2
1000 S3
SM1631
M3
R

6

API	Instruction	Operand	Description
2807	COPRW	S_1 , S_2 , S_3	Writing and reading the CANopen communication data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●								
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
S_5		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
—	AS	—

Symbol:

S_1 : Station address of a servo

S_2 : Request code

S_3 : Index

S_4 : Sub-index

S_5 : Read/write device

D : Communication completion flag

Explanation:

1. The range of **S_1**: 1~127. If the value is out of the range (<1 or >127), the minimum or maximum value will be automatically seen as the value of **S_1** by the instruction.

2. **S_2** can only specify four types of request codes as shown in the following table.

H23	Writing the 4-byte data
H2B	Writing the 2-byte data

H2F	Writing the 1-byte data
H40	Reading the data. The data length is contained in the SDO response message.

3. For **S_3** and **S_4**, refer to the object dictionary in Delta servo operation manual.

4. It is suggested that COPRW instruction should be executed after the execution of INITC instruction is finished in case that the written parameter values may be overwritten by the INITC instruction. COPRW instruction is used for the CANopen SDO communication. If the station address of a slave exceeds the range of 1~8, COPRW instruction can be used for the setting, reading and writing of slave-related parameters.

5. SM1682 will be set to ON when an error occurs during the communication. Besides, SR658 will keep the number of the axis in which an error occurs and SR659 will keep the error code.

   Note: When COPRW instruction is used, users must edit the process of dealing with the communication errors in order to avoid resulting in incorrect communication process incurred by unexpected communication errors.

6. The timing diagram of COPRW instruction.

   ① When COPRW instruction is enabled for the first time, the instruction will send the command code immediately if no other CANopen communication uses it.

   ② The instruction is sending the command code.

   ③ The sending is finished and the finish flag is set to ON.

   ④ Users modify the next data to be sent out. Then the next command code is sent out immediately after the finish flag is cleared to OFF.

   ⑤The sending is finished and the COPRW instruction is disabled.

7. COPRW instruction only supports the following “Delta ASD-A2 control” mode for the hardware configuration setting.

Most of the parameters in Delta ASDA-A2 are displayed in the decimal format. Users can convert the parameters into index addresses by using the formula: the index address of PX-YY=0x2000 +（X << 8）+ YY

Example: P2-10 = 0x2000 +（0x0002 << 8）+ 0x000A = 0x220A

P5-04 = 0x2000 +（0x0005 << 8）+ 0x0004 = 0x2504

P1-44 = 0x2000 +（0x0001 << 8）+ 0x002C = 0x212C

Example:

1. When M0 changes from OFF to ON, the INITC instruction starts to make the servo of station addresses 1~3 initialized till SM1681 is ON.

2. When M1 changes from OFF to ON, PLC writes the 2-byte data in D100 to P5-04 of servo 3 by using the COPRW instruction. When the writing is finished, M100 is ON.

3. When M2 changes from OFF to ON, PLC reads the value of P4-07 of servo 3 and the read value is stored in D106. When the reading is finished, M106 is ON.

Network 2
M1
COPRW
En
3
S1
D
M100
16#002B
S2
16#2504
S3
16#0000
S4
D100
S5
M100
M1
R
Network 3
M2
COPRW
En
3
S1
D
M106
16#0040
S2
16#2407
S3
16#0000
S4
D106
S5
M106
M2
R

7

Chapter 7 Troubleshooting

Table of Contents

7.1 Troubleshooting 7-2

7.1.1 Basic troubleshooting steps 7-2

7.1.2 Clear the States of Errors 7-2

7.1.3 Troubleshooting SOP 7-3

7.1.4 System Log 7-4

7.2 Troubleshooting for CPU Modules 7-5

7.2.1 ERROR LED Indicator's Being ON 7-5

7.2.2 ERROR LED Indicator's Blinking Every 0.5 Seconds 7-5

7.2.3 ERROR LED Indicator's Rapid Blinking Every 0.2 Seconds 7-7

7.2.4 ERROR LED Indicator's Slow Blinking Every 3 Seconds and Lighting up for 1 Second 7-7

7.2.5 BAT. LOW LED Indicator's Being ON 7-7

7.2.6 BAT. LOW LED Indicator's Blinking Every 0.5 Seconds 7-7

7.2.7 The LED Indicators of RUN and ERROR are Blinking Every 0.5 Seconds Simultaneously 7-7

7.2.8 The LED Indicators of RUN and ERROR are Blinking One After Another Every 0.5 Seconds 7-8

7.2.9 Other Errors (Without LED Indicators) 7-8

7.3 Troubleshooting for I/O Modules 7-14

7.3.1 Troubleshooting for Analog Modules (AD/DA/XA) and Temperature Modules (RTD/TC) 7-14

7.3.2 Troubleshooting for Load Cell Module AS02LC 7-15

7.3.3 Troubleshooting for Module AS00SCM as a Communication Module 7-15

7.3.4 Troubleshooting for Module AS00SCM as a Remote Module 7-17

7.4 Error Codes and LED Indicators for CPU Modules 7-18

7.4.1 Error Codes and LED Indicators for CPU Modules 7-18

7.4.2 Error Codes and LED Indicators for Analog/Temperature Modules 7-23

7.4.3 Error Codes and LED Indicators for Load Cell Module AS02LC 7-24

7.4.4 Error Codes and LED Indicators for Module AS00SCM as a Communication Module 7-25

7.4.5 Error Codes and LED Indicators for Module AS00SCM as a Remote Module 7-25

7.1 Troubleshooting

7.1.1 Basic troubleshooting steps

This chapter includes kinds of possible errors occurred during operation and the causes of them and what actions should be taken to correct the errors.

(1) Check the followings:

- PLC should be operated in a safe environment (the environmental, electronical, vibrational safeties should be considered.)
- Power supply should be correctly connected and supply power to the PLC.
- Installations of modules, terminals and cables are secured.
- All the LED indicators are shown correct.
- All the switches are correctly set.

(2) Check the followings for the AS series to operate:

- Switch the RUN/STOP
- Check the settings for the AS series to RUN/STOP
- Check and eliminate the errors from the external devices
- Use the System Log function of the ISPSoft to check the system operation and the logs.

(3) Identify the source of the possible causes:

- AS series or external device
- CPU modules or the extension modules
- Setting parameters or programs

7.1.2 Clear the States of Errors

Use the following methods to clear the status of errors when errors occurred. But if the source of error is not fixed, the system will still show errors.

(1) Switch the state of the CPU model to STOP and then to RUN.

(2) Turn off the CPU and turn on again.

(3) Use the ISPSoft to clear the error logs.

(4) Reset the CPU and set the settings to defaults and download the project again to operate.

7.1.3 Troubleshooting SOP

7.1.4 System Log

If ISPSoft is connected to an AS series normally, users can view the actions and the errors occur in the AS series after they click **System Log** on the **PL**C menu. Up to 20 piece of error logs can be stored in the CPU. After the 20 sets are stored, the 1st log will be replaced with the 21st if there are new logs coming in; the old logs will be replaced with the new ones accordingly. When the memory card is installed in the CPU module, 20 pieces of the old logs will be backed up in the memory card and up to 10000 logs can be recorded. If the stored log exceeds the limitation of 1000, the oldest 20 logs will be replaced with the newest 20 logs in the memory card.

(1) Click **System Log** on the **PL**C menu. PLC> System Log.

(2) After users click **System Log** on the **PL**C menu, the **System Log** window will appear. After users click **Clear Log**, the error log in the window and the error log in the CPU module will be cleared, and the CPU module will be reset.

- Group No.: The number 1 indicates the error occurred in the CPU module or the right-side module 1. The number 2~16 indicates the error occurred in the remote module 1~15.
- Module No.: The number 0 indicates the error occurred in the CPU module or the remote module. The number 1~32 indicates the error occurred in the right-side module of the CPU module / remote module. (The number 1 represents the closest module to the CPU module or the remote module; this number increases from the closest to the furthest to the CPU module or the remote module.) Note: Up to 8 extension modules can be connected to the right-side of the remote module.
- PLC/Module name: Model names of the CPU modules, remote modules and the extension modules.

- Error Code: Error codes of the error log.
- Date & Time: The error occurred date and time. The most recent occurred error will be listed on the top.
- The last column shows the relative descriptions for the error.

7.2 Troubleshooting for CPU Modules

Check the LED indicators and the error codes from the CPU module and refer to the following table for troubleshooting. V in the Log column indicates the error will be recorded in the log; X in the Log column indicates the error will not be recorded in the log; H in the Log column indicates whether to record the error in the log or not can be set in HWCONFIG.

7.2.1 ERROR LED Indicator's Being ON

Error Code (16#)	Description	Solution	Flag	Log
000A	Scan timeout	1. Check the setting of the watchdog timer in HWCONFIG. 2. Check whether the program causes the long scan time	SM8	V

7.2.2 ERROR LED Indicator's Blinking Every 0.5 Seconds

Error Code (16#)	Description	Solution	Flag	Log
000C	The program in the PLC is damaged.	Download the program again.	SM9	V
0010	The access to the memory in the CPU is denied.	Please contact the factory.	SM9	V
002E	The access to the external memory of the CPU is denied.	Please contact the factory.	SM9	V
002F	PLC programs are not consistent with the system logs.	Download the program again.	SM34	V
0070	The actual arrangement of the function cards is not consistent with the settings.	Check whether the settings in HWCONFIG are consistent with the actual arrangement of the function cards.	SM10	V
0102	The interrupt number exceeds the range.	Check the program, compile the program again, and download the program again.	SM5	X
0202	The MC instruction exceeds the range.	Check the program, compile the program again, and download the program again.	SM5	X
0302	The MCR instruction exceeds the range.	Check the program, compile the program again, and download the program again.	SM5	X
0D03	The operands used in DHSCS are not used properly.	Check the program, compile the program again, and download the program again.	SM5	X
0E05	The operands HCXXX used in DCNT are not used properly.	Check the program, compile the program again, and download the program again.	SM5	X

Error Code (16#)	Description	Solution	Flag	Log
1300 ~ 130F	Errors occurred in the remote modules	Refer to section 12.3.4 for more information on the error codes of the remote modules.	SM30	V
1402	The actual arrangement of the I/O modules is not consistent with the settings.	Check whether the settings in HWCONFIG are consistent with the actual arrangement of the I/O modules.	SM10	V
140B	The communication modules exceed the limit of 4.	Check the total number of the communication modules.	SM10	V
140D	The extension modules exceed the limit of 32.	Check the total number of the extension modules.	SM10	V
140E	The remote modules exceed the limit of 8 on the right side of the CPU module.	Check the total number of the remote modules on the right side of the CPU module.	SM30	V
1600	The ID of the extension module exceeds the range.	1. Make sure the module is well-connected to the CPU module and turn-on the modules again. 2. If the error still occurs, please contact the factory.	SM10	V
1601	The ID of the extension module cannot be set.	1. Make sure the module is well-connected to the CPU module and turn-on the modules again. 2. If the error still occurs, please contact the factory.	SM10	V
1602	The ID of the extension module is duplicated.	1. Make sure the module is well-connected to the CPU module and turn-on the modules again. 2. If the error still occurs, please contact the factory.	SM10	V
1603	The extension module cannot be operated.	1. Make sure the module is well-connected to the CPU module and turn-on the modules again. 2. If the error still occurs, please contact the factory.	SM10	V
1604	Extension module communication timeout	1. Make sure the module is well-connected to the CPU module and turn-on the modules again. 2. If the error still occurs, please contact the factory.	SM10	V
1605	Hardware failure	Please contact the factory.	SM10	V
1606	Errors on the function card of the communication module	Make sure the function card is well-connected to the CPU module and turn-on the modules again.	SM10	V
1607	The external voltage is abnormal.	Check whether the external 24 V power supply to the module is normal.	SM10	V
1608	The Internal factory calibration or the CJC is abnormal.	Please contact the factory.	SM10	V
1609 ~ 160F	Reserved (Error codes for the extension modules)			
200A	Invalid instruction	Check the program, compile the program again, and download the program again.	SM5	V
6010	The number of the MODBUS TCP connections exceeds the range.	Check if the number of the superior devices exceeds the limit of 32.	SM 1092	V
6011	The number of the EtherNet/IP connections exceeds the range.	Check if the number of the connections exceeds the range of 16.	SM 1093	V

7.2.3 ERROR LED Indicator's Rapid Blinking Every 0.2 Seconds

This happens when the power supply 24VDC of the CPU module is disconnecting or the power supply is not sufficient, not stable or abnormal so that it cannot be operated.

Error Code (16#)	Description	Solution	Flag	Log
002A	The external voltage is abnormal.	Check whether the external 24 V power supply to the module is normal.	SM7	V

7.2.4 ERROR LED Indicator's Slow Blinking Every 3 Seconds and Lighting up for 1 Second

Error Code (16#)	Description	Solution	Flag	Log
1500	Connection lost in the remote modules	Please check the network connection cable.	SM30	V
1502 ~ 150F	Errors occurred in the remote modules	Refer to section 12.3.4 for more information on the error codes of the remote modules.	SM30	V
1800 ~ 180F	Errors occurred in the extension modules	Refer to section 12.3 for more information on the error codes of the extension modules.	SM10	V
1900 ~ 191C	Heartbeat errors occurred in the slave of Delta ASD-A2 control.	1. Check the CANopen connection cable. 2. Check if the specific salve is working properly. Note: The last 2 digits of the error code represent the ID number of the slave (hexadecimal should convert to decimal).	-	V

7.2.5 BAT. LOW LED Indicator's Being ON

This happens when there is no battery (CR1620) or the power is low. Users can set this option off in the HWCONFIG > CPU > Device Setting > Show Battery Low Voltage Error CPU, when users don't need the function of RTC to keep track of the current time. (Default is enabled.)

Error Code (16#)	Description	Solution	Flag	Log
0027	Battery Low	Change battery or set this option off	SM219	X

7.2.6 BAT. LOW LED Indicator's Blinking Every 0.5 Seconds

This happens when RTC cannot keep track of the current time.

Error Code (16#)	Description	Solution	Flag	Log
0026	RTC cannot keep track of the current time	Please contact the factory.	SM218	V

7.2.7 The LED Indicators of RUN and ERROR are Blinking Every 0.5 Seconds Simultaneously

This happens when the firmware of the CPU module is being upgraded. If this happens once the power is supplied to the

CPU module, it means errors occurred during the previous firmware upgrade. Users need to upgrade the firmware again or contact your point of purchase.

7.2.8 The LED Indicators of RUN and ERROR are Blinking One After Another Every 0.5 Seconds.

This happens when the memory card of the CPU module is backing up / restoring / or saving.

7.2.9 Other Errors (Without LED Indicators)

Error Code (16#)	Description	Solution	Flag	Log
0011	The PLC ID is incorrect.	Please check the PLC ID.	SM34	V
0012	The PLC password is incorrect.	Please check the PLC password.	SM34	V
002D	The PLC maximum password attempts exceeded.	Reset the CPU module or restore the CPU module to its factory settings.	SM34	V
0050	The memories in the latched special auxiliary relays are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, please contact the factory.	SM6	V
0051	The latched special data registers are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, please contact the factory.	SM6	V
0052	The memories in the latched auxiliary relays are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, please contact the factory.	SM6	V
0054	The latched counters are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, please contact the factory.	SM6	V
0055	The latched 32-bit counters are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, please contact the factory.	SM6	V
0056	The latched special auxiliary relay is abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, please contact the factory.	SM6	V
0059	The latched data registers are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, contact the factory.	SM6	V
005D	The CPU module does not detect a memory card.	Check whether a memory card is inserted into the CPU module correctly.	SM453	V
005E	The memory card is initialized incorrectly.	Check whether the memory card is broken.	SM453	V
0063	An error occurs when data is written to the memory card.	Check whether the file path is correct, or whether the memory card breaks down.	SM453	V

0064	A file in the memory card cannot be read.	Check whether the file path is correct, or whether the file is damaged.	SM453	V
1950	The initialization of Delta ASD-A2 control has not yet been completed, the CANopen instructions cannot be executed.	1. Check the CANopen connection cable. 2. Check if the specific salve is working properly. 3. If nothing is wrong, initialize Delta ASD-A2 again.	-	V
2001	Without using the FCOMP card or not in the right mode for the ASDA-A2 while using the CANopen communication instruction.	Make sure to use the FCOMP card in the function card 2 and check if the operation mode is correct.	SM0	V
2003	The device used in the program exceeds the device range.	Check the program, compile the program again, and download the program again.	SM0	V
200B	The operand n or the other constant operands K/H exceed the range.	Check the program, compile the program again, and download the program again.	SM0	V
200C	The operands overlap.	Check the program, compile the program again, and download the program again.	SM0	V
200D	The binary to the binary-coded decimal conversion is incorrect.	Check the program, compile the program again, and download the program again.	SM0	V
200E	The string does not end with 00.	Check the program, compile the program again, and download the program again.	SM0	V
2012	Incorrect division operation	Check the program, compile the program again, and download the program again.	SM0	V
2013	The value exceeds the range of values which can be represented by the floating-point numbers.	Check the program, compile the program again, and download the program again.	SM0	V
2014	The task designated by TKON/YKOFF is incorrect, or exceeds the range.	Check the program, compile the program again, and download the program again.	SM0	V
2017	The instruction BREAK is written outside of the FOR-NEXT.	Check the program, compile the program again, and download the program again.	SM0	V
2027	No such position planning table number or the format is incorrect.	1. Check the program, compile the program again, and download the program again. 2. Check the settings of the position planning table.	SM0	V
2028	The high speed output instruction is being executed. Only one instruction can be executed at a time.	Refer to SR28 for the record of the axis number and rearrange the output control procedures.	-	V
6004	The IP address filter is set incorrectly.	Set the Ethernet parameter for the CPU module in HWCONFIG again.	SM1108	X
600D	RJ45 port is not connected.	Check the connection.	SM1100	X
6012	There are devices using the same IP address.	1. Check if there are devices using the same IP address. 2. Check if there is more than 1 DHCP or BOOTP server on the network.	SM1101	V
6100	The email connection is busy.	Retry the email connection later. (This error does not cause the PLC to stop running. Users can perform the corresponding solution by means of the related flag in the program.)	SM1113	X
6103	The trigger attachment mode in the email is set incorrectly.	Set up the trigger attachment mode in HWCONFIG > CPU Module > Device Setting > Options > Ethernet Port Advanced > Email > Trigger Setting > Trigger Attachment Mode.	SM1113	X

6104	The attachment in the email does not exist.	Check whether the attachment exists in the memory card.	SM1113	X
6105	The attachment in the email is oversized.	Check the size of the attachment. If the size is over 2 MB, the file cannot be sent as an attachment.	SM1113	X
6106	There is an SMTP server response timeout.	Check for the correct address and set up the SMTP server in HWCONFIG > CPU Module > Device Setting > Options > Ethernet Port Advanced > Email again.	SM1113	X
6107	There is an SMTP server response timeout.	1. Check whether the status of the SMTP server is normal. 2. Retry the sending of the email later. (This error does not cause the PLC to stop running. Users can perform the corresponding solution by means of the related flag in the program.)	SM1113	X
6108	SMTP verification failed	Check for the correct ID/Password and set up in HWCONFIG > CPU Module > Device Setting > Options > Ethernet Port Advanced > Email again.	SM1113	X
6200	The remote communication IP address set in the TCP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X
6201	The local communication port set in the TCP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X
6202	The remote communication port set in the TCP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X
6203	The device from which the data is sent in the TCP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X
6206	The device which receives the data in the TCP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X
6208	The data which is received through the TCP socket exceeds the device range.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X

6209	The remote communication IP address set in the UDP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
620A	The local communication port set in the UDP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
620C	The device from which the data is sent in the UDP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
620F	The device which receives the data in the UDP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
6210	The data which is received through the UDP socket exceeds the device range.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
6212	There is no response from the remote device after the timeout period.	Make sure that the remote device is connected.	-	X
6213	The data received exceeds the limit.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
6214	The remote device refuses the connection.	Make sure that the remote device operates normally.	-	X
6215	The socket is not opened.	Check whether operational sequence in the program is correct.	-	X
6217	The socket is opened.	Check whether operational sequence in the program is correct.	-	X
6218	The data has been sent through the socket.	Check whether operational sequence in the program is correct.	-	X
6219	The data has been received through the socket.	Check whether operational sequence in the program is correct.	-	X
621A	The socket is closed.	Check whether operational sequence in the program is correct.	-	X
7011	The device communication function code in COM1 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H

7012	The device communication address used in COM1 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7013	The device used in COM1 exceeds the device range.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7014	The device length of the communication data in COM1 exceeds the limit.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7017	The device checksum for the communication serial port of COM1 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7021	The device communication function code in COM2 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7022	The device communication address used in COM2 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7023	The device used in COM2 exceeds the device range.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7024	The device length of the communication data in COM2 exceeds the limit.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7027	The device checksum for the communication serial port of COM2 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7031	The device communication function code in the Ethernet is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7032	The device communication address used in the Ethernet is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7033	The device used in the Ethernet exceeds the device range.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7034	The device length of the communication data in the Ethernet exceeds the limit.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7037	The device checksum for the communication serial port of the Ethernet is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7041	The device communication function code in the USB is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7042	The device communication address used in the USB is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7043	The device used in the USB exceeds the device range.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H

7044	The device length of the communication data in the USB exceeds the limit.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7047	The device checksum for the communication serial port of the USB is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
70B1	The device communication function code in the function card 1 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
70B2	The device communication address used in the function card 1 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
70B3	The device used in the function card 1 exceeds the device range.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
70B4	The device length of the communication data in the function card 1 exceeds the limit.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
70B7	The device checksum for the communication serial port of the function card 1 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
70C1	The device communication function code in the function card 2 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
70C2	The device communication address used in the function card 2 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
70C3	The device used in the function card 2 exceeds the device range.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
70C4	The device length of the communication data in the function card 2 exceeds the limit.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
70C7	The device checksum for the communication serial port of the function card 2 is incorrect.	1. Check the communication setting in the master, and the communication setting in slave. 2. Check the communication cable.	-	H
7203	Invalid communication function code	Refer to the function codes defined by the communication protocols	-	H
8105	The contents of the program downloaded are incorrect. The program syntax is incorrect.	Download the program and the parameters again.	-	H
8106	The contents of the program downloaded are incorrect. The length of the execution code exceeds the limit.	Download the program and the parameters again.	-	H
8107	The contents of the program downloaded are incorrect. The length of the source code exceeds the limit.	Download the program and the parameters again.	-	H

7.3 Troubleshooting for I/O Modules

- **Introduction of modules**

Digital I/O modules, analog I/O modules, temperature measurement modules, load cell modules, and network modules can be installed in an AS series system. There are 2 types of error codes, for errors and for warning. The CPU module and its modules will stop operating when errors occurred. The CPU modules and its modules will not stop operating when warnings triggered.

7.3.1 Troubleshooting for Analog Modules (AD/DA/XA) and Temperature Modules (RTD/TC)

7.3.1.1 ERROR LED Indicator's Being ON

The Following errors will be specified as warnings. Users need to set up in HWCONFIG to have them shown as errors when the following errors occurred.

Error Code	Description	Solution
16#1605	Hardware failure	Please contact the factory.
16#1607	The external voltage is abnormal.	Check the power supply.
16#1608	The factory calibration or the CJC is abnormal.	Please contact the factory.

7.3.1.2 ERROR LED Indicator's Blinking Every 0.5 Seconds

The following errors are specified as warnings to ensure the CPU module can still run even when the warnings are triggered by its AIO modules. Users can set up in HWCONFIG to have them shown as errors when the first 4 errors occurred.

Error Code	Description	Solution
16#1801	The external voltage is abnormal.	Check the power supply.
16#1802	Hardware failure	Please contact the factory.
16#1804	The factory calibration is abnormal.	Please contact the factory.
16#1807	The CJC is abnormal.	Please contact the factory.
16#1808	The signal received by channel 1 exceeds the range of analog inputs (temperature).	Check the signal received by channel 1
16#1809	The signal received by channel 2 exceeds the range of analog inputs (temperature).	Check the signal received by channel 2
16#180A	The signal received by channel 3 exceeds the range of analog inputs (temperature).	Check the signal received by channel 3
16#180B	The signal received by channel 4 exceeds the range of analog inputs (temperature).	Check the signal received by channel 4

7.3.2 Troubleshooting for Load Cell Module AS02LC

7.3.2.1 ERROR LED Indicator's Being ON

Users can set up in HWCONFIG to have them shown as errors when the following errors occurred.

Error Code	Description	Solution
16#1605	Hardware failure (e.g. the diver board)	Please contact the factory.
16#1607	The external voltage is abnormal.	Check the power supply.

7.3.2.2 ERROR LED Indicator's Blinking Every 0.5 Seconds

The following errors are specified as warnings to ensure the CPU module can still run even when the warnings are triggered by its AIO modules. Users can set up in HWCONFIG to have them shown as errors when the first 3 errors occurred.

Error Code	Description	Solution
16#1801	The external voltage is abnormal.	Check the power supply.
16#1802	Hardware failure	Please contact the factory.
16#1807	Diver board failure	Please contact the factory.
16#1808	The signal received by channel 1 exceeds the range of analog inputs or the SEN voltage is abnormal.	Check the signal received by channel 1 and the cable connections.
16#1809	The signal received by channel 1 exceeds the weight limit.	Check the value inputted in channel 1 and the setting of the maximum weight.
16#180A	The factory calibration in channel 1 is incorrect.	Check the weight calibration in channel 1.
16#180B	The signal received by channel 2 exceeds the range of analog inputs or the SEN voltage is abnormal.	Check the signal received by channel 2 and the cable connections.
16#180C	The signal received by channel 2 exceeds the weight limit.	Check the value inputted in channel 2 and the setting of the maximum weight.
16#180D	The factory calibration in channel 2 is incorrect.	Check the weight calibration in channel 1.

7.3.3 Troubleshooting for Module AS00SCM as a Communication Module

7.3.3.1 ERROR LED Indicator's Being ON

The following error codes are for users to identify possible errors occurred when the AS00SCM module is installed on the right side of the CPU module and acts as a communication module.

Error Code	Description	Solution
16#1605	Hardware failure	1. Check if the module is securely installed. 2. Change and install a new AS00SCM or contact the factory.

Error Code	Description	Solution
16#1606	The setting of the function card is incorrect.	1. Check if the function card is securely installed. 2. Change and install a new function card or contact the factory. 3. Check if the setting in HWCONFIG is consistent with the actual setting in the function card. 4. Change and install a new AS00SCM or contact the factory.

7.3.3.2 ERROR LED Indicator's Blinking Every 0.5 Seconds

The following error codes are for users to identify possible errors occurred when the AS00SCM module is installed on the right side of the CPU module and acts as a communication module.

Error Code	Description	Solution
16#1802	Incorrect parameters	Check the parameter in HWCONFIG, and the parameter. Download the parameter again.
16#1803	Communication timeout	1. Check whether the communication cable is connected well. 2. Check if the station number and the communication format are correctly set. 3. Check if the connection with the function card is working fine.
16#1804	The setting of the UD Link is incorrect.	1. Check the settings of the UD Link. 2. Check the settings to trigger warnings in the PLC.

The following error codes can only be viewed via SCMSoft; when the following errors occurred, they will not be shown on the LED indicators and the system will not send the error messages to the CPU module.

Error Code	Description	Solution
16#0107	The settings in HWCONFIG and actual manual settings are not consistent for the function card 1.	Check the settings in HWCONFIG and actual manual settings for the function card 1.
16#0108	The settings in HWCONFIG and actual manual settings are not consistent for the function card 2.	Check the settings in HWCONFIG and actual manual settings for the function card 2.
16#0201	Incorrect parameters	Check the parameter in HWCONFIG, and the parameter. Download the parameter again.
16#0301	Function card 1 communication timeout	1. Check if the station number and the communication format are correctly set. 2. Check if the connection with the function card is working fine.
16#0302	Function card 2 communication timeout	1. Check if the station number and the communication format are correctly set. 2. Check if the connection with the function card is working fine.
16#0400	Invalid UD Link Group ID for the function card 1	1. Check the settings of the UD Link. 2. Check the settings to trigger warnings in the PLC.
16#0401	Invalid UD Link Group ID for the function card 2	1. Check the settings of the UD Link. 2. Check the settings to trigger warnings in the PLC.

Error Code	Description	Solution
16#0402	Invalid UD Link Command for the function card 1	1. Check the settings of the UD Link. 2. Check the settings to trigger warnings in the PLC.
16#0403	Invalid UD Link Command for the function card 1	1. Check the settings of the UD Link. 2. Check the settings to trigger warnings in the PLC.

7.3.4 Troubleshooting for Module AS00SCM as a Remote Module

Errors from the remote modules are regarded as warnings for AS CPU modules. The LED indicator of the CPU module will blink and the CPU module can still operate. Users can use the flag SM30 to work with the programs in the PLC to manage the ways to present the errors from the remote modules.

7.3.4.1 Error LED Indicator's Being ON

Error codes for the error type

Error Code	Description	Solution
16#1301	Hardware failure	1. Check if the module is securely installed. 2. Change and install a new AS00SCM or contact the factory.
16#1302	The setting of the function card is incorrect.	1. Check if the function card is securely installed with the AS-FCOPM card. 2. Change and install a new function card or contact the factory. 3. Check if the setting in HWCONFIG is consistent with the actual setting in the function card. 4. Change and install a new AS00SCM or contact the factory.

7.3.4.2 ERROR LED Indicator's Blinking Every 0.5 Seconds

Error codes for the warning type

Error Code	Description	Solution
16#1502	Incorrect parameters	Check the parameter in HWCONFIG, and the parameter. Download the parameter again.
16#1503	Extension module communication timeout	Make sure the module is well-connected to the CPU module and turn-on the modules again.

7.3.4.3 ERROR LED Indicator's Blinking Every 0.2 Seconds

This happens when the power supply of 24VDC for the remote module is not sufficient. Please check the power supply. If the power supply is normal, remove the extension module from the CPU module and then check if the SCM remote module is out of order. The error codes below are of the warning types.

Error Code	Description	Solution
16#1303	24VDC power supply is not sufficient and then is recovered from a low-voltage less than 10ms situation.	Check whether the 24 V power supply to the module is normal.

7.4 Error Codes and LED Indicators for CPU Modules

A. Columns

a. Error code: If the error occurs in the system, the error code is generated.

b. Description: The description of the error

c. CPU status: If the error occurs, the CPU stops running, keeps running, or in the status defined by users.

- Stop: The CPU stops running when the error occurs.
- Continue: The CPU keeps running when the error occurs.

d. LED indicator status: If the error occurs, the LED indicator is ON, OFF, or blinks.

- ERROR: The system error

- **Descriptions**

Module Type	LED indicator	Descriptions
CPU	Error LED	There are 5 types of error indicator status for the errors of the CPU module, including LED indicator ON, OFF, blinking fast, blinking normally, and blinking slowly. When the LED indicator is ON, blinking fast/normally, users need to clear the problems first in order to run the CPU module. When the LED indicator is blinking slowly, indicating a warning type of error codes, it does not require immediate action. It is suggested to clear the problems when the module is power-off. Error type: ON: A serious error occurs in the module. Blinking fast (every 0.2 seconds): unstable power supply or hardware failure Blinking normally (every 0.5 second): system program errors or system cannot run. Warning type: Blinking slowly (every 1 second and stop for 3 seconds): a warning is triggered, but the system can still run. OFF: a warning is triggered, but the system can still run. Users can modify the rules of how a warning is triggered or use the SM/SR to show the warnings.

7.4.1 Error Codes and LED Indicators for CPU Modules

Note: refer to the section 12.3 for the status descriptions of the Error LED indicators.

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
000A	Scan timeout	Stop	V				
000C	The program in the PLC is damaged.	Stop			V		
0010	The access to the memory in the CPU is denied.	Stop			V		
0011	The PLC ID is incorrect.	Continue					V

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
0012	The PLC password is incorrect.	Continue					V
0026	RTC cannot keep track of the current time (The battery LED is blinking.)	Continue					
0027	Battery low (The battery LED is ON.)	Continue					
002A	24VDC power supply is not sufficient and then is recovered from a low-voltage less than 10ms situation.	Continue		V			
002D	The PLC maximum password attempts exceeded.	Continue					V
002E	The access to the external memory of the CPU is denied.	Stop			V		
002F	PLC programs are not consistent with the system logs.	Stop			V		
0050	The memories in the latched special auxiliary relays are abnormal.	Continue					V
0051	The latched special data registers are abnormal.	Continue					V
0052	The memories in the latched auxiliary relays are abnormal.	Continue					V
0054	The latched counters are abnormal.	Continue					V
0055	The latched 32-bit counters are abnormal.	Continue					V
0056	The latched special auxiliary relay is abnormal.	Continue					V
0059	The latched data registers are abnormal.	Continue					V
005D	The CPU module does not detect a memory card.	Continue					V
005E	The memory card is initialized incorrectly.	Continue					V
0063	An error occurs when data is written to the memory card.	Continue					V
0064	A file in the memory card cannot be read.	Continue					V
0070	The actual arrangement of the function cards is not consistent with the settings.	Stop			V		
0102	The interrupt number exceeds the range.	Stop			V		
0202	The MC instruction exceeds the range.	Stop			V		
0302	The MCR instruction exceeds the range.	Stop			V		
0D03	The operands used in DHSCS are not used properly.	Stop			V		
0E05	The operands HCXXX used in DCNT are not used properly.	Stop			V		
1300 ~ 130F	Errors occurred in the remote modules	Continue				V	
1402	The actual arrangement of the I/O modules is not consistent with the settings.	Stop			V		
140B	The communication modules exceed the limit of 4.	Stop			V		
140D	The extension modules exceed the limit of 32.	Stop			V		
140E	The remote modules exceed the limit of 8 on the right side of the CPU module.	Stop			V		

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
1500	Connection lost in the remote modules	Continue				V	
1502 ~ 150F	Errors occurred in the remote modules	Continue				V	
1600	The ID of the extension module exceeds the range.	Stop			V		
1601	The ID of the extension module cannot be set.	Stop			V		
1602	The ID of the extension module is duplicated.	Stop			V		
1603	The extension module cannot be operated.	Stop			V		
1604	Extension module communication timeout	Stop			V		
1605	Hardware failure	Stop			V		
1606	Errors on the function card of the communication module	Stop			V		
1607	The external voltage is abnormal.	Stop			V		
1608	The Internal factory calibration or the CJC is abnormal.	Stop			V		
1609 ~ 160F	Reserved (Error codes for the extension modules)	Stop			V		
1800 ~ 180F	Errors occurred in the extension modules	Continue				V	
1900 ~ 191C	Heartbeat errors occurred in the slave of Delta ASD-A2 control.	Continue				V	
1950	The initialization of Delta ASD-A2 control has not yet been completed, the CANopen instructions cannot be executed.	Continue					V
2001	Without using the FCOMP card or not in the right mode for the ASDA-A2 while using the CANopen communication instruction.	Continue					V
2003	The device used in the program exceeds the device range.	Continue					V
200A	Invalid instruction	Stop			V		
200B	The operand n or the other constant operands K/H exceed the range.	Continue					V
200C	The operands overlap.	Continue					V
200D	The binary to the binary-coded decimal conversion is incorrect.	Continue					V
200E	The string does not end with 00.	Continue					V
2012	Incorrect division operation	Continue					V
2013	The value exceeds the range of values which can be represented by the floating-point numbers.	Continue					V

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
2014	The task designated by TKON/YKOFF is incorrect, or exceeds the range.	Continue					V
2017	The instruction BREAK is written outside of the FOR-NEXT.	Continue					V
2027	No such position planning table number or the format is incorrect.	Continue					V
2028	The high speed output instruction is being executed. Only one instruction can be executed at a time.	Continue					V
6004	The IP address filter is set incorrectly.	Continue					V
600D	RJ45 port is not connected.	Continue					V
6010	The number of the MODBUS TCP connections exceeds the range.	Continue			V		
6011	The number of the EtherNet/IP connections exceeds the range.	Continue			V		
6012	There are devices using the same IP address.	Continue					V
6100	The email connection is busy.	Continue					V
6103	The trigger attachment mode in the email is set incorrectly.	Continue					V
6104	The attachment in the email does not exist.	Continue					V
6105	The attachment in the email is oversized.	Continue					V
6106	There is an SMTP server response timeout.	Continue					V
6107	There is an SMTP server response timeout.	Continue					V
6108	SMTP verification failed	Continue					V
6200	The remote communication IP address set in the TCP socket function is illegal.	Continue					V
6201	The local communication port set in the TCP socket function is illegal.	Continue					V
6202	The remote communication port set in the TCP socket function is illegal.	Continue					V
6203	The device from which the data is sent in the TCP socket function is illegal.	Continue					V
6206	The device which receives the data in the TCP socket function is illegal.	Continue					V
6208	The data which is received through the TCP socket exceeds the device range.	Continue					V
6209	The remote communication IP address set in the UDP socket function is illegal.	Continue					V
620A	The local communication port set in the UDP socket function is illegal.	Continue					V
620C	The device from which the data is sent in the UDP socket function is illegal.	Continue					V
620F	The device which receives the data in the UDP socket function is illegal.	Continue					V

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
6210	The data which is received through the UDP socket exceeds the device range.	Continue					V
6212	There is no response from the remote device after the timeout period.	Continue					V
6213	The data received exceeds the limit.	Continue					V
6214	The remote device refuses the connection.	Continue					V
6215	The socket is not opened.	Continue					V
6217	The socket is opened.	Continue					V
6218	The data has been sent through the socket.	Continue					V
6219	The data has been received through the socket.	Continue					V
621A	The socket is closed.	Continue					V
7011	The device communication function code in COM1 is incorrect.	Continue					V
7012	The device communication address used in COM1 is incorrect.	Continue					V
7013	The device used in COM1 exceeds the device range.	Continue					V
7014	The device length of the communication data in COM1 exceeds the limit.	Continue					V
7017	The device checksum for the communication serial port of COM1 is incorrect.	Continue					V
7021	The device communication function code in COM2 is incorrect.	Continue					V
7022	The device communication address used in COM2 is incorrect.	Continue					V
7023	The device used in COM2 exceeds the device range.	Continue					V
7024	The device length of the communication data in COM2 exceeds the limit.	Continue					V
7027	The device checksum for the communication serial port of COM2 is incorrect.	Continue					V
7031	The device communication function code in the Ethernet is incorrect.	Continue					V
7032	The device communication address used in the Ethernet is incorrect.	Continue					V
7033	The device used in the Ethernet exceeds the device range.	Continue					V
7034	The device length of the communication data in the Ethernet exceeds the limit.	Continue					V
7037	The device checksum for the communication serial port of the Ethernet is incorrect.	Continue					V
7041	The device communication function code in the USB is incorrect.	Continue					V
7042	The device communication address used in the USB is incorrect.	Continue					V

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
7043	The device used in the USB exceeds the device range.	Continue					V
7044	The device length of the communication data in the USB exceeds the limit.	Continue					V
7047	The device checksum for the communication serial port of the USB is incorrect.	Continue					V
70B1	The device communication function code in the function card 1 is incorrect.	Continue					V
70B2	The device communication address used in the function card 1 is incorrect.	Continue					V
70B3	The device used in the function card 1 exceeds the device range.	Continue					V
70B4	The device length of the communication data in the function card 1 exceeds the limit.	Continue					V
70B7	The device checksum for the communication serial port of the function card 1 is incorrect.	Continue					V
70C1	The device communication function code in the function card 2 is incorrect.	Continue					V
70C2	The device communication address used in the function card 2 is incorrect.	Continue					V
70C3	The device used in the function card 2 exceeds the device range.	Continue					V
70C4	The device length of the communication data in the function card 2 exceeds the limit.	Continue					V
70C7	The device checksum for the communication serial port of the function card 2 is incorrect.	Continue					V
7203	Invalid communication function code	Continue					V
8105	The contents of the program downloaded are incorrect. The program syntax is incorrect.	Continue					V
8106	The contents of the program downloaded are incorrect. The length of the execution code exceeds the limit.	Continue					V
8107	The contents of the program downloaded are incorrect. The length of the source code exceeds the limit.	Continue					V

7.4.2 Error Codes and LED Indicators for Analog/Temperature Modules

Error code	Description	ERROR LED indicator status	
		A → D / D → A / A ↔D	ERROR
16#1605	Hardware failure	OFF	ON
16#1607	The external voltage is abnormal.	OFF	ON

Error code	Description	ERROR LED indicator status	
		A → D / D → A / A ↔D	ERROR
16#1608	The factory calibration or the CJC is abnormal.	OFF	ON
16#1801*1	The external voltage is abnormal.	OFF	Blinking
16#1802*1	Hardware failure	OFF	Blinking
16#1804*1	The factory calibration is abnormal.	RUN: Blinking STOP: OFF	Blinking
16#1807*1	The CJC is abnormal.	OFF	Blinking
16#1808	The signal received by channel 1 exceeds the range of analog inputs (temperature).	RUN: Blinking STOP: OFF	Blinking
16#1809	The signal received by channel 2 exceeds the range of analog inputs (temperature).		
16#180A	The signal received by channel 3 exceeds the range of analog inputs (temperature).		
16#180B	The signal received by channel 4 exceeds the range of analog inputs (temperature).		

*1: The following errors are specified as warnings to ensure the CPU module can still run even when the warnings are triggered by its AIO modules. Users can set up in HWCONFIG to have them shown as errors when the first 4 errors occurred.

7.4.3 Error Codes and LED Indicators for Load Cell Module AS02LC

Error code	Description	ERROR LED indicator status	
		A → D	ERROR
16#1605	Hardware failure (the diver board included)	OFF	ON
16#1607	The external voltage is abnormal.	OFF	ON
16#1801*1	The external voltage is abnormal.	OFF	Blinking
16#1802*1	Hardware failure	OFF	Blinking
16#1807*1	Diver board failure	OFF	Blinking
16#1808	The signal received by channel 1 exceeds the range of analog inputs or the SEN voltage is abnormal.	RUN: Blinking STOP: OFF	Blinking
16#1809	The signal received by channel 1 exceeds the weight limit.		
16#180A	The factory calibration in channel 1 is incorrect.		
16#180B	The signal received by channel 2 exceeds the range of analog inputs or the SEN voltage is abnormal.		
16#180C	The signal received by channel 2 exceeds the weight limit.		
16#180D	The factory calibration in channel 2 is incorrect.		

*1: The following errors are specified as warnings to ensure the CPU module can still run even when the warnings are triggered by its AIO modules. Users can set up in HWCONFIG to have them shown as errors when the 3 errors occurred.

7.4.4 Error Codes and LED Indicators for Module AS00SCM as a Communication Module

Error Code	Description	ERROR LED indicator status	
		ON	Blinking
16#1605	Hardware failure	V	
16#1606	The setting of the function card is incorrect.	V	
16#1802	Incorrect parameters		V
16#1803	Communication timeout		V
16#1804	The setting of the UD Link is incorrect.		V

7.4.5 Error Codes and LED Indicators for Module AS00SCM as a Remote Module

Error Code	Description	ERROR LED indicator status		
		ON	**Blinking**	**Blinking fast**
16#1301	Hardware failure	V		
16#1302	The setting of the function card is incorrect.	V		
16#1303	24VDC power supply is not sufficient and then is recovered from a low-voltage less than 10ms situation.			V
16#1502	Incorrect parameters		V	
16#1503	Extension module communication timeout		V	

MEMO