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

**Deltronics (The Netherlands) B.V.**
Eindhoven Office
De Witbogt 15, 5652 AG Eindhoven, The Netherlands
TEL: 31-40-2592850 / FAX: 31-40-2592851



AH-0101620-03





2016-08-15





*AH500* **Motion Control Module Manual**



# *AH500* Motion Control Module Manual

www.deltaww.com

Smarter. Greener. Together.

# AH500 Motion Control Module Manual

## Revision History

| Version | Revision | Date |
|---|---|---|
| 1st | The first version was published. | 2013/09/30 |
| 2nd | 1. In section 2.1.2, the electrical specifications for the input terminals on AH05PM-5A are updated.<br>2. In section 2.3, the descriptions of the DMCNET port on AH20MC-5A are updated.<br>3. Six Delta DMCNET cable models are added to section 2.3.<br>4. The descriptions of SM303 and SM304 are added to section 3.12.1.<br>5. The instruction tables in section 4.1 are updated.<br>6. The device tables in section 4.2 are updated.<br>7. The descriptions of SM/SR devices and X/Y devices are added to section 5.2.<br>8. In section 5.6, the device tables, the explanation of the instruction WDT, and the additional remarks on the instructions WDT and XCH are updated.<br>9. The descriptions of pins in section 5.10~section 5.13 are updated.<br>10. The description of (SR1069+100*N, SR1068+100*N) is added to section 7.1.<br>11. A note is added to section 8.3.1.3.<br>12. Section 11.2 and section 11.4 are updated.<br>13. In section A.1, the table of servo drive error codes is updated. | 2014/12/01 |
| 3rd | 1. Update information on section 6.2.<br>2. Add information concerning SR1069, SR1042~1068 to section 7.1<br>3. Update information concerning C220 counter to section 8.2.2.1.<br>4. Update information concerning instruction G02~G04 to chapter 9.<br>5. Update information on section 11.2~11.3.<br>6. Update images of updating firmware to section 12.3~12.5.<br>7. Update information concerning SR1078 to the section 14.6.<br>8. Chapter 15, page 8 and 11 are updated. | 2016/08/15 |

# AH500 Motion Control Module Manual

## Table of Contents

# Chapter 1 Framework of an AH500 Series Motion Control Module

## Table of Contents

The Delta programmable logic controllers AH20MC-5A, AH15PM-5A, AH10PM-5A, and AH05PM-5A can put axes in particular positions at high-speeds, create linear interpolations, and circular interpolations. They can execute basic instructions, applied instructions, motion instructions, and G-codes.

In this manual, AH20MC-5A, AH15PM-5A, AH10PM-5A, and AH05PM-5A are called AH500 series motion control modules.

In this chapter, the basic frameworks of AH20MC-5A, AH15PM-5A, AH10PM-5A, and AH05PM-5A are described. Owing to the fact that the functionality of an AH500 series motion control module is composed of sequence control and positioning control, a program comprises O100, Ox motion subroutines, and P subroutines. O100, Ox motion subroutines, and P subroutines are described in this chapter. Basic instructions, applied instructions, motion instructions, and G-codes will be introduced in other chapters.

## 1.1 Framework of a AH500 Series Motion Control Module

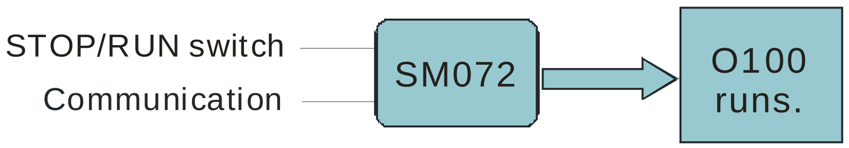
<table>
<tr><th rowspan="2">Item</th><th colspan="4">Specifications</th></tr>
<tr><th>AH20MC-5A</th><th>AH10PM-5A</th><th>AH15PM-5A</th><th>AH05PM-5A</th></tr>
<tr><td>Number of substantial axes supported</td><td>12 axes<br>(Axis 1~axis 12)</td><td>6 axes<br>(Axis 1~axis 6)</td><td>4 axes<br>(Axis 1~axis 4)</td><td>2 axes<br>(Axis 1~axis 2)</td></tr>
<tr><td>Number of virtual axes supported</td><td>4 axes<br>(Axis 13~axis 16)</td><td>10 axes<br>(Axis 7~axis 16)</td><td>12 axes<br>(Axis 5~axis 16)</td><td>14 axes<br>(Axis 3~axis 16)</td></tr>
<tr><td>Storage</td><td colspan="4">The capacity of the built-in storage is 64K steps.</td></tr>
<tr><td>Unit</td><td colspan="4">Motor unit, mechanical unit, and compound unit</td></tr>
<tr><td>Connection with a CPU module</td><td colspan="4">Users can set the initial register involved in data exchange in a CPU module, and the number of registers involved in the data exchange in the CPU module. Four hundred D/M registers at most can be involved in data exchange.</td></tr>
<tr><td rowspan="2">Motor control</td><td rowspan="2">Delta high-speed motion control system DMCNET (Delta Motion Control Network)<br>The response time is one millisecond.</td><td colspan="3">Three selectable pulse output modes: Pulse/Dir, FP(CW)/RP(CCW), A/B-phase.</td></tr>
<tr><td>The output terminals used as the first axis~the fourth axis are differential output terminals. The output terminals used as the fifth axis~the sixth axis are transistors whose collectors are open collectors.</td><td colspan="2">All are differential outputs.</td></tr>
<tr><td>Maximum speed</td><td colspan="4">Single axis: 1M pps<br>Multiaxial interpolation: 1M pps</td></tr>
</table>

| Item | | Specifications: AH20MC-5A | AH10PM-5A | AH15PM-5A | AH05PM-5A |
|---|---|---|---|---|---|
| Input signal | Operating switch | STOP/RUN switch | | | None |
| | Input terminal | X0.0+, X0.0-, X0.1+, X0.1-, X0.2+, X0.2-, X0.3+, X0.3-, X0.8+, X0.8-, X0.9+, X0.9-, X0.10+, X0.10-, X0.11+, X0.11-, X0.12+, X0.12-, X0.13+, X0.13, X0.14+, X0.14-, X0.15+, and X0.15- | X0.0+, X0.0-, X0.1+, X0.1-, X0.2+, X0.2-, X0.3+, X0.3-, X0.8, X0.9, X0.10, X0.11, X0.12, X0.13, X0.14, and X0.15 | X0.0+, X0.0-, X0.1+, X0.1-, X0.2+, X0.2-, X0.3+, X0.3-, X0.4, X0.5, X0.6, X0.7, X0.8+, X0.8-, X0.9+, X0.9-, X0.10, X0.11, X0.12, X0.13, X0.14, X0.15, X1.0, X1.1, X1.2, X1.3, X1.4, and X1.5 | X0.0, X0.1, X0.8, X0.9, X0.12, and X0.13 |
| Output signal | Output terminal | Y0.8, Y0.9, Y0.10, and Y0.11 | Y0.0+, Y0.0-, Y0.1+, Y0.1-, Y0.2+, Y0.2-, Y0.3+, Y0.3-, Y0.4+, Y0.4-, Y0.5+, Y0.5-, Y0.6+, Y0.6-, Y0.7+, Y0.7-, Y0.8, Y0.9, Y0.10, and Y0.11 | Y0.0+, Y0.0-, Y0.1+, Y0.1-, Y0.2+, Y0.2-, Y0.3+, Y0.3-, Y0.4+, Y0.4-, Y0.5+, Y0.5-, Y0.6+, Y0.6-, Y0.7+, Y0.7-, Y0.8, Y0.9, Y0.10, and Y0.11 | Y0.0+, Y0.0-, Y0.1+, Y0.1-, Y0.2+, Y0.2-, Y0.3+, Y0.3-, Y0.8, and Y0.9 |
| | External communication port | Mini USB port<br>Ethernet port<br>DMCNET port | Mini USB port<br>Ethernet port | Mini USB port<br>Ethernet port | Mini USB port |
| Expansion storage device | | Mini SD card<br>The maximum capacity is 32 GB. | | | None |
| Number of basic instructions | | 27 | | | |
| Number of applied instructions | | 130 | | | |
| M-code | | Ox0~Ox99 (motion subroutine/positioning program): M02 (The execution of a program stops. (END))<br>M00~M01, M03~M101, and M103~M65535: The execution of a program pauses. (WAIT)<br>Users can use them freely. | | | |
| G-code | | G0 (rapid positioning), G1 (linear interpolation), G2 (circular interpolation, clockwise), G3 (circular interpolation, counterclockwise), G4 (dwell), G17 (XY plane selection), G18 (ZX plane selection), G19 (YZ plane selection), G90 (absolute programming), and G91 (incremental programming) | | | |
| Number of counters | | 6 | 6 | 6 | 1 |
| Number of high-speed catchers/comparators | | 8 | 8 | 8 | 3 |
| Number of interrupt devices | | 9 | 9 | 9 | 5 |

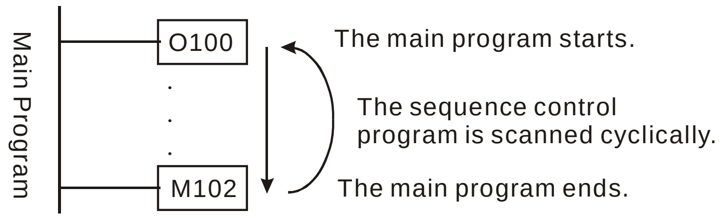
## 1.2 Structure of O100

O100 is a sequence control program. It is the main program in an AH500 series motion control module. It only supports basic instructions and applied instructions. Users can use these two types of instructions to process I/O data, call P subroutines, and enable Ox motion subroutines (Ox0~Ox99). O100 functions as a main program. Motion subroutines are enabled through O100. There is hierarchical relation between O100 and motion subroutines. The characteristics of O100 are described below.

1. There are two methods of enabling O100.
   - If the STOP/RUN switch of an AH500 series motion control module is turned form the “STOP” position to the “RUN” position when the AH500 series motion control module is powered, SM072 will be ON, and O100 will run.
   - If an AH500 series motion control module is powered, users can use communication to set SM072 to ON, and to run O100.




2. O100 is scanned cyclically. The scan of the main program O100 starts from the starting flag O100. After the ending instruction M102 is scanned, the scan of the main program O100 will go back to the starting flag O100.




3. There are three methods of disabling O100.
   - If the STOP/RUN switch of an AH500 series motion control module is turned form the “RUN” position to the “STOP” position when the AH500 series motion control module is powered, SM072 will be OFF, and O100 will stop. If O100 stops, Ox motion subroutines and P subroutines will not be executed.
   - If an AH500 series motion control module is powered, users can use communication to set SM072 to OFF, and to stop O100. If O100 stops, Ox motion subroutines and P subroutines will not be executed.
   - If an error occurs when O100 is compiled or when O100 runs, O100 will stop automatically.
4. O100 supports basic instructions and applied instructions. Users can write a control program according to their needs. They can set the parameters of motion instructions, and motion subroutine numbers (Ox0~Ox99) in O100.
   - O100 does not support motion instructions and G-codes. Motion instructions and G-codes must be used in the motion subroutines Ox0~Ox99. Please refer to section 1.2 for more information.
   - O100 can call P subroutines. Please refer to section 1.4 for more information.
5. The description of O100 is shown below.

| O100 | Description |
|---|---|
| **Enabling O100** | Starting flag O100 (If O100 is a ladder diagram in PMSoft, the starting flag in O100 will be set automatically, and users do not have to write the starting flag.) |

| O100 | Description |
|---|---|
| **Disabling O100** | Ending instruction M102 (If O100 is a ladder diagram in PMSoft, the ending instruction M102 will be set automatically, and users do not have to write the ending instruction M102.) |
| **Executing O100** | 1. The STOP/RUN switch of an AH500 series motion control module is turned form the “STOP” position to the “RUN” position.<br>2. Users use communication to set SM072 to ON. |
| **Operation characteristic** | O100 is scanned cyclically. |
| **Instruction** | Basic instructions and applied instructions are supported. |
| **Number** | There is only one O100 in a program. |
| **Characteristic and function** | 1. It is a sequence control program.<br>2. It can enable the motion subroutines Ox0~Ox99, and call P subroutines.<br>3. If O100 is used with Ox motion subroutines and P subroutines, O100, the Ox motion subroutines, and the P subroutines can be arranged in any order. |

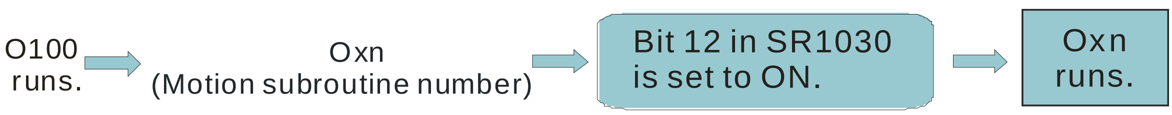
### 1.2.1 Manual Function of O100

Users can set motion modes by means of special registers in O100. (Please refer to Chapter 7 for more information.)

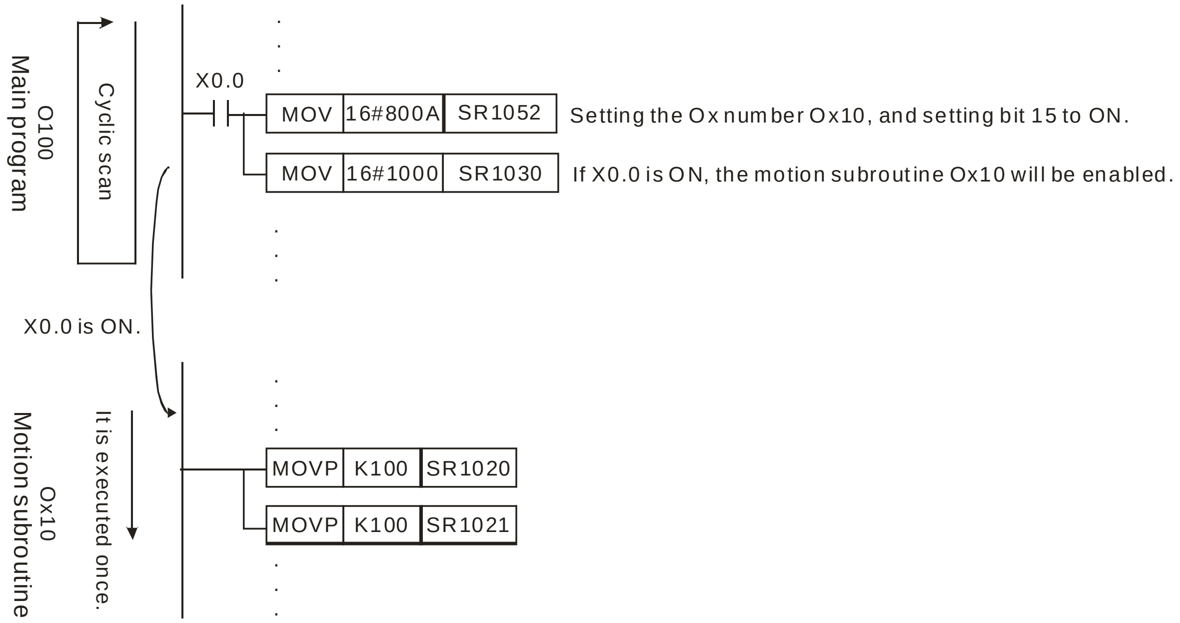
## 1.3 Structure of Ox Motion Subroutines

The motion subroutines Ox0~Ox99 are motion control programs. They are subroutines which control the motion of the axes of an AH500 series motion control module. Ox0~Ox99 support basic instructions, applied instructions, motion instructions, and G-codes. They can call P subroutines. Users can control the paths of the axes of an AH500 series motion control module through Ox motion subroutines. The characteristics of Ox motion subroutines are described below.

1. There are two methods of enabling an Ox motion subroutine.
   - When O100 runs, users can set motion subroutine numbers in O100. (The motion subroutine numbers must be in the range of Ox0 to Ox99. The users can set a motion subroutine number in O100 by setting SR1052. The value in SR1052 must be in the range of 16#8000 to 16#8063.) If the users want to enable an Ox motion subroutine, they have to set bit 12 in SR1030 to ON.
   - Before an Ox motion subroutine is enabled, users have to make sure that no Ox motion subroutine runs.



2. Whenever an Ox motion subroutine is enabled, it is executed once. After O100 enables an Ox motion subroutine, the execution of the Ox motion subroutine will start from the starting flag in the Ox motion subroutine. After the ending instruction M2 in the Ox motion subroutine is executed, the execution of the Ox motion subroutine will stop.




If X0.0 is ON, the motion subroutine Ox10 will be enabled. After the ending instruction M2 in Ox10 is executed, the execution of Ox10 will stop. (Ox10 is executed once. If Ox10 needs to be executed again, X0.0 has to be set to ON.)

3. There are four methods of disabling an Ox motion subroutine.
   - If the STOP/RUN switch of an AH500 series motion control module is turned form the "RUN" position to the "STOP" position when the AH500 series motion control module is powered, SM072 will be OFF, O100 will stop, and Ox motion subroutines will not be executed.
   - If an AH500 series motion control module is powered, users can use communication to set SR1030 to 0, or to set SM072 to OFF, and to stop the execution of Ox motion subroutines.
   - If an error occurs when an Ox motion subroutine is compiled or when an Ox motion subroutine is executed, the execution of the Ox motion subroutine will stop automatically.
4. An Ox motion subroutine supports basic instructions, applied instructions, motion instructions, and G-codes. Users can write a motion program according to their needs. They can control the motion of the axes of an AH500 series motion control module by setting the parameters of the axes.
   - Basic instructions, applied instructions, motion instructions and G-codes must be used in the motion subroutines Ox0~Ox99.
   - Ox motion subroutines can call P subroutines. Please refer to section 1.4 for more information.
5. The description of Ox motion subroutines is shown below.

| Ox motion subroutine | Description |
|---|---|
| **Enabling an Ox motion subroutine** | There are 100 Ox motion subroutines (Ox0~Ox99).<br>(If an Ox motion subroutine is a ladder diagram in PMSoft, the starting flag in the Ox motion subroutine will be set automatically, and users do not have to write the starting flag.) |
| **Disabling an Ox motion subroutine** | Ending instruction M2 (If an Ox motion subroutine is a ladder diagram in PMSoft, the ending instruction M2 will be set automatically, and users do not have to write the ending instruction M2.) |

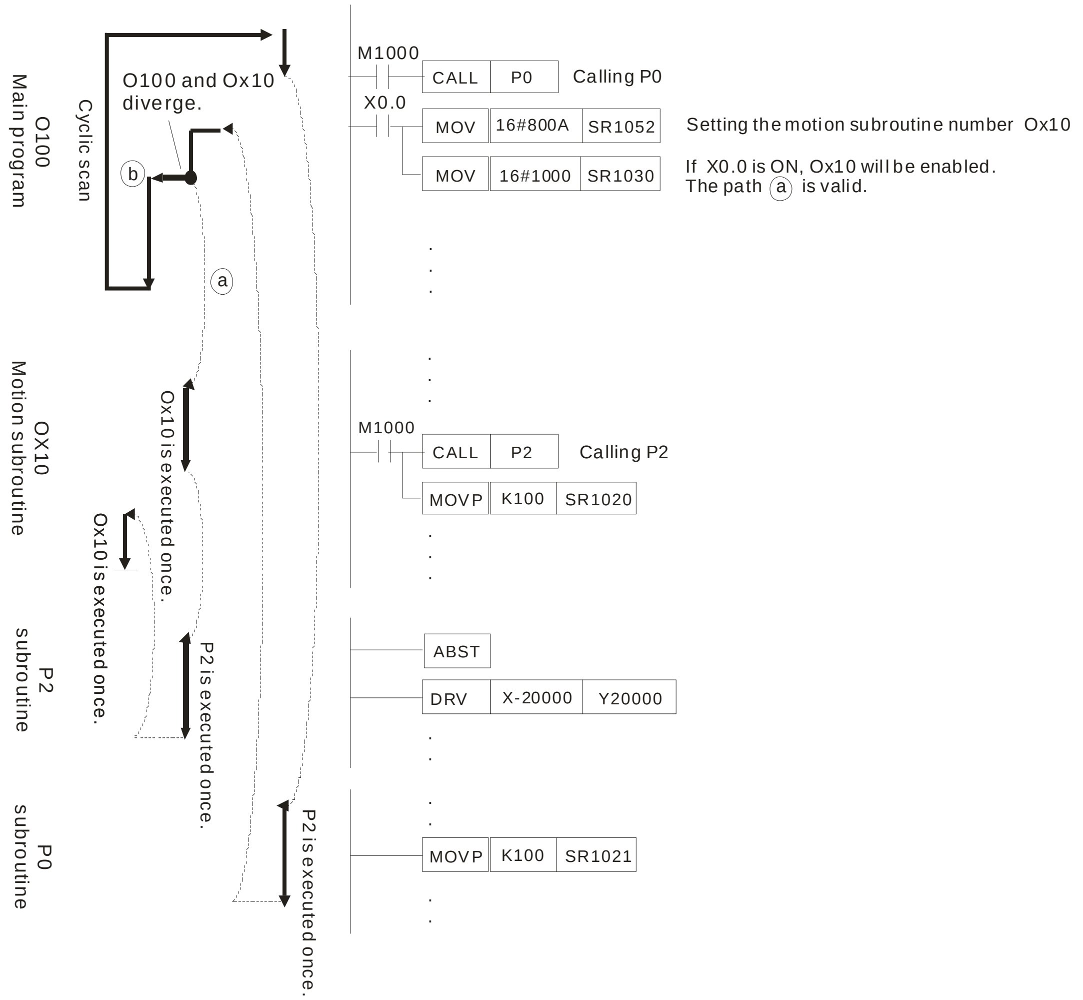
| Ox motion subroutine | Description |
|---|---|
| **Executing an Ox motion subroutine** | 1. If users set bit 12 in SR1030 to ON when O100 runs, an Ox motion subroutine will be enabled.<br>2. If users use communication to set bit 12 in SR1030 to ON when O100 runs, an Ox motion subroutine will be enabled.<br>Note: Before an Ox motion subroutine is enabled, users have to make sure that no Ox motion subroutine runs. |
| **Operation characteristic** | Whenever an Ox motion subroutine is enabled, it is executed once. If an Ox motion subroutine needs to be executed again, it has to be enabled again. |
| **Instruction** | Basic instructions, applied instructions, motion instructions, and G-codes are supported.<br>Note: Users have to avoid using pulse instructions. |
| **Number** | There are 100 Ox motion subroutines in a program. If users want to enable a motion subroutine number, they have to set SR1052, and set bit 12 in SR1030 to ON. |
| **Characteristic and function** | 1. Ox0~Ox99 are motion subroutines. (They can only be enabled by O100.)<br>2. They can control the motion of the axes of an AH500 series motion control module. Please refer to the description of G-code for more information.<br>3. An Ox motion subroutine can be enabled/disabled by a program or communication.<br>4. Ox motion subroutines can call P subroutines.<br>5. If Ox motion subroutines are used with O100 and P subroutines, the Ox motion subroutines, O100, and the P subroutines can be arranged in any order. |

## 1.4 Structure of P Subroutines

P subroutines are general subroutines. They can be called by O100 and Ox motion subroutines. If P subroutines are called by O100, the P subroutines will support basic instructions and applied instructions. If P subroutines are called by Ox0~Ox99, the P subroutines will support basic instructions, applied instructions, motion instructions, and G-codes. After O100 or an Ox motion subroutine calls a P subroutine, the P subroutine will be executed. After SRET in the P subroutine is executed, the lines under the instruction which calls the P subroutine will be executed.

1. There are two methods of enabling a P subroutine.
   - O100 can call P subroutines.
   - Ox motion subroutines can call P subroutines.

2. Whenever a P subroutine is called, it is executed once. After O100 or an Ox motion subroutine calls a P subroutine, the P subroutine will be executed. After the ending instruction SRET in the P subroutine is executed, the execution of the P subroutine will stop, and the lines under the instruction which calls the P subroutine will be executed.




The subroutine P0 supports basic instructions and applied instructions. The subroutine P2 supports basic instructions, applied instructions, motion instructions, and G-codes.

3. There are three methods of disabling a P subroutine.
   - If the STOP/RUN switch of an AH500 series motion control module is turned form the “RUN” position to the “STOP” position when the AH500 series motion control module is powered, SM072 will be OFF, O100 will stop, and Ox motion subroutines and P subroutines will not be executed.
   - If an AH500 series motion control module is powered, users can use communication to set SR1030 to 0, to stop the execution of Ox motion subroutines, and to stop the execution of P subroutines.
   - If an error occurs when a P subroutine is executed, the execution of the P subroutine will stop automatically. Please refer to appendix A for more information.
4. If P subroutines are called by O100, the P subroutines will support basic instructions and applied instructions. If P subroutines are called by Ox0~Ox99, the P subroutines will support basic instructions, applied instructions, motion instructions, and G-codes.

5. The description of P subroutines is shown below.

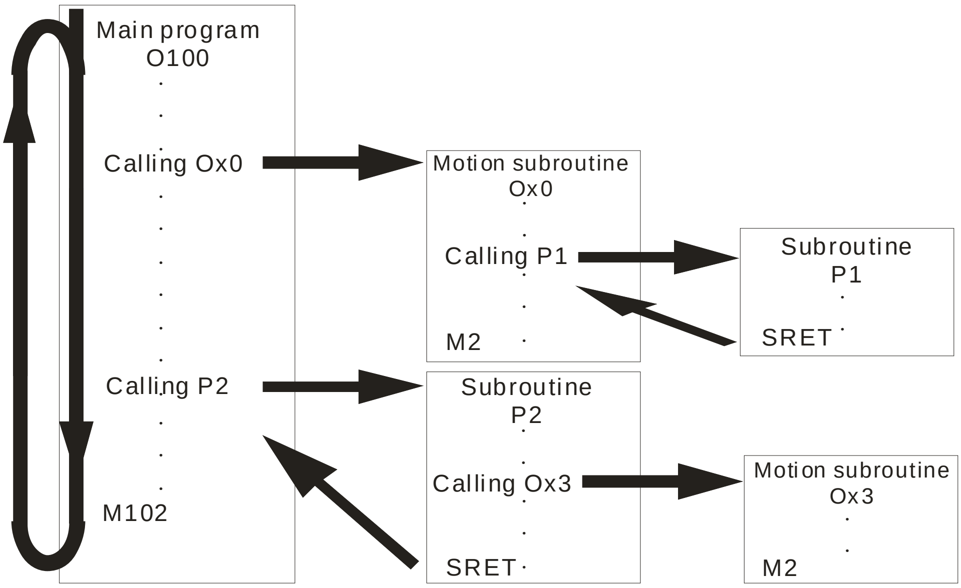
| P subroutine | Description |
|---|---|
| **Enabling a P subroutine** | There are 256 P subroutines (P0~P255).<br>(If a P subroutine is a ladder diagram in PMSoft, the starting flag in the P subroutine will be set automatically, and users do not have to write the starting flag.) |
| **Disabling a P subroutine** | Ending instruction SRET (If a P subroutine is a ladder diagram in PMSoft, the ending instruction SRET will be set automatically, and users do not have to write the ending instruction SRET.) |
| **Executing a P subroutine** | 1. O100 can call P subroutines.<br>2. Ox motion subroutines can call P subroutines. |
| **Operation characteristic** | Whenever a P subroutine is enabled, it is executed once. If a Pn subroutine needs to be executed again, it has to be enabled again. |
| **Instruction** | 1. If P subroutines are called by O100, the P subroutines will support basic instructions and applied instructions.<br>2. If P subroutines are called by Ox motion subroutines, the P subroutines will support basic instructions, applied instructions, motion instructions, and G-codes.<br>Note: If P subroutines are called by Ox motion subroutines, users have to avoid using pulse instructions. |
| **Number** | There are 256 P subroutines in a program. |
| **Characteristic and function** | 1. P subroutines are general subroutines.<br>2. P subroutines can be called by O100 and Ox motion subroutines.<br>3. If P subroutines are used with O100 and Ox motion subroutines, the P subroutines, O100, and the Ox motion subroutines can be arranged in any order. |

# 1.5 Using O100, Ox Motion Subroutines, and P Subroutines

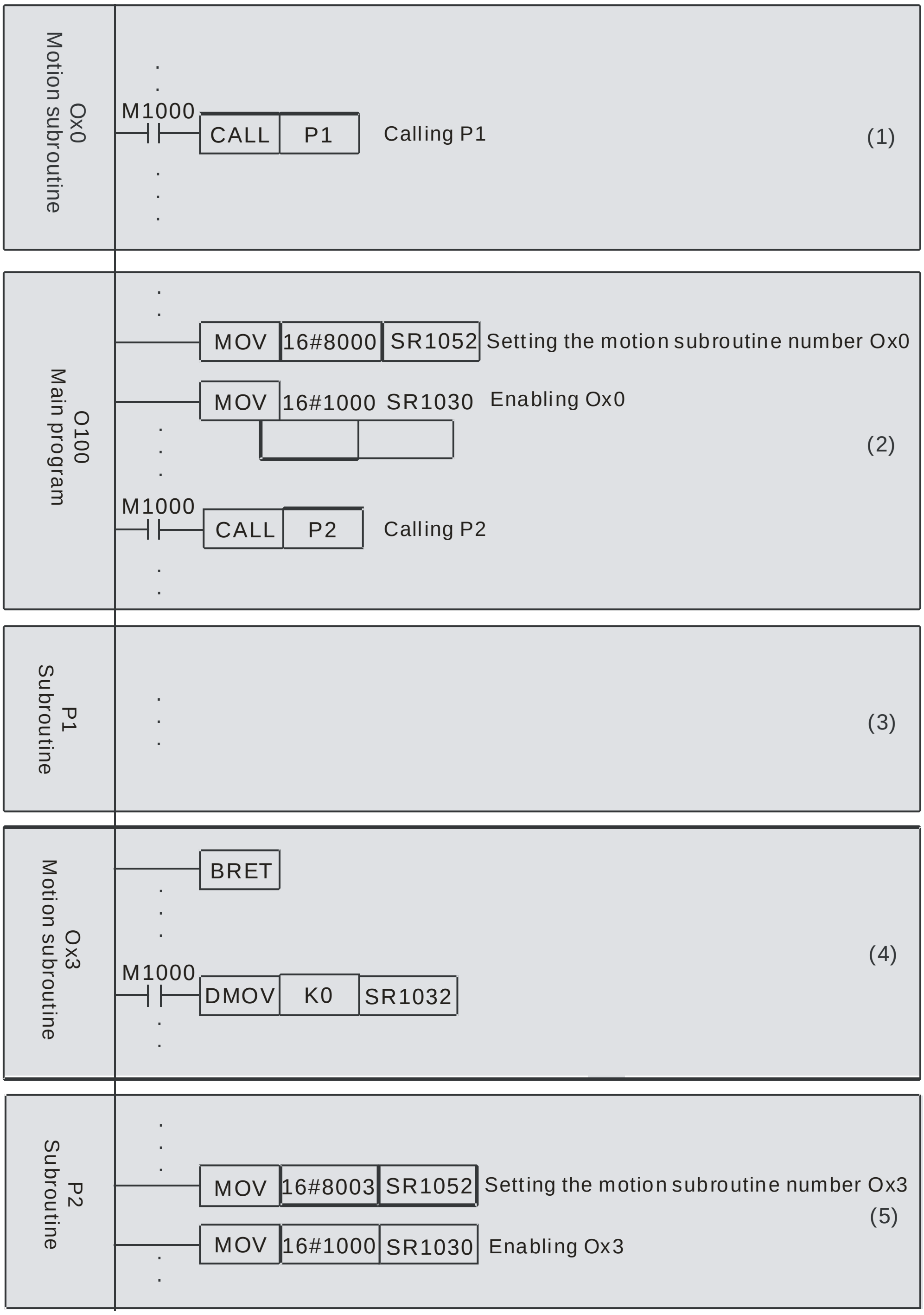
O100, Ox motion subroutines, and P subroutines are introduced in section 1.1~section 1.3. In this section, a program composed of O100, Ox motion subroutines, and P subroutines is described.

## 1.5.1 Structure of a Program

Suppose a program is composed of O100, Ox0, Ox3, P1, and P2. The five program blocks are shown below.



In order to describe the program, the program is divided into 5 sections (section (1)~section (5)).



The program is described below.
1. Section (1)~section (5) are created in numerical order, but they can be arranged in any order.
2. There is only one O100. O100 can not be called by another program, but it can freely call Ox motion subroutines and P subroutines.
3. Ox motion subroutines can be called by O100 and P subroutines, and it can call P subroutines.
4. P subroutines can be called by O100 and Ox motion subroutines, and it can call Ox motion subroutines.

Note:
1. One Ox motion subroutine is executed at a time. If Ox0 is executed, Ox3 can not be executed. If Ox3 is executed, Ox0 can not be executed.
2. After O100 or a P subroutine enables an Ox motion subroutine, the next line will be executed, and the execution of the Ox motion subroutine will be ignored.
3. Whenever an Ox motion subroutine is enabled, it is executed once. If an Ox motion subroutine needs to be executed again, it has to be enabled again.

The instructions supported by O100, Ox0, Ox3, P1 and P3 are described below. (O: Supported; X: Not supported)

| Section | O100 | Ox0 and Ox3 | P1 | P2 |
|---|---|---|---|---|
| Basic instruction | O | O | O | O |
| Applied instruction | O | O | O | O |
| Motion instruction | X | O | O | X |
| G-code | X | O | O | X |
| Description | - | - | P1 is called by Ox0, and therefore it supports motion instructions and G-codes. | P2 is called by O100, and therefore it does not support motion instructions and G-codes. |

Additional remark:

| | Main program | Subroutine | Motion subroutine |
|---|---|---|---|
| Order | In any order | In any order | In any order |
| Execution | It runs normally. | P subroutines can be called by O100 or Ox motion subroutines. | Ox motion subroutines can be called by O100 or P subroutines. |
| Operation | It is scanned cyclically. | Whenever a subroutine is called, it is executed once. | Whenever a motion subroutine is called, it is executed once. |
| Number | 1 main program | 256 subroutines<br>They can be used according to users' needs. | 100 motion subroutines<br>They can be used according to users' needs. |

# Chapter 2 Hardware Specifications and Wiring

## Table of Contents

# 2.1 Hardware Specifications

Electrical specifications and wiring are described in this chapter. Please refer to other chapters for more information about the writing of a program and the use of instructions. For more information about the peripherals purchased, please refer to the manuals attached to them.

## 2.1.1 General Specifications

| Item | Specifications |
|---|---|
| **Connector type** | High precision connector<br>It must be connected to an external terminal module. |
| **Supply voltage** | 5 V DC (-15~20%), 24 V DC (-15~20%)<br>(AHPS05-5A supplies power through a bus.) |
| **Electric energy consumption** | 2 W |
| **Insulation voltage** | 2,500 VDC |
| **Weight** | 150 g |
| **Noise immunity** | ESD (IEC 61131-2, IEC 61000-4-2): ±10 kV air discharge<br>EFT (IEC 61131-2, IEC 61000-4-4): Communication I/O: ±4 kV<br>CS (IEC 61131-2, IEC 61000-4-6): 0.15~80 MHz, 3 Vrms<br>RS (IEC 61131-2, IEC 61000-4-3): 80~100 MHz, 10 V/m, 1.4~2.0 GHz |
| **Operating/Storage environment** | Operating environment: -20~70°C (Temperature), 5~95% (Humidity), pollution degree 2<br>Storage environment: -40~85°C (Temperature), 5~95% (Humidity) |
| **Vibration/Shock resistance** | International standards IEC 61131-2, IEC 68-2-6 (TEST Fc)/IEC61131-2 & IEC 68-2-27 (TEST Ea) |
| **Standard** | CE, cULus |

## 2.1.2 Electrical Specifications for the Input Terminals

**1. AH20MC-5A**

| Item / Specifications | | Differential input |
|---|---|---|
| | | **High speed of 200 kHz** |
| **Wiring type** | | Independent wiring |
| **Input voltage** | | 5~24 V DC |
| **Maximum input current** | | 15 mA |
| **Action level** | **OFF→ON** | 20 us |
| | **ON→OFF** | 30 us |
| **Response time/Noise reduction** | | 10 ms/0.5 us |

| Terminal | Description | Response characteristic | Maximum input | |
|---|---|---|---|---|
| | | | Current | Voltage |
| **X0.0+, X0.0-, X0.1+, X0.1-, X0.2+, X0.2-, X0.3+, and X0.3-** | 1. They are differential input terminals.<br>2. Functions of the terminals:<br>● High-speed count:<br>◆ The terminals are the Reset input terminals for counter 0~counter 5.<br>◆ X0.0+ and X0.0- are for counter 0. X0.1+ and X0.1- are for counter 1. X0.2+ and X0.2- are for counter 2 and counter 4. X0.3+ and X0.3- are for counter 3 and counter 5.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture. | 200 kHz | 15 mA | 5~24 V |
| **X0.8+, X0.8-, X0.9+, and X0.9-** | 1. They are differential input terminals.<br>2. Functions of the terminals:<br>● Motion control: The terminals are for a manual pulse generator.<br>● High-speed count:<br>◆ The terminals are for counter 0.<br>◆ X0.8+ and X0.8- are the A-phase input terminals for counter 0. X0.9+ and X0.9- are the B-phase input terminals for counter 0.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture.<br>● Interrupt input terminals | 200 kHz | 15 mA | 5~24 V |

| Terminal | Description | Response characteristic | Maximum input | |
|---|---|---|---|---|
| | | | Current | Voltage |
| **X0.10+, X0.10-, X0.11+, X0.11-, X0.12+, X0.12-, X0.13+, X0.13-, X0.14+, X0.14-, X0.15+, and X0.15-** | 1. They are differential input terminals.<br>2. Functions of the terminals:<br>● Motion control: They are the DOG input terminals for axis 1~axis 6. This function is used for inserting single-speed/two-speed motion.<br>● High-speed count:<br>◆ The terminals are for counter 1~counter 5.<br>◆ X0.10+ an X0.10- are the A-phase input terminals for counter 1. X0.12+ and X0.12- are the A-phase input terminals for counter 2 and counter 4. X0.14+ and X0.14- are the A-phase input terminals for counter 3 and counter 5.<br>◆ X0.11+ and X0.11- are the B-phase input terminals for counter 1. X0.13+ and X0.13- are the B-phase input terminals for counter 2 and counter 4. X0.15+ and X0.15- are the B-phase input terminals for counter 3 and counter 5.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture.<br>● Interrupt input terminals | 200 kHz | 15 mA | 5~24 V |

**2. AH10PM-5A**

| Item / Specifications | | Differential input | Open collector |
|---|---|---|---|
| | | **High speed of 200 kHz** | **100 kHz (*1)** |
| **Wiring type** | | Independent wiring | Current flows into the common terminal S/S (sinking), or current flows from the common terminal S/S (sourcing). |
| **Input voltage** | | 5~24 V DC | 24 V DC |
| **Maximum input current** | | 15 mA | |
| **Action level** | **OFF→ON** | 20 us | |
| | **ON→OFF** | 30 us | |
| **Response time/Noise reduction** | | 10 ms/0.5 us | |

| Terminal | Description | Response characteristic | Maximum input | |
|---|---|---|---|---|
| | | | Current | Voltage |
| **X0.0+, X0.0-, X0.1+, X0.1-, X0.2+, X0.2-, X0.3+, and X0.3-** | 1. They are differential input terminals.<br>2. Functions of the terminals:<br>● Motion control: They are the PG input terminals for axis 1~axis 4.<br>● High-speed count: X0.0+ and X0.0- are the Reset input terminals for counter 0. X0.1+ and X0.1- are the Reset input terminals for counter 1. X0.2+ and X0.2- are the Reset input terminals for counter 2 and counter 4. X0.3+ and X0.3- are the Reset input terminals for counter 3 and counter 5.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture. | 200 kHz | 15 mA | 5~24 V |
| **X0.8 and X0.9** | 1. They are single/A/B-phase input terminals.<br>2. Functions of the terminals:<br>● Motion control: The terminals are for a manual pulse generator.<br>● High-speed count:<br>◆ The terminals are for counter 0.<br>◆ X0.8 is the A-phase input terminal for counter 0, and X0.9 is the B-phase input terminal for counter 0.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture.<br>● Interrupt input terminals | 100 kHz (*1) | 15 mA | 24 V |
| **X0.10, X0.11, X0.12, X0.13, X0.14, and X0.15** | 1. They are single/A/B-phase input terminals.<br>2. Functions of the terminals:<br>● Motion control: They are the DOG input terminals for axis 1~axis 6.<br>● High-speed count:<br>◆ The terminals are for counter 1~counter 5.<br>◆ X0.10 is the A-phase input terminal for counter 1, X0.12 is the A-phase input terminal for counter 2 and counter 4, and X0.14 is the A-phase input terminal for counter 3 and counter 5.<br>◆ X0.11 is the B-phase input terminal for counter 1, X0.13 is the B-phase input terminal for counter 2 and counter 4, and X0.15 is the B-phase input terminal for counter 3 and counter 5.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture.<br>● Interrupt input terminals | 100 kHz (*1) | 15 mA | 24 V |

*1. If the frequency of A/B-phase input signals must be 200 kHz, every channel must be connected to a 1 kΩ (2 W) resistor in parallel.

**3. AH15PM-5A**

<table>
<tr><th colspan="2">Item<br>Specifications</th><th>Differential input<br>High speed of 200 kHz</th><th>Open collector<br>100 kHz (*1)</th></tr>
<tr><td colspan="2">Wiring type</td><td>Independent wiring</td><td>Current flows into the common terminal S/S (sinking), or current flows from the common terminal S/S (sourcing).</td></tr>
<tr><td colspan="2">Input voltage</td><td>5~24 V DC</td><td>24 V DC</td></tr>
<tr><td colspan="2">Maximum input current</td><td colspan="2">15 mA</td></tr>
<tr><td rowspan="2">Action level</td><td>OFF→ON</td><td colspan="2">20 us</td></tr>
<tr><td>ON→OFF</td><td colspan="2">30 us</td></tr>
<tr><td colspan="2">Response time / Noise reduction</td><td colspan="2">10 ms/0.5 us</td></tr>
</table>

<table>
<tr><th rowspan="2">Terminal</th><th rowspan="2">Description</th><th rowspan="2">Response characteristic</th><th colspan="2">Maximum input</th></tr>
<tr><th>Current</th><th>Voltage</th></tr>
<tr><td>X0.0+, X0.0-, X0.1+, X0.1-, X0.2+, X0.2-, X0.3+, and X0.3-</td><td>1. They are differential input terminals.<br>2. Functions of the terminals:<br>● Motion control: They are the PG input terminals for axis 1~axis 4.<br>● High-speed count: X0.0+ and X0.0- are the Reset input terminals for counter 0. X0.1+ and X0.1- are the Reset input terminals for counter 1. X0.2+ and X0.2- are the Reset input terminals for counter 2 and counter 4. X0.3+ and X0.3- are the Reset input terminals for counter 3 and counter 5.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture.</td><td>200 kHz</td><td>15 mA</td><td>5~24 V</td></tr>
<tr><td>X0.4, X0.5, X0.6, and X0.7</td><td>1. They are single/A/B-phase input terminals.<br>2. Function of the terminals:<br>● Motion control: They are the DOG input terminals for axis 1~axis 4.</td><td>100 kHz (*1)</td><td>15mA</td><td>24 V</td></tr>
<tr><td>X0.8+, X0.8-, X0.9+, and X0.9-</td><td>1. They are differential input terminals.<br>2. Functions of the terminals:<br>● Motion control: The terminals are for a manual pulse generator.<br>● High-speed count:<br>◆ The terminals are for counter 0.<br>◆ X0.8+ and X0.8- are the A-phase input terminals for counter 0, and X0.9+ and X0.9- are the B-phase input terminal for counter 0.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture.<br>● Interrupt input terminals</td><td>200 kHz</td><td>15 mA</td><td>5~24 V</td></tr>
</table>

| Terminal | Description | Response characteristic | Maximum input | |
|---|---|---|---|---|
| | | | Current | Voltage |
| **X0.10, X0.11, X0.12, X0.13, X0.14, X0.15, X1.0 and X1.1** | 1. They are single/A/B-phase input terminals.<br>2. Functions of the terminals:<br>● Motion control: They are the LSP/NSP input terminals for axis 1~axis 4.<br>LSP input terminals: X0.10, X0.12, X0.14, and X1.0<br>LSN input terminals: X0.11, X0.13, X0.15, and X1.1<br>● High-speed count:<br>◆ The terminals are for counter 1~counter 5.<br>◆ X0.10 is the A-phase input terminal for counter 1, X0.12 is the A-phase input terminal for counter 2 and counter 4, and X0.14 is the A-phase input terminal for counter 3 and counter 5.<br>◆ X0.11 is the B-phase input terminal for counter 1, X0.13 is the B-phase input terminal for counter 2 and counter 4, and X0.15 is the B-phase input terminal for counter 3 and counter 5.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture.<br>● Interrupt input terminals: X0.10, X0.11, X0.12, X0.13, X0.14, and X0.15 | 100 kHz (*1) | 15 mA | 24 V |
| **X1.2, X1.3, X1.4, and X1.5** | 1. They are single/A/B-phase input terminals. | 100 kHz (*1) | 15 mA | 24 V |

*1. If the frequency of A/B-phase input signals must be 200 kHz, every channel must be connected to a 1 kΩ (2 W) resistor in parallel.

**4. AH05PM-5A**

| Specifications \ Item | | Open collector |
|---|---|---|
| | | High speed of 100 kHz (*1) |
| **Wiring type** | | Current flows into the common terminal S/S (sinking), or current flows from the common terminal S/S (sourcing). |
| **Input voltage** | | 24 V DC |
| **Maximum input current** | | 15 mA |
| **Action level** | **OFF→ON** | 20 us |
| | **ON→OFF** | 30 us |
| **Response time/Noise reduction** | | 10 ms/0.5 us |

| Terminal | Description | Response characteristic | Maximum input | |
|---|---|---|---|---|
| | | | Current | Voltage |
| **X0.0 and X0.1** | 1. They are single/A/B-phase input terminals.<br>2. Functions of the terminals:<br>● Motion control: X0.0 is the PG input terminal for axis 1, and X0.1 is the PG input terminal for axis 2.<br>● High-speed count: X0.0 is the Reset input terminal for counter 0.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture. | 100 kHz (*1) | 15 mA | 24 V |
| **X0.8 and X0.9** | 1. They are single/A/B-phase input terminals.<br>2. Functions of the terminals:<br>● Motion control: X0.8 and X0.9 are for a manual pulse generator.<br>● High-speed count:<br>◆ X0.8 is the A-phase input terminal for counter 0, and X0.9 is the B-phase input terminal for counter 0.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture.<br>● Interrupt input terminals | 100 kHz (*1) | 15 mA | 24 V |
| **X0.12 and X0.13** | 1. They are single/A/B-phase input terminals.<br>2. Functions of the terminals:<br>● Motion control: X0.12 is the DOG input terminal for axis 1, and X0.13 is the DOG input terminal for axis 2.<br>● High-speed comparison and capture: The terminals can function as trigger signals for high-speed capture.<br>● Interrupt input terminals | 100 kHz (*1) | 15 mA | 24 V |

*1. If the frequency of A/B-phase input signals must be 200 kHz, every channel must be connected to a 1 kΩ (2 W) resistor in parallel.

## 2.1.3 Electrical Specifications for the Output Terminals

**5. AH20MC-5A**

| Specifications \ Item | | Transistor output |
|---|---|---|
| **Maximum exchange (working) frequency** | | 200 kHz |
| **Output terminal** | | Y0.8~Y0.11 |
| **Working voltage** | | 5~30 V DC |
| **Maximum output current** | | 40 mA |
| **Isolation** | | Optocoupler |
| **Current specifications** | **Resistance** | 0.5 A/output terminal (2 A/COM) |
| | **Inductance** | 9 W (24 V DC) |
| | **Bulb** | 2 W (24 V DC) |
| **Response time** | **OFF→ON** | 0.2 us |
| | **ON→OFF** | |

| Item / Specifications | Transistor output |
|---|---|
| **Overcurrent protection** | N/A |

| Terminal | Description | Response characteristic | Maximum output Current | Maximum output Voltage |
|---|---|---|---|---|
| **Y0.8, Y0.9, Y0.10, and Y0.11** | 1. The high-speed pulse output terminals are transistors whose collectors are open collectors.<br>2. Function of the terminals:<br>● High-speed comparison and capture: They can be used as high-speed comparison output terminals. | 200 kHz | 15 mA | 24 V |

6. **AH10PM-5A**

| Item / Specifications | | Differential output | Transistor output |
|---|---|---|---|
| **Maximum exchange (working) frequency** | | 1 MHz | 200 kHz |
| **Output terminal** | | Y0.0~Y0.7 | Y0.8~Y0.11 |
| **Working voltage** | | 5 VDC | 5~30 VDC |
| **Maximum output current** | | 40 mA | 40 mA |
| **Isolation** | | Digital isolator | Optocoupler |
| **Current specifications** | **Resistance** | <25 mA | 0.5 A/output terminal (4 A/COM) |
| | **Inductance** | -- | 12 W (24 V DC) |
| | **Bulb** | -- | 2 W (24 V DC) |
| **Response time** | **OFF→ON** | 0.2 us | |
| | **ON→OFF** | | |
| **Overcurrent protection** | | Yes | No |

| Terminal | Description | Response characteristic | Maximum output Current | Maximum output Voltage |
|---|---|---|---|---|
| **Y0.8, Y0.9, Y0.10, and Y0.11** | 1. The high-speed pulse output terminals are transistors whose collectors are open collectors.<br>2. Functions of the terminals:<br>● Motion control:<br>◆ The terminals are the CLR output terminals for axis 1~axis 4.<br>◆ Y0.8 and Y0.9 are for axis 5. Y0.10 and Y0.11 are for axis 6. Y0.8 is the A-phase output terminal for axis 5, and Y0.10 is the A-phase output terminal for axis 6. Y0.9 is the B-phase output terminal for axis 5, and Y0.11 is the B-phase output terminal for axis 6.<br>● High-speed comparison and capture: They can be used as high-speed comparison output terminals. | 200 kHz | 15 mA | 24 V |

| Terminal | Description | Response characteristic | Maximum output | |
|---|---|---|---|---|
| | | | Current | Voltage |
| **Y0.0+, Y0.0-, Y0.1+, Y0.1-, Y0.2+, Y0.2-, Y0.3+, Y0.3-, Y0.4+, Y0.4-, Y0.5+, Y0.5-, Y0.6+, Y0.6-, Y0.7+, and Y0.7-** | 1. They are differential output terminals.<br>2. Function of the terminals:<br>● Motion control:<br>◆ The terminals are for axis 1~axis 4.<br>◆ Y0.0+ and Y0.0- are the A-phase output terminals for axis 1. Y0.2+ and Y0.2- are the A-phase output terminals for axis 2. Y0.4+ and Y0.4- are the A-phase output terminals for axis 3. Y0.6+ and Y0.6- are the A-phase output terminals for axis 4.<br>◆ Y0.1+ and Y0.1- are the B-phase output terminals for axis 1. Y0.3+ and Y0.3- are the B-phase output terminals for axis 2. Y0.5+ and Y0.5- are the B-phase output terminals for axis 3. Y0.7+ and Y0.7- are the B-phase output terminals for axis 4.<br>◆ Y0.0+ and Y0.0- are the CLR output terminals for axis 5. Y0.1+ and Y0.1- are the CLR output terminals for axis 6. | 1 MHz | 5 mA | 5 V |

7. **AH15PM-5A**

| Specifications \ Item | | Differential output | Transistor output |
|---|---|---|---|
| **Maximum exchange (working) frequency** | | 1 MHz | 200 kHz |
| **Output terminal** | | Y0.0~Y0.7 | Y0.8~Y0.11 |
| **Working voltage** | | 5 VDC | 5~30 VDC |
| **Maximum output current** | | 40 mA | 40 mA |
| **Isolation** | | Digital isolator | Optocoupler |
| **Current specifications** | **Resistance** | <25 mA | 0.5 A/output terminal (4 A/COM) |
| | **Inductance** | -- | 12 W (24 V DC) |
| | **Bulb** | -- | 2 W (24 V DC) |
| **Response time** | **OFF→ON** | 0.2 us | |
| | **ON→OFF** | | |
| **Overcurrent protection** | | Yes | No |

| Terminal | Description | Response characteristic | Maximum output | |
|---|---|---|---|---|
| | | | Current | Voltage |
| **Y0.8, Y0.9, Y0.10, and Y0.11** | 1. The high-speed pulse output terminals are transistors whose collectors are open collectors.<br>2. Functions of the terminals:<br>● Motion control:<br>◆ The terminals are the CLR output terminals for axis 1~axis 4.<br>● High-speed comparison and capture: The terminals can function as high-speed comparison output terminals. | 200 kHz | 15 mA | 24 V |
| **Y0.0+, Y0.0-, Y0.1+, Y0.1-, Y0.2+, Y0.2-, Y0.3+, Y0.3-, Y0.4+, Y0.4-, Y0.5+, Y0.5-, Y0.6+, Y0.6-, Y0.7+, and Y0.7-** | 1. They are differential output terminals.<br>2. Function of the terminals:<br>● Motion control:<br>◆ The terminals are for axis 1~axis 4.<br>◆ Y0.0+ and Y0.0- are the A-phase output terminals for axis 1. Y0.2+ and Y0.2- are the A-phase output terminals for axis 2. Y0.4+ and Y0.4- are the A-phase output terminals for axis 3. Y0.6+ and Y0.6- are the A-phase output terminals for axis 4.<br>◆ Y0.1+ and Y0.1- are the B-phase output terminals for axis 1. Y0.3+ and Y0.3- are the B-phase output terminals for axis 2. Y0.5+ and Y0.5- are the B-phase output terminals for axis 3. Y0.7+ and Y0.7- are the B-phase output terminals for axis 4.<br>◆ Y0.0+ and Y0.0- are the CLR output terminals for axis 5. Y0.1+ and Y0.1- are the CLR output terminals for axis 6. | 1 MHz | 5 mA | 5 V |

8. **AH05PM-5A**

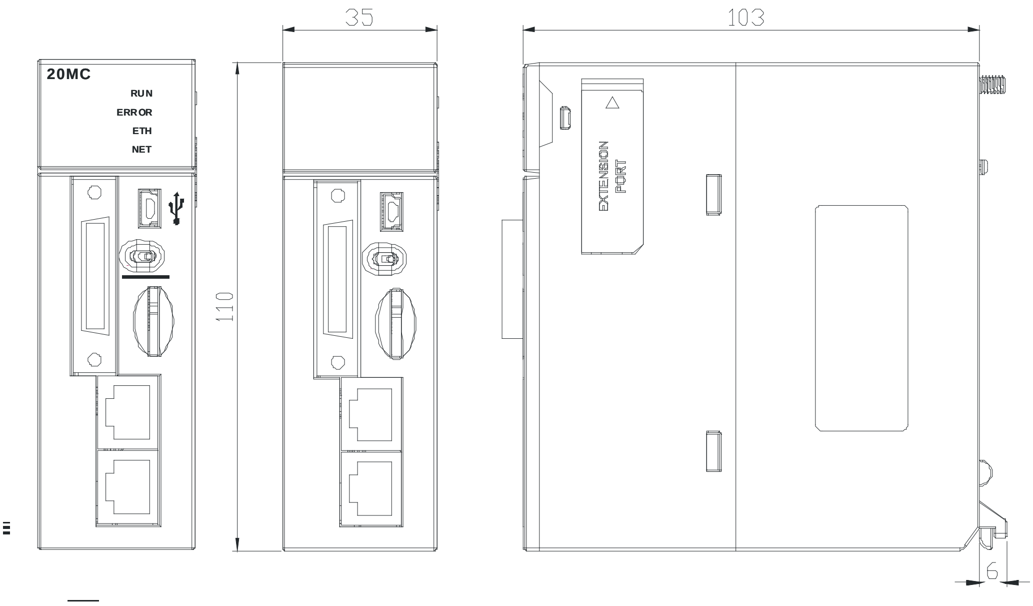
| Item / Specifications | | Differential output | Transistor output |
|---|---|---|---|
| **Maximum exchange (working) frequency** | | 1 MHz | 200 kHz |
| **Output terminal** | | Y0.0~Y0.3 | Y0.8~Y0.9 |
| **Working voltage** | | 5 V DC | 5~30 V DC |
| **Maximum output current** | | 40 mA | 40 mA |
| **Isolation** | | Digital isolator | Optocoupler |
| **Current specifications** | **Resistance** | <25 mA | 0.5 A/ output terminal (4 A/COM) |
| | **Inductance** | -- | 12 W (24 V DC) |
| | **Bulb** | -- | 2 W (24 V DC) |

| Specifications \ Item | | Differential output | Transistor output |
|---|---|---|---|
| **Response time** | **OFF→ON** | 0.2 us | |
| | **ON→OFF** | | |
| **Overcurrent protection** | | Yes | No |

| Terminal | Description | Response characteristic | Maximum output | |
|---|---|---|---|---|
| | | | Current | Voltage |
| **Y0.8 and Y0.9** | 1. The high-speed pulse output terminals are transistors whose collectors are open collectors.<br>2. Functions of the terminals:<br>● Motion control: Y0.8 is the CLR output terminal for axis 1, and Y0.9 is the CLR output terminal for axis 2.<br>● High-speed comparison and capture: The terminals can function as high-speed comparison output terminals. | 200 kHz | 15 mA | 24 V |
| **Y0.0+, Y0.0-, Y0.1+, Y0.1-, Y0.2+, Y0.2-, Y0.3+, and Y0.3-** | 1. They are differential output terminals.<br>2. Function of the terminals:<br>● Motion control:<br>◆ Y0.0+ and Y0.0- are the A-phase output terminals for axis 1. Y0.2+ and Y0.2- are the A-phase output terminals for axis 2.<br>◆ Y0.1+ and Y0.1- are the B-phase output terminals for axis 1. Y0.3+ and Y0.3- are the B-phase output terminals for axis 2. | 1 MHz | 5 mA | 5 V |

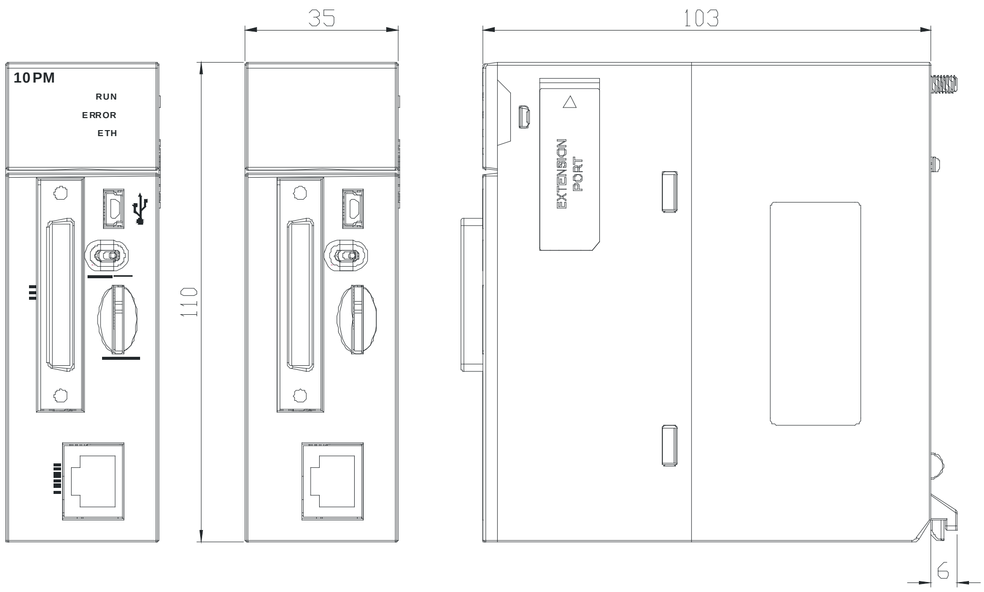
## 2.1.4 Dimensions

**9. AH20MC-5A**




Unit: mm

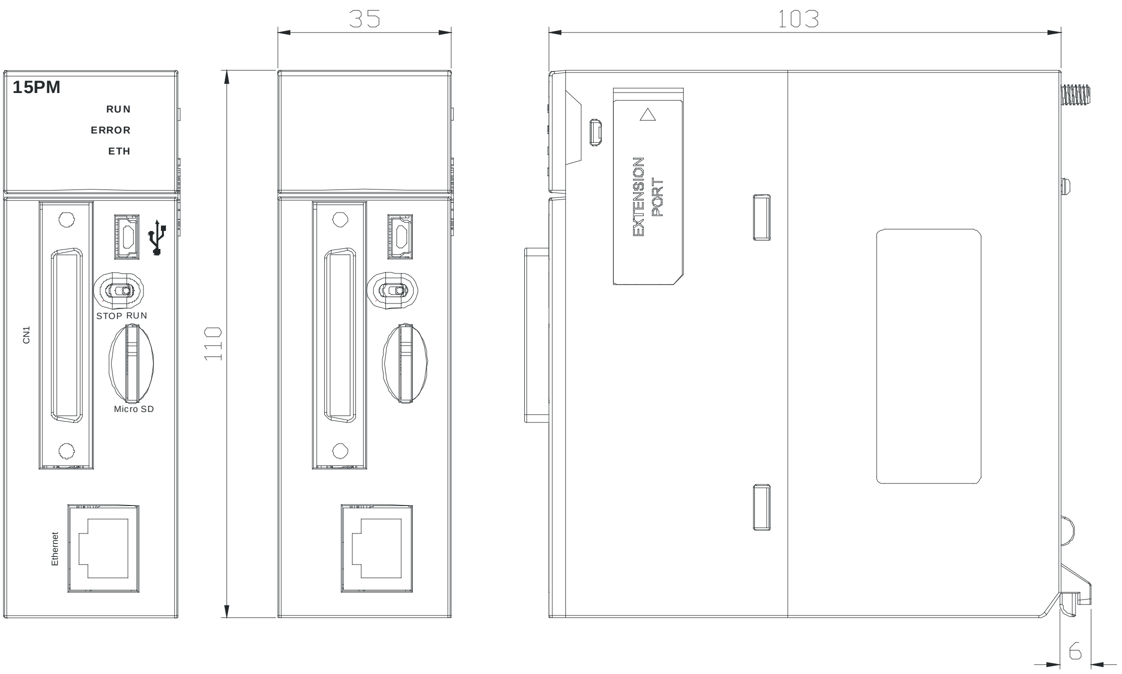
**10. AH10PM-5A**




Unit: mm

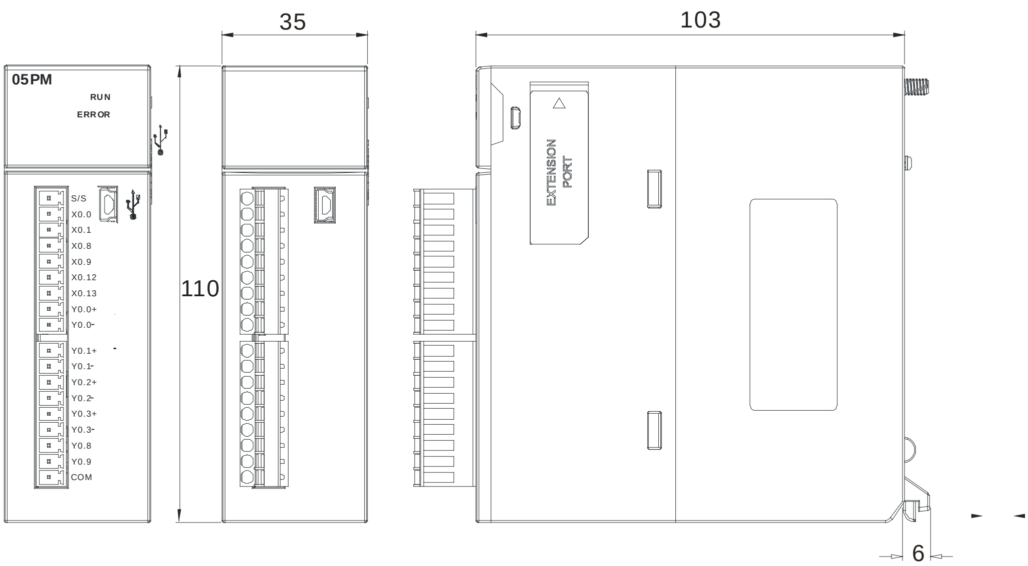
2

### 11. AH15PM-5A




Unit: mm

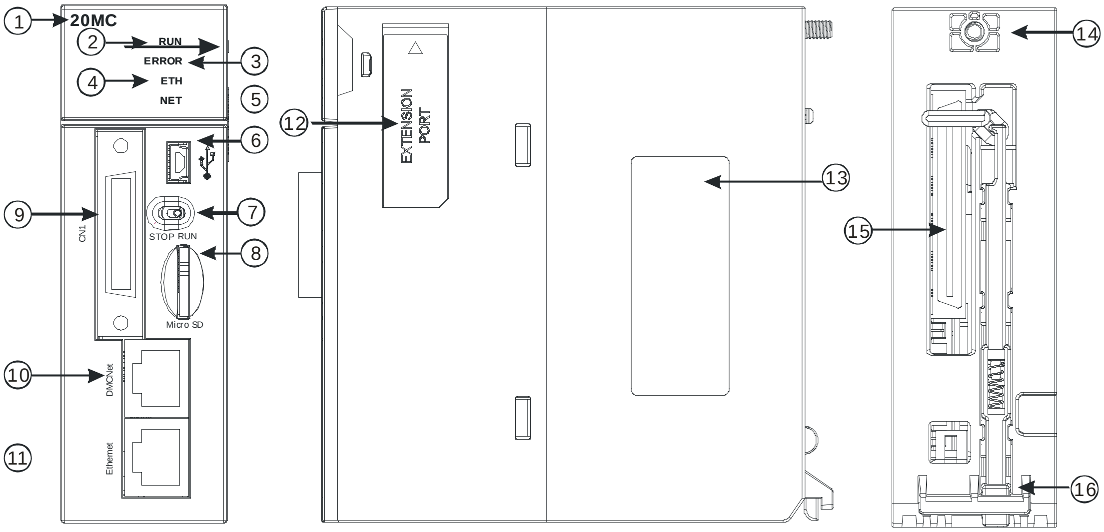
### 12. AH05PM-5A




Unit: mm

## 2.1.5 Profiles

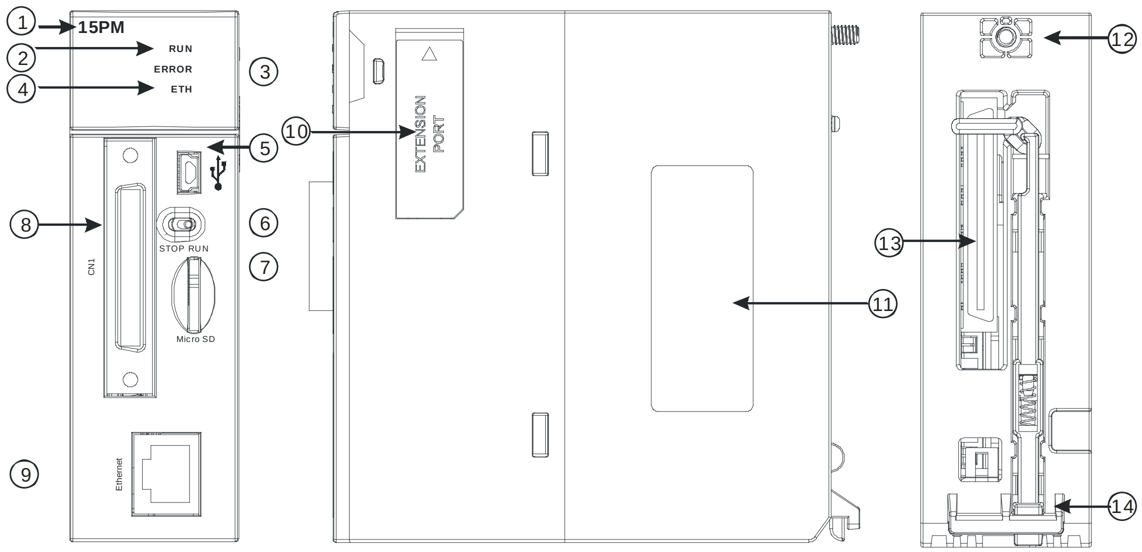
### 13. AH20MC-5A




| Number | Name | Description |
|---|---|---|
| 1 | Model name | Model name of the module |
| 2 | RUN LED indicator (green) | Operating status of the module<br>ON: The module is running.<br>OFF: The module stops running. |
| 3 | ERROR LED indicator (red) | Error status of the module<br>Blink: The module is abnormal. |
| 4 | Ethernet connection LED indicator (green) | Status of the Ethernet connection<br>ON: The Ethernet connection is being connected.<br>OFF: The Ethernet connection is disconnected. |
| 5 | DMCNET connection LED indicator (green) | Status of the DMCNET connection<br>ON: The DMCNET connection is being connected.<br>OFF: The DMCNET connection is disconnected. |
| 6 | USB port | Providing the mini USB communication interface |
| 7 | STOP/RUN switch | RUN: The user program is executed.<br>STOP: The execution of the user program stops. |
| 8 | SD slot | Providing the SD interface |
| 9 | Connector | Connecting the module and an I/O extension cable. |
| 10 | DMCNET port | Providing the DMCNET communication interface |
| 11 | Ethernet port | Providing the Ethernet communication interface |
| 12 | Extension port | For Used for updating the firmware |
| 13 | Label | Nameplate |
| 14 | Set screw | Used for securing the module |
| 15 | Connector | Connecting the module and a backplane |
| 16 | Projection | Used for securing the module |

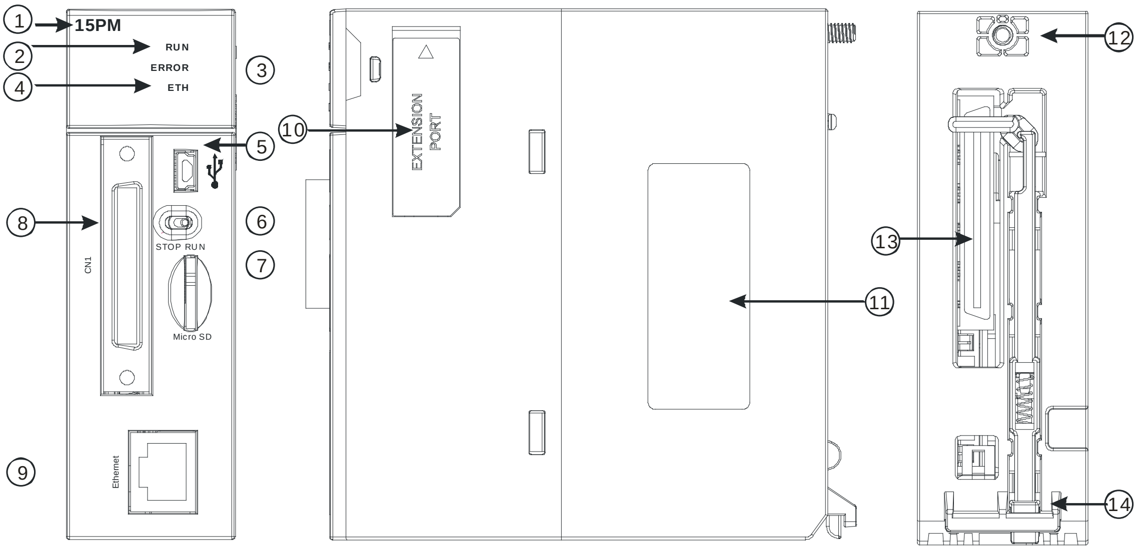
2

### 14. AH10PM-5A



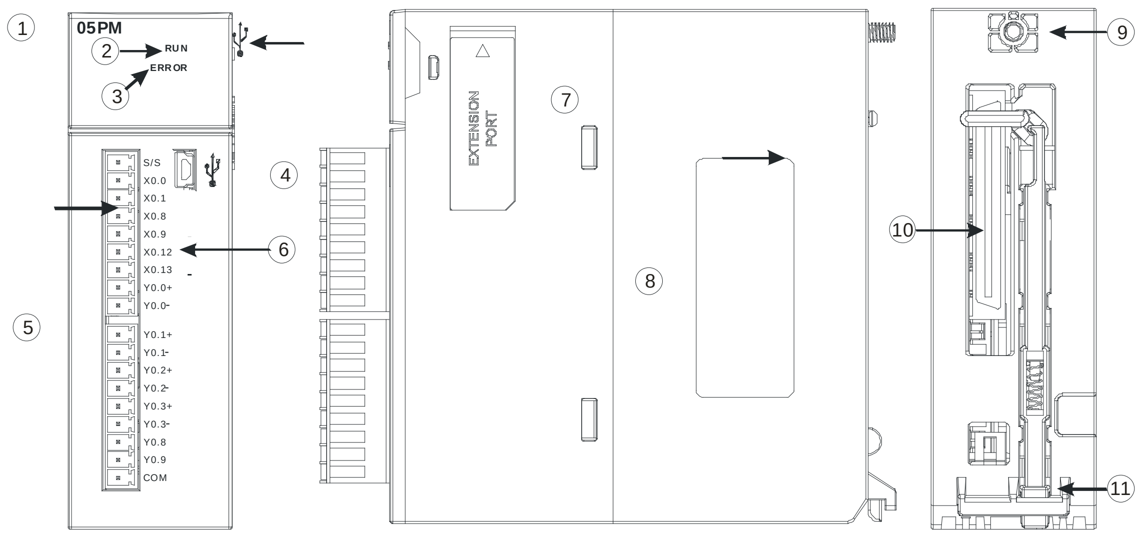

| Number | Name | Description |
|---|---|---|
| 1 | Model name | Model name of the module |
| 2 | RUN LED indicator (green) | Operating status of the module<br>ON: The module is running.<br>OFF: The module stops running. |
| 3 | ERROR LED indicator (red) | Error status of the module<br>Blink: The module is abnormal. |
| 4 | Ethernet connection LED indicator (green) | Status of the Ethernet connection<br>ON: The Ethernet connection is being connected.<br>OFF: The Ethernet connection is disconnected. |
| 5 | USB port | Providing the mini USB communication interface |
| 6 | RUN/STOP switch | RUN: The user program is executed.<br>STOP: The execution of the user program stops. |
| 7 | SD slot | Providing the SD interface |
| 8 | I/O Connector | Connectors for pulse input/output |
| 9 | Ethernet port | Providing the Ethernet communication interface |
| 10 | Extension port | Used for Used for updating the firmware |
| 11 | Label | Nameplate |
| 12 | Set screw | Used for securing the module |
| 13 | Connector | Connecting the module and a backplane |
| 14 | Projection | Used for securing the module |

15. **AH15PM-5A**




| Number | Name | Description |
|---|---|---|
| 1 | Model name | Model name of the module |
| 2 | RUN LED indicator (green) | Operating status of the module<br>ON: The module is running.<br>OFF: The module stops running. |
| 3 | ERROR LED indicator (red) | Error status of the module<br>Blink: The module is abnormal. |
| 4 | Ethernet connection LED indicator (green) | Status of the Ethernet connection<br>ON: The Ethernet connection is being connected.<br>OFF: The Ethernet connection is disconnected. |
| 5 | USB port | Providing the mini USB communication interface |
| 6 | STOP/RUN switch | RUN: The user program is executed.<br>STOP: The execution of the user program stops. |
| 7 | SD slot | Providing the SD interface |
| 8 | Connector | Connecting the module and an I/O extension cable |
| 9 | Ethernet port | Providing the Ethernet communication interface |
| 10 | Extension port | Used for updating the firmware |
| 11 | Label | Nameplate |
| 12 | Set screw | Used for securing the module |
| 13 | Connector | Connecting the module and a backplane |
| 14 | Projection | Used for securing the module |

- **AH05PM-5A**




| Number | Name | Description |
|---|---|---|
| 1 | Model name | Model name of the module |
| 2 | RUN LED indicator (green) | Operating status of the module<br>ON: The module is running.<br>OFF: The module stops running. |
| 3 | ERROR LED indicator (red) | Error status of the module<br>Blink: The module is abnormal. |
| 4 | USB port | Providing the mini USB communication interface |
| 5 | Terminals | Input/Output terminals |
| 6 | Arrangement of the input/output terminals | Arrangement of the terminals |
| 7 | Extension port | Used for updating the firmware |
| 8 | Label | Nameplate |
| 9 | Set screw | Used for securing the module |
| 10 | Connector | Connecting the module and a backplane |
| 11 | Projection | Used for securing the module |

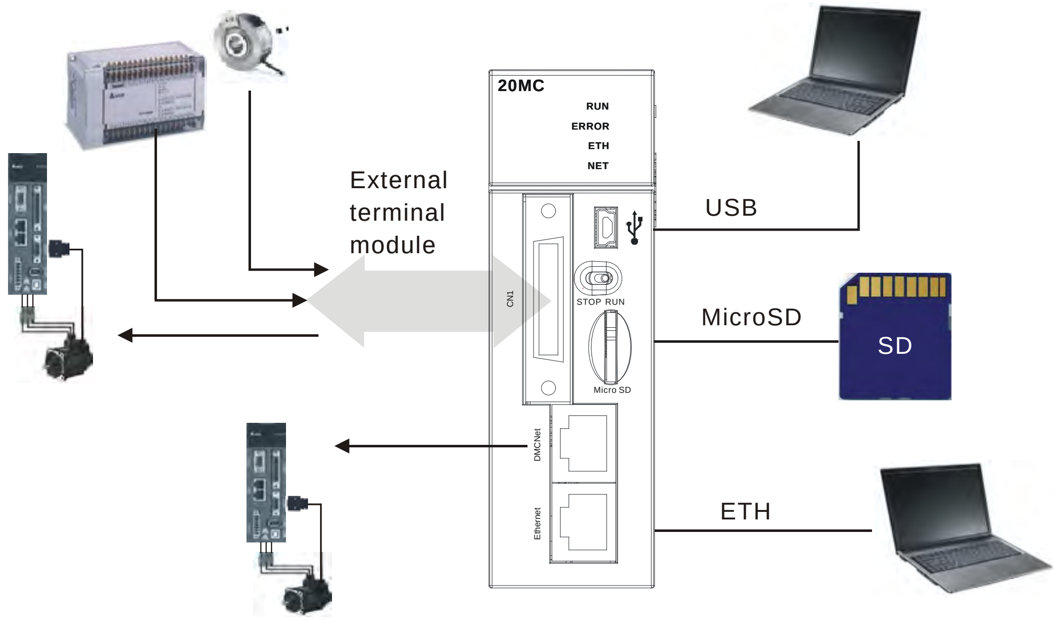
## 2.2 Wiring

An AH500 series motion control module is an open-type device. It has to be installed in a control box which is free from dust, moisture, and shock/vibration. In order to prevent people who are not maintenance men from operating the device, protective measures are required (e.g. Users need a special tool or a key to open the control box).

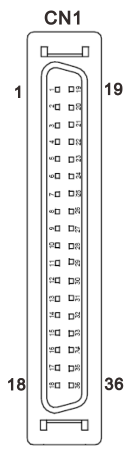
An AC power supply can not be connected to input/output terminals, otherwise the device will be seriously damaged. Before users power the device, they have to check the wiring of the power supply. In order to increase the immunity against noise, the ground terminal ⏚ on the device must be grounded correctly.

## 2.2.1 I / O Extension Cables, and External Terminal Modules

### 1. External devices for AH20MC-5A

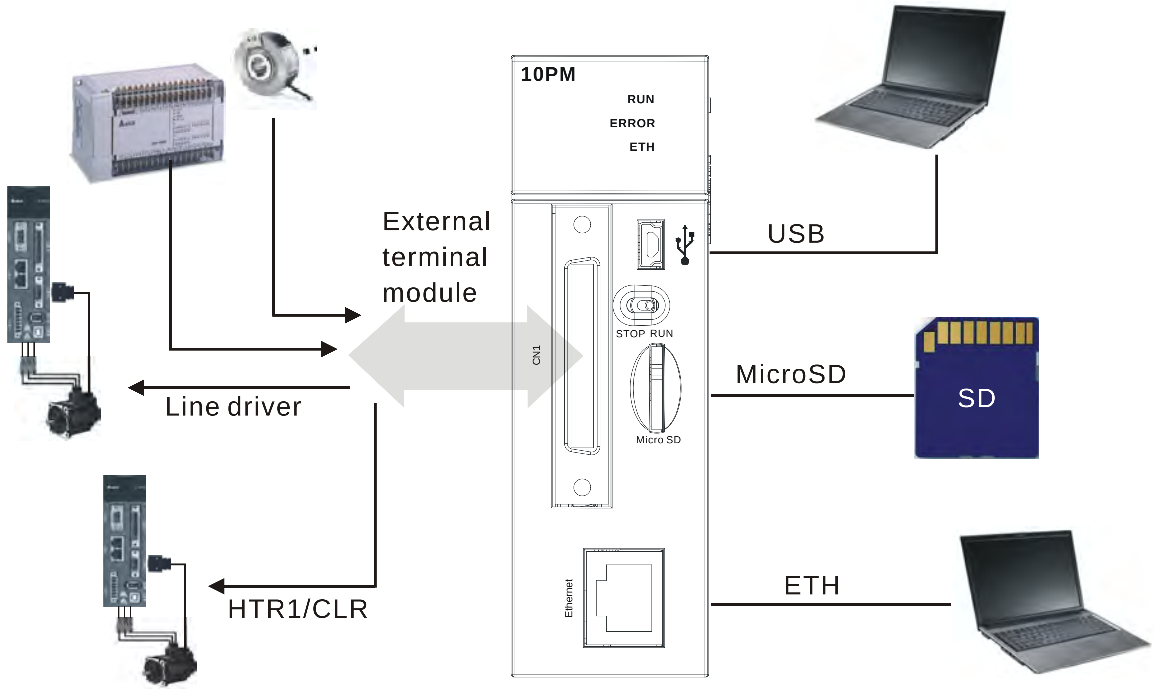



### 2. Connector on AH20MC-5A

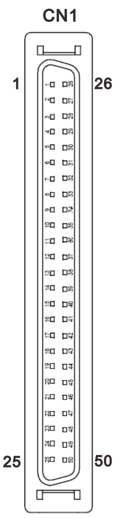



| Pin | Terminal | Function | | Pin | Terminal | Function | |
|---|---|---|---|---|---|---|---|
| | | Pulse | Count | | | Pulse | Count |
| 1 | C3 | - | COM3 | 19 | Y0.11 | - | Out3 |
| 2 | C2 | - | COM2 | 20 | Y0.10 | - | Out2 |
| 3 | C1 | - | COM1 | 21 | Y0.9 | - | Out1 |
| 4 | C0 | - | COM0 | 22 | Y0.8 | - | Out0 |
| 5 | NC | - | - | 23 | NC | - | - |
| 6 | NC | - | - | 24 | NC | - | - |
| 7 | X0.3- | - | Reset3-/Reset5- | 25 | X0.3+ | - | Reset3+/Reset5+ |
| 8 | X0.15- | DOG3- | CntB3-/CntB5+ | 26 | X0.15+ | DOG3+ | CntB3+/CntB5+ |
| 9 | X0.14- | DOG2- | CntA3-/CntA5+ | 27 | X0.14+ | DOG2+ | CntA3+/CntA5+ |
| 10 | X0.2- | - | Reset2-/Reset4- | 28 | X0.2+ | - | Reset2+/Reset4+ |
| 11 | X0.13- | DOG1- | CntB2-/CntB4- | 29 | X0.13+ | DOG1+ | CntB2+/CntB4+ |
| 12 | X0.12- | DOG0- | CntA2-/CntA4- | 30 | X0.12+ | DOG0+ | CntA2+/CntA4+ |
| 13 | X0.1- | - | Reset1- | 31 | X0.1+ | - | Reset1+ |
| 14 | X0.11- | DOG5- | CntB1- | 32 | X0.11+ | DOG5+ | CntB1+ |
| 15 | X0.10- | DOG4- | CntA1- | 33 | X0.10+ | DOG4+ | CntA1+ |
| 16 | X0.0- | - | Reset0- | 34 | X0.0+ | - | Reset0+ |
| 17 | X0.9- | MPGB- | CntB0- | 35 | X0.9+ | MPGB+ | CntB0+ |
| 18 | X0.8- | MPGA- | CntA0- | 36 | X0.8+ | MPGA+ | CntA0+ |

3. **External devices for AH10PM-5A**

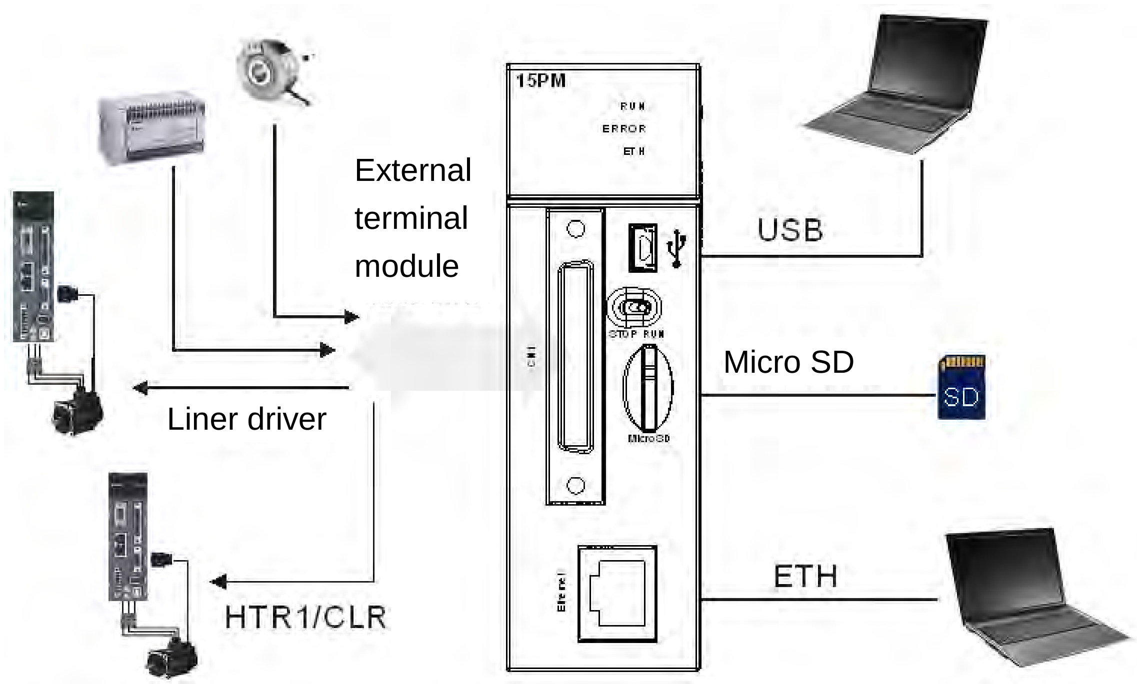



4. **Connector on AH10PM-5A**

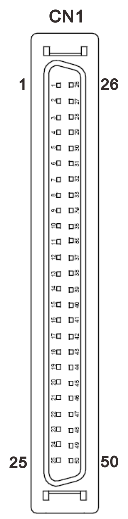



| Pin | Terminal | Function | | Pin | Terminal | Function | |
|---|---|---|---|---|---|---|---|
| | | Pulse | Count | | | Pulse | Count |
| 1 | C3 | COM3 | - | 26 | Y0.11 | CLR3/B5 | - |
| 2 | C2 | COM2 | - | 27 | Y0.10 | CLR2/A5 | - |
| 3 | C1 | COM1 | - | 28 | Y0.9 | CLR1/B4 | - |
| 4 | C0 | COM0 | - | 29 | Y0.8 | CLR0/A4 | - |
| 5 | NC | - | - | 30 | NC | - | - |
| 6 | Y0.7- | B3- | - | 31 | Y0.7+ | B3+ | - |
| 7 | Y0.6- | A3- | - | 32 | Y0.6+ | A3+ | - |
| 8 | Y0.5- | B2- | - | 33 | Y0.5+ | B2+ | - |
| 9 | Y0.4- | A2- | - | 34 | Y0.4+ | A2+ | - |
| 10 | Y0.3- | B1- | - | 35 | Y0.3+ | B1+ | - |
| 11 | Y0.2- | A1- | - | 36 | Y0.2+ | A1+ | - |
| 12 | Y0.1- | B0-/CLR5- | - | 37 | Y0.1+ | B0+/CLR5+ | - |
| 13 | Y0.0- | A0-/CLR4- | - | 38 | Y0.0+ | A0+/CLR4+ | - |
| 14 | NC | - | - | 39 | NC | - | - |
| 15 | NC | - | - | 40 | S/S | S/S | S/S |
| 16 | X0.15 | DOG3 | CntB3/CntB5 | 41 | X0.14 | DOG2 | CntB3/CntA5 |
| 17 | X0.13 | DOG1 | CntB2/CntB4 | 42 | X0.12 | DOG0 | CntA2/CntA4 |
| 18 | X0.11 | DOG5 | CntB1 | 43 | X0.10 | DOG4 | CntA1 |
| 19 | X0.9 | MPGB | CntB0 | 44 | X0.8 | MPGA | CntA0 |
| 20 | NC | - | - | 45 | NC | - | - |
| 21 | NC | - | - | 46 | NC | - | - |
| 22 | X0.3- | Pg3- | Reset3-/Reset5- | 47 | X0.3+ | Pg3+ | Reset3+/Reset5+ |
| 23 | X0.2- | Pg2- | Reset2-/Reset4- | 48 | X0.2+ | Pg2+ | Reset2+/Reset4+ |
| 24 | X0.1- | Pg1- | Reset1- | 49 | X0.1+ | Pg1+ | Reset1+ |
| 25 | X0.0- | Pg0- | Reset0- | 50 | X0.0+ | Pg0+ | Reset0+ |

5. **External devices for AH15PM-5A**



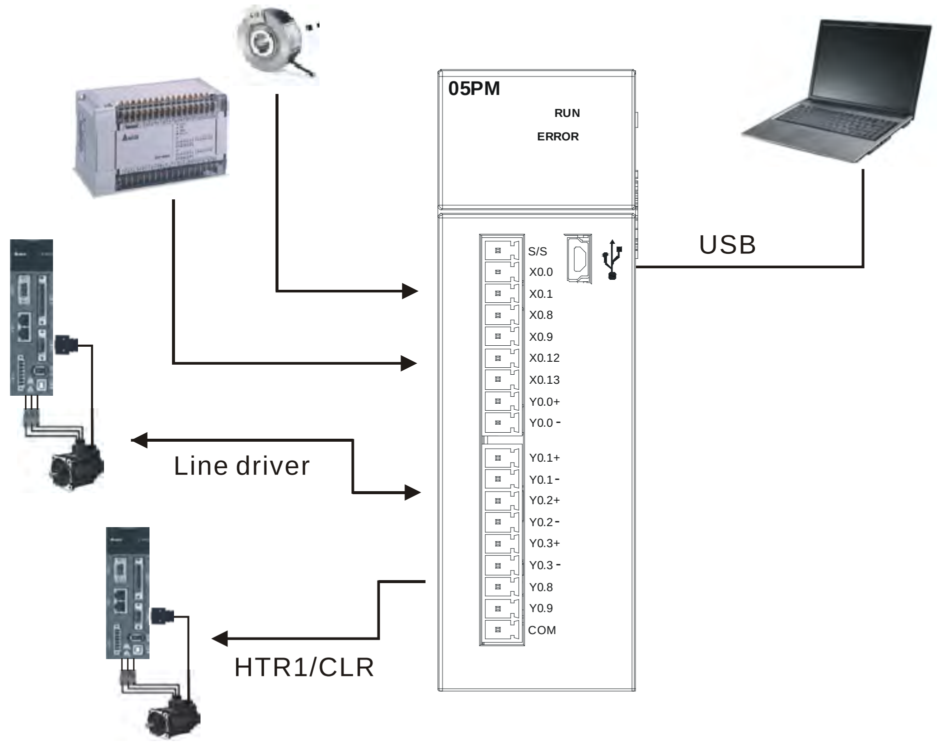

6. **Connector on AH15PM-5A**




| Pin | Terminal | Function | | Pin | Terminal | Function | |
|---|---|---|---|---|---|---|---|
| | | Pulse | Count | | | Pulse | Count |
| 1 | Y0.11 | CLR3 | - | 26 | Y0.10 | CLR2 | - |
| 2 | Y0.9 | CLR1 | - | 27 | Y0.8 | CLR0 | |
| 3 | COM | COM | - | 28 | Y0.7+ | B3+ | - |
| 4 | Y0.7- | B3- | - | 29 | Y0.6+ | A3+ | - |
| 5 | Y0.6- | A3- | - | 30 | Y0.5+ | B2+ | - |
| 6 | Y0.5- | B2- | - | 31 | Y0.4+ | A2+ | - |
| 7 | Y0.4- | A2- | - | 32 | Y0.3+ | B1+ | - |
| 8 | Y0.3- | B1- | - | 33 | Y0.2+ | A1+ | - |
| 9 | Y0.2- | A1- | - | 34 | Y0.1+ | B0+ | - |
| 10 | Y0.1- | B0- | - | 35 | Y0.0+ | A0+ | - |
| 11 | Y0.0- | A0- | - | 36 | S/S | S/S | S/S |
| 12 | X1.5 | CHG3 | - | 37 | X1.4 | CHG2 | - |
| 13 | X1.3 | CHG1 | - | 38 | X1.2 | CHG0 | - |
| 14 | X1.1 | LSN3 | - | 39 | X1.0 | LSP3 | - |
| 15 | X0.15 | LSN2 | CntB3/CntB5 | 40 | X0.14 | LSP2 | CntB3/CntA5 |
| 16 | X0.13 | LSN1 | CntB2/CntB4 | 41 | X0.12 | LSP1 | CntA2/CntA4 |
| 17 | X0.11 | LSN0 | CntB1 | 42 | X0.10 | LSP0 | CntA1 |
| 18 | X0.9- | MPGB- | CntB0- | 43 | X0.9+ | MPGB+ | CntB0+ |
| 19 | X0.8- | MPGA- | CntA0- | 44 | X0.8+ | MPGA+ | CntA0+ |
| 20 | X0.7 | DOG3 | - | 45 | X0.6 | DOG2 | - |
| 21 | X0.5 | DOG1 | - | 46 | X0.4 | DOG0 | - |
| 22 | X0.3- | Pg3- | Reset3-/Reset5- | 47 | X0.3+ | Pg3+ | Reset3+/Reset5+ |
| 23 | X0.2- | Pg2- | Reset2-/Reset4- | 48 | X0.2+ | Pg2+ | Reset2+/Reset4+ |
| 24 | X0.1- | Pg1- | Reset1- | 49 | X0.1+ | Pg1+ | Reset1+ |
| 25 | X0.0- | Pg0- | Reset0- | 50 | X0.0+ | Pg0+ | Reset0+ |

2

7. **External devices for AH05PM-5A**

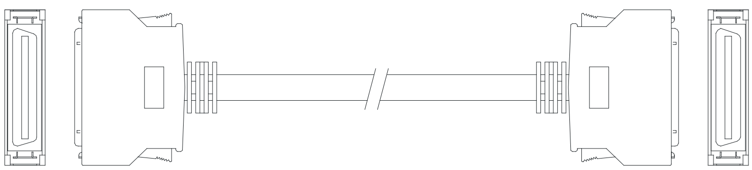



8. **Terminals on AH05PM-5A**

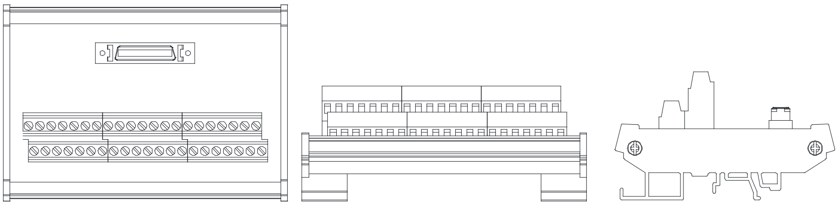
| Terminal | Function | | Terminal | Function | |
|---|---|---|---|---|---|
| | Pulse | Count | | Pulse | Count |
| S/S | S/S | S/S | Y0.1+ | B0+ | - |
| X0.0 | PG0 | Reset0 | Y0.1- | B0- | - |
| X0.1 | PG1 | - | Y0.2+ | A1+ | - |
| X0.8 | MPGA | CntA0 | Y0.2- | A1- | - |
| X0.9 | MPGB | CntB0 | Y0.3+ | B1+ | - |
| X0.12 | DOG0 | - | Y0.3- | B1- | - |
| X0.13 | DOG1 | - | Y0.8 | CLR0 | - |
| Y0.0+ | A0+ | - | Y0.9 | CLR1 | - |
| Y0.0- | A0- | - | COM | - | - |

An I/O extension cable connects the connector on an AH500 series motion control module to an external terminal module. Users can wire terminal blocks on the external terminal module.

- I/O extension cable DVPACAB7D10/DVPACAB7E10
  DVPACAB7D10 (36 pins): I/O extension cable for AH04HC-5A/AH20MC-5A
  DVPACAB7E10 (50 pins): I/O extension cable for AH10PM-5A

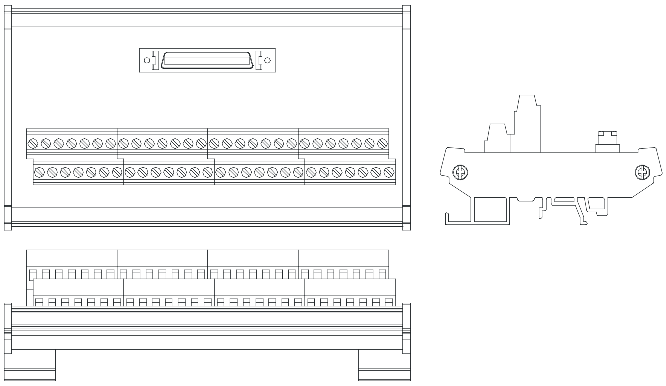


■ External terminal module for AH20MC-5A: DVPAETB-IO16C



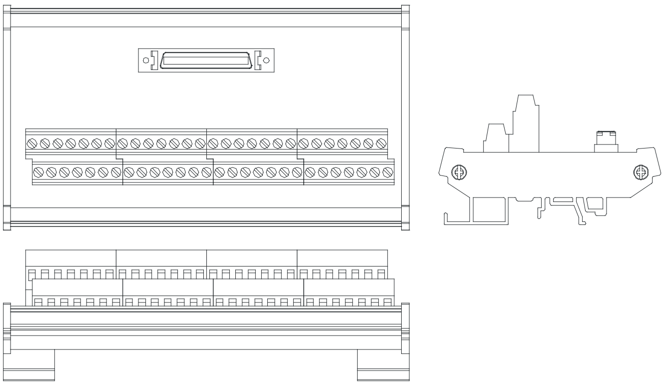
| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C3 | C2 | C1 | C0 | NC | NC | X0.3- | X0.15- | X0.14- | X0.2- | X0.13- | X0.12- | X0.1- | X0.11- | X0.10- | X0.0- | X0.9- | X0.8- | 24G | 24G | FE |
| Y0.11 | Y0.10 | Y0.9 | Y0.8 | NC | NC | X0.3+ | X0.15+ | X0.14+ | X0.2+ | X0.13+ | X0.12+ | X0.1+ | X0.11+ | X0.10+ | X0.0+ | X0.9+ | X0.8+ | NC | 24V | 24V |

■ External terminal module for AH10PM-5A: DVPAETB-IO24C



| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1st from the upper left** | C3 | C2 | C1 | C0 | N/C | Y0.7- | Y0.6- | Y0.5- | Y0.4- | Y0.3- | Y0.2- | Y0.1- | Y0.0- | N/C |
| **15th from the upper left** | N/C | X0.15 | X0.13 | X0.11 | X0.9 | N/C | N/C | X0.3- | X0.2- | X0.1- | X0.0- | 24G | 24G | FE |
| **1st from the lower left** | Y0.11 | Y0.10 | Y0.9 | Y0.8 | N/C | Y0.7+ | Y0.6+ | Y0.5+ | Y0.4+ | Y0.3+ | Y0.2+ | Y0.1+ | Y0.0+ | N/C |
| **15th from the lower left** | S/S | X0.14 | X0.12 | X0.10 | X0.8 | N/C | N/C | X0.3+ | X0.2+ | X0.1+ | X0.0+ | N/C | 24V | 24V |

■ External terminal module for AH15PM-5A: DVPAETB-IO34C



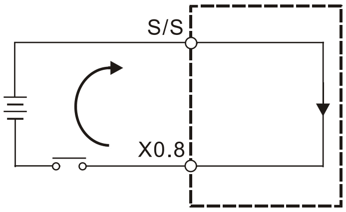
| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1st from the upper left** | Y0.11 | Y0.9 | COM | Y0.7- | Y0.6- | Y0.5- | Y0.4- | Y0.3- | Y0.2- | Y0.1- | Y0.0- | X1.5 | X1.3 | X1.1 |
| **15th from the upper left** | X0.15 | X0.13 | X0.11 | X0.9- | X0.8- | X0.7 | X0.5 | X0.3- | X0.2- | X0.1- | X0.0- | 24G | 24G | FE |
| **1st from the lower left** | Y0.10 | Y0.8 | Y0.7+ | Y0.6+ | Y0.5+ | Y0.4+ | Y0.3+ | Y0.2+ | Y0.1+ | Y0.0+ | S/S | Y1.4 | Y1.2 | Y1.0 |
| **15th from the lower left** | X0.14 | X0.12 | X0.10 | X0.9+ | X0.8+ | X0.6 | X0.4 | X0.3+ | X0.2+ | X0.1+ | X0.0+ | N/C | 24V | 24V |

## 2.2.2 Wiring Input Terminals

Input signals are direct-current power inputs. Sinking and sourcing are the current driving capability of a circuit. They are defined as follows.

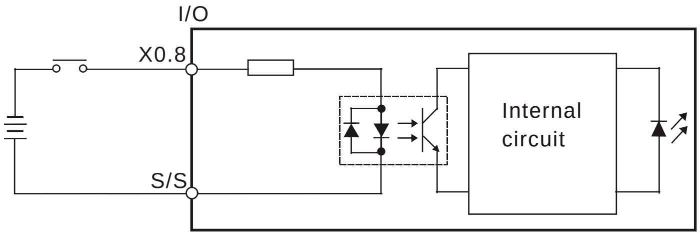
● **Sinking**

Direct current



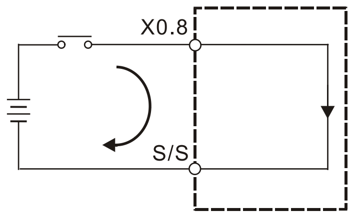

Current flows into the common terminal S/S.

**Equivalent circuit of the input circuit**



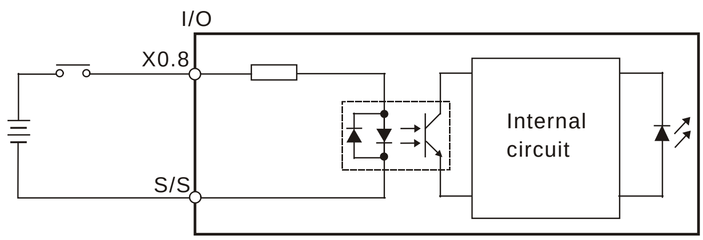
- **Sourcing**

Direct current




Current flows from the common terminal S/S.

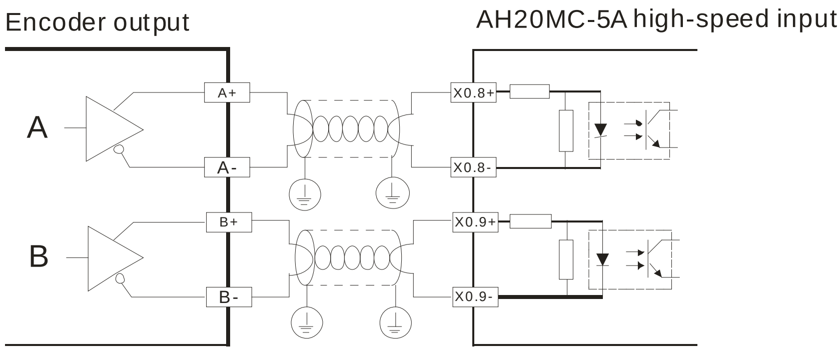
**Equivalent circuit of the input circuit**




- **Wiring differential input terminals**

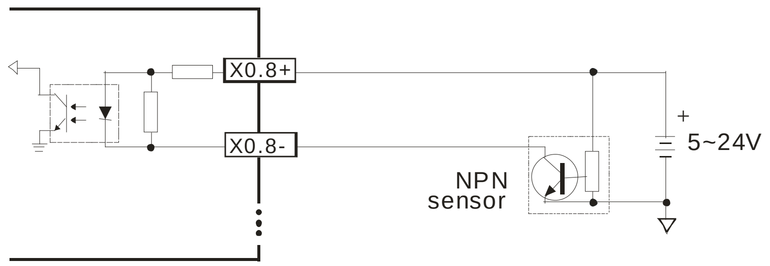
For example, the direct-current signals ranging in voltage from 5 V to 24 V can pass through the high-speed input terminals X0.8+~X0.15+, and X0.8-~X0.15- on AH20MC-5A. The frequency of input signals can be up to 200 kHz. These high-speed input terminals are connected to a differential (two-wire) line driver.

The wiring of differential input terminals is shown below. (The wiring is used for high speed and noise):

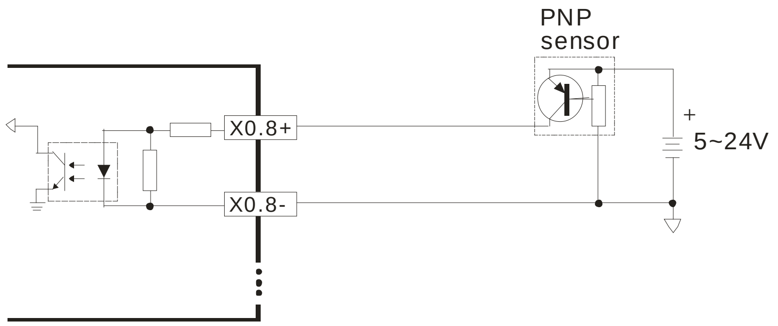


If the frequency of input signals is less than 50 kHz and there is not much noise, these high-speed input terminals can be connected to a direct-current (sinking/sourcing) power supply whose voltage is in the range of 5 V to 24 V, as shown below. Take AH20MC-5A for instance.

Sinking:

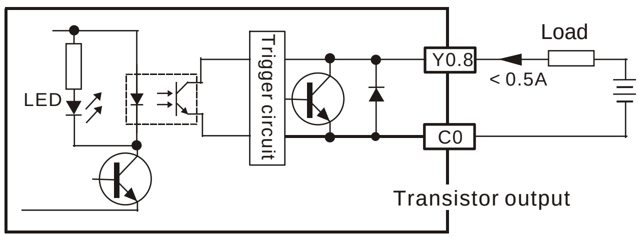


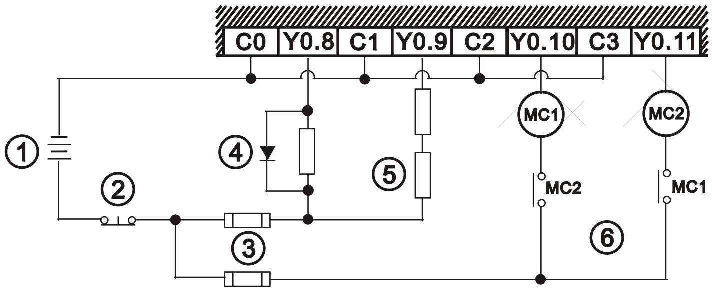

Sourcing:



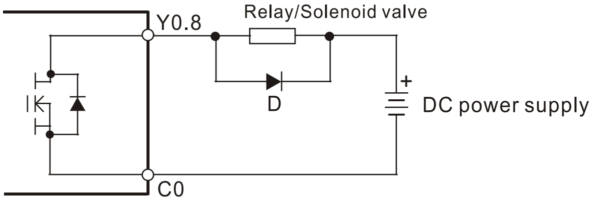
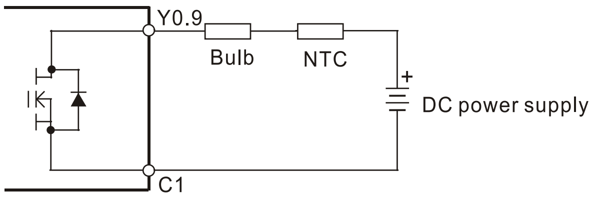
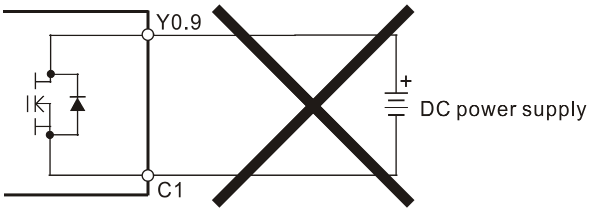

## 2.2.3 Wiring Output Terminals

1. Transistor output circuit



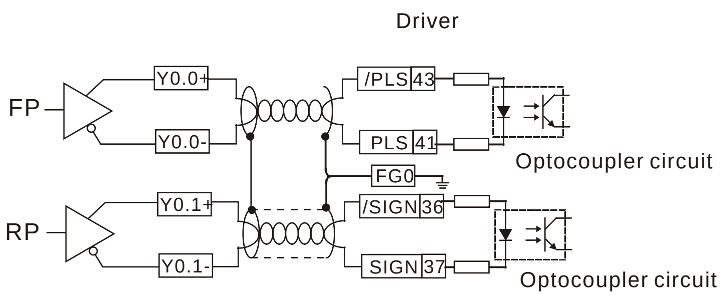

| | | | | | |
|---|---|---|---|---|---|
| ① | Direct-current power supply | ② | Emergency stop | ③ | Fuse |

<table>
<tr><td>④</td><td>The output terminals of a transistor module are open-collector output terminals. If Y0.8 is a pulse train output terminal of a transistor module, the output current passing through its output pull-up resistor must be greater than 0.1 A to ensure that the transistor module operates normally.<br>A relay or a solenoid valve is used as a DC load. A diode is connected in parallel to absorb the surge voltage which occurs when the load is OFF.<br><br><br>D: 1N4001 diode</td></tr>
<tr><td>⑤</td><td>A bulb (incandescent lamp) is used as a DC load. A thermistor is connected in series to absorb the surge current which occurs when the load is ON.<br><br><br>NTC: 10Ω<br><br>Y0.9 can not be connected to a power supply directly. It must be connected to a load.<br><br></td></tr>
<tr><td>⑥</td><td>Mutually exclusive output: For example, Y0.10 controls the clockwise rotation of the motor, and Y0.11 controls the counterclockwise rotation of the motor. The interlock circuit which is formed, and the program in the PLC ensure that there are protective measures if an abnormal condition occurs.</td></tr>
</table>

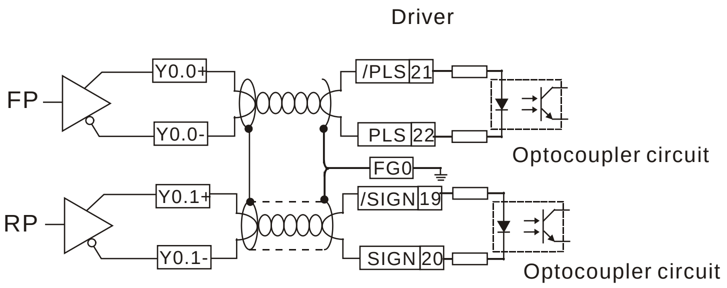
2. Wiring differential output terminals

   Take AH10PM-5A for instance. The wiring of the differential output terminals on AH10PM-5A is described below.

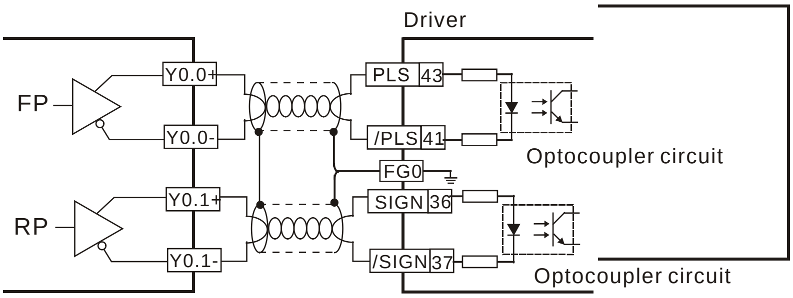
- Wiring differential output terminals on AH10PM-5A, and an ASDA-A/ASDA-A+/ASDA-A2 series AC servo drive



- Wiring differential output terminals on AH10PM-5A, and an ASDA-B series AC servo drive



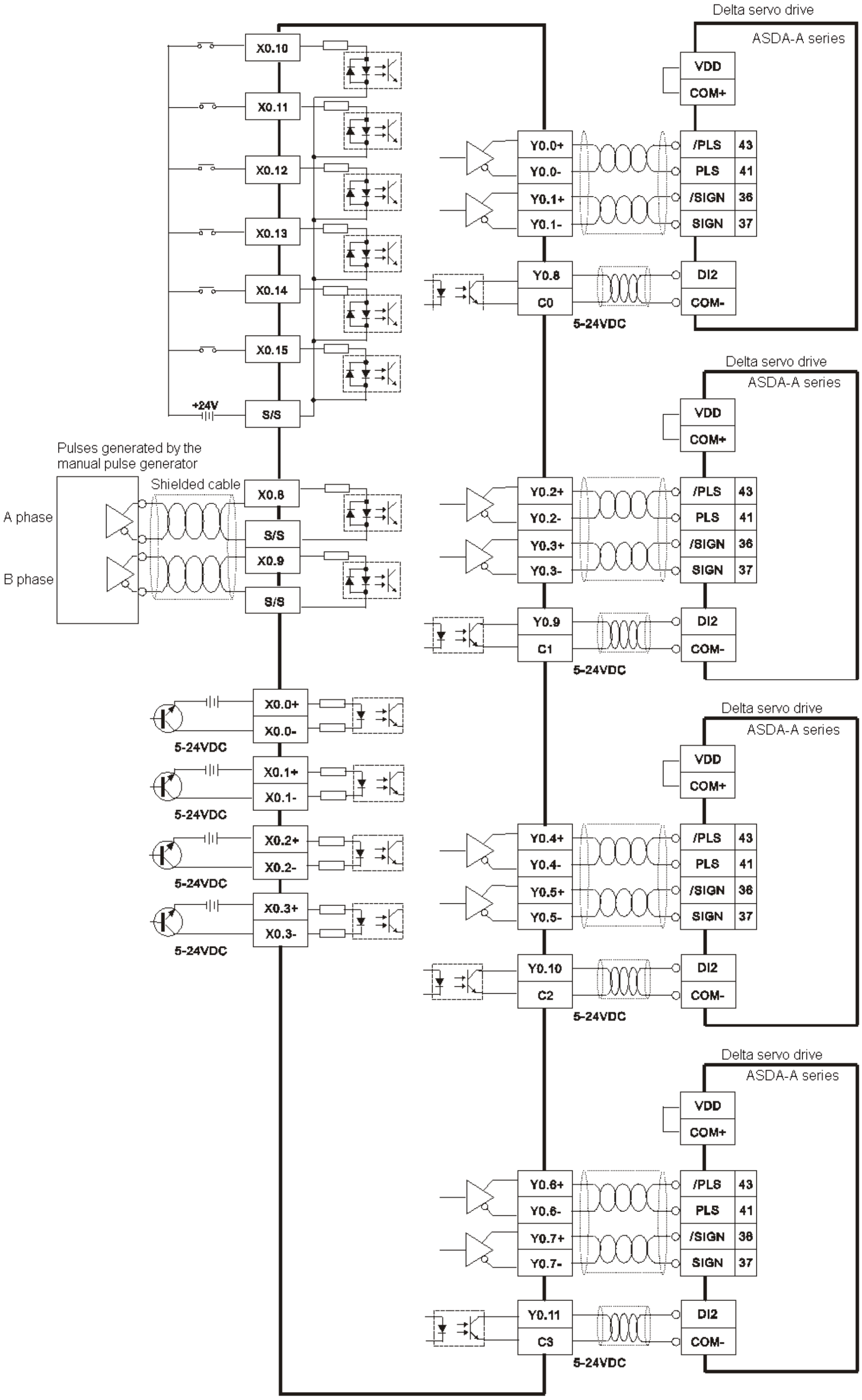

- Wiring differential output terminals on AH10PM-5A, and an ASDA-AB series AC servo drive



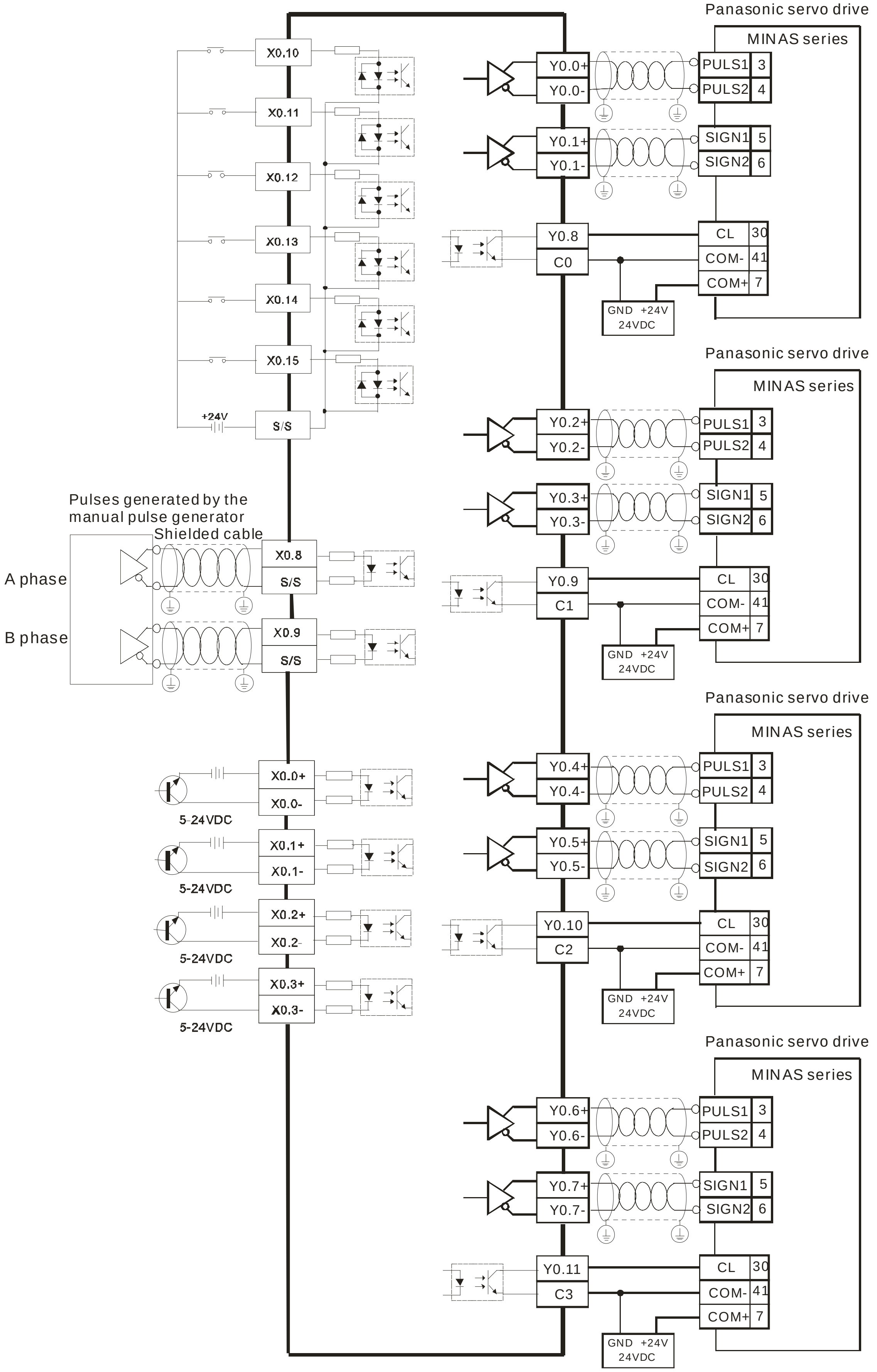
## 2.2.4 Wiring AH10PM-5A and an Inferior Servo Drive

Take AH10PM-5A for instance. The wiring of AH10PM-5A and an inferior servo drive is described below.

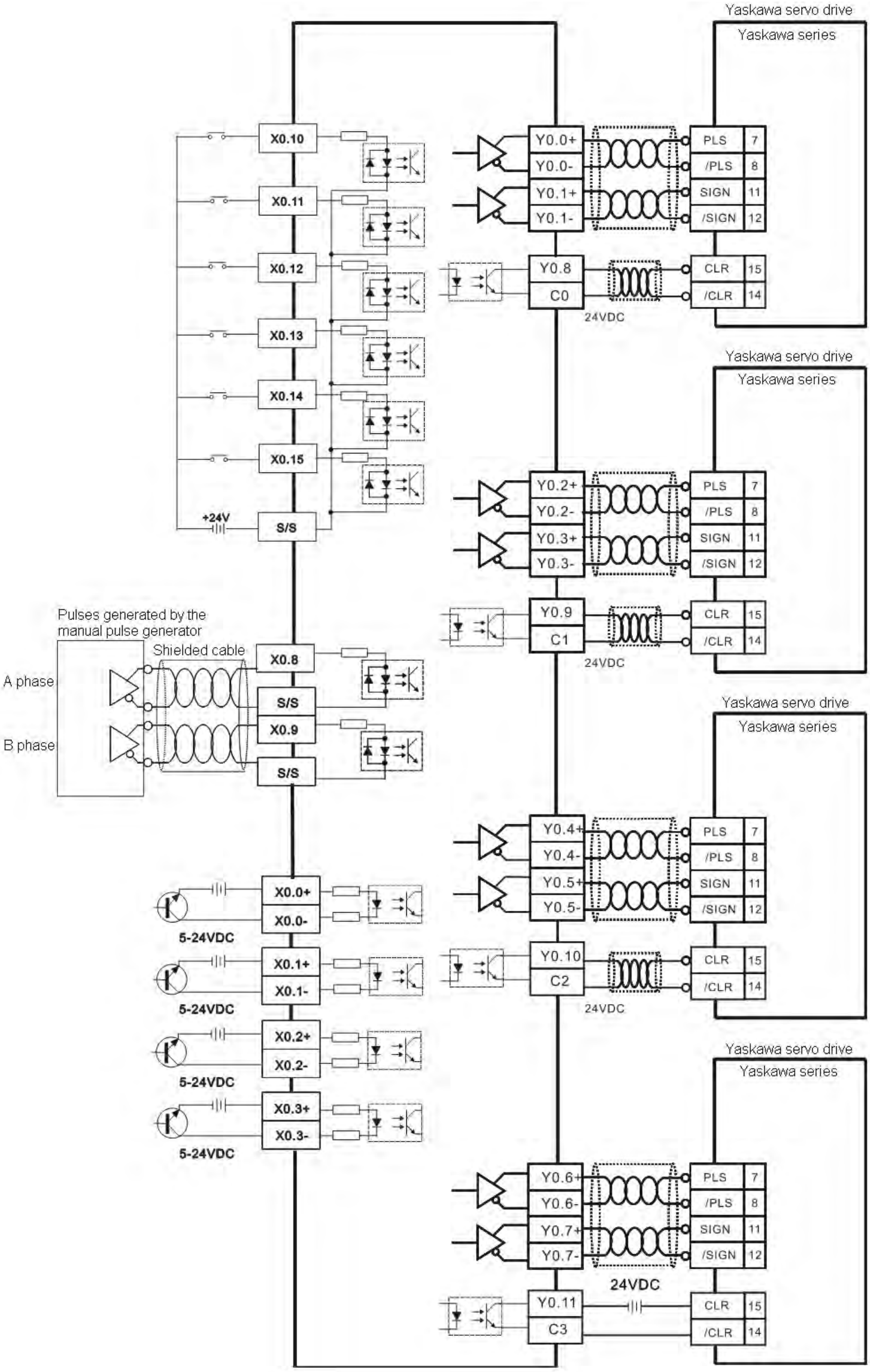
Wiring AH10PM-5A and a Delta ASDA-A series AC servo drive:



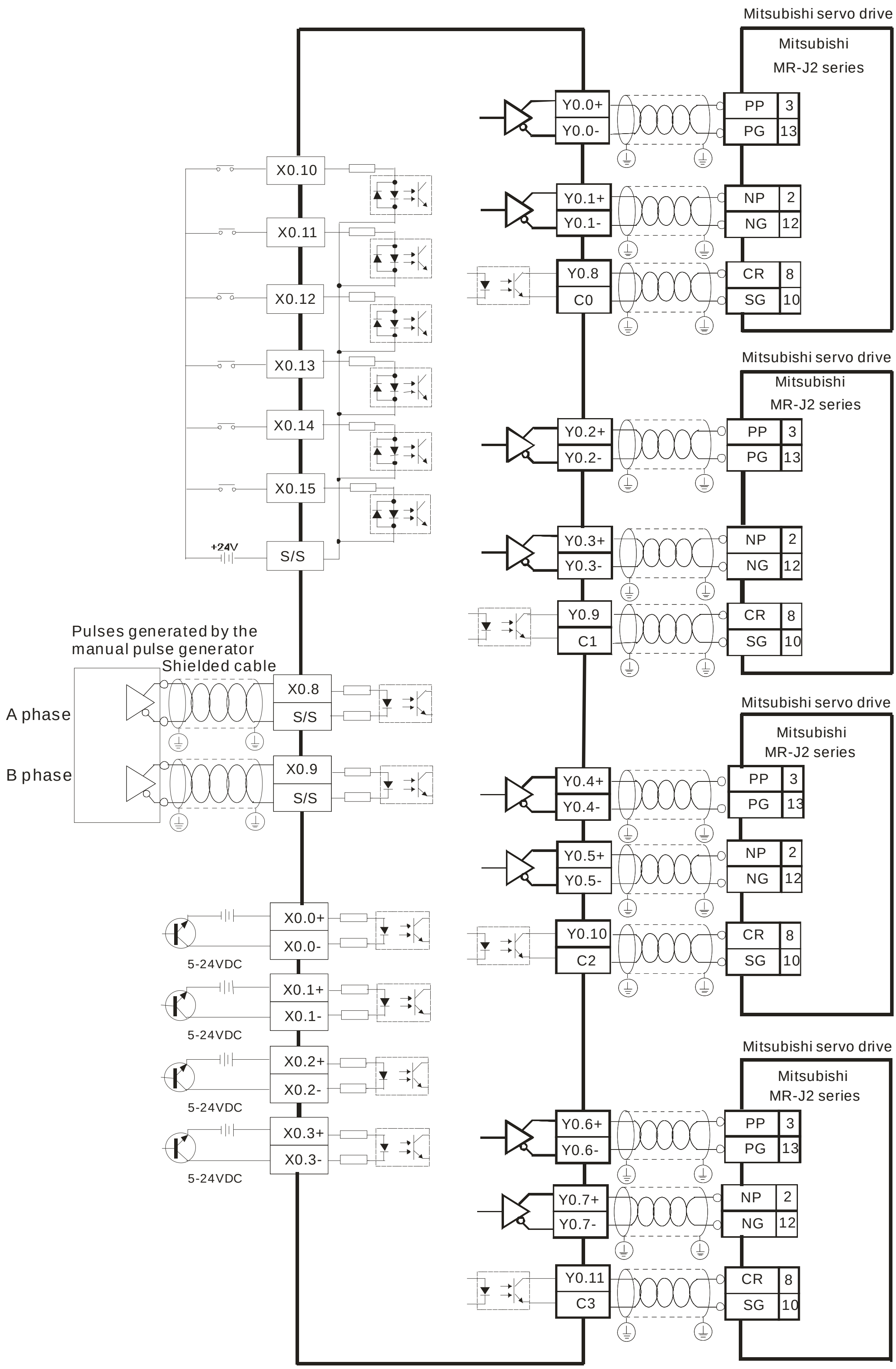
Wiring AH10PM-5A and a Panasonic MINAS series servo drive:



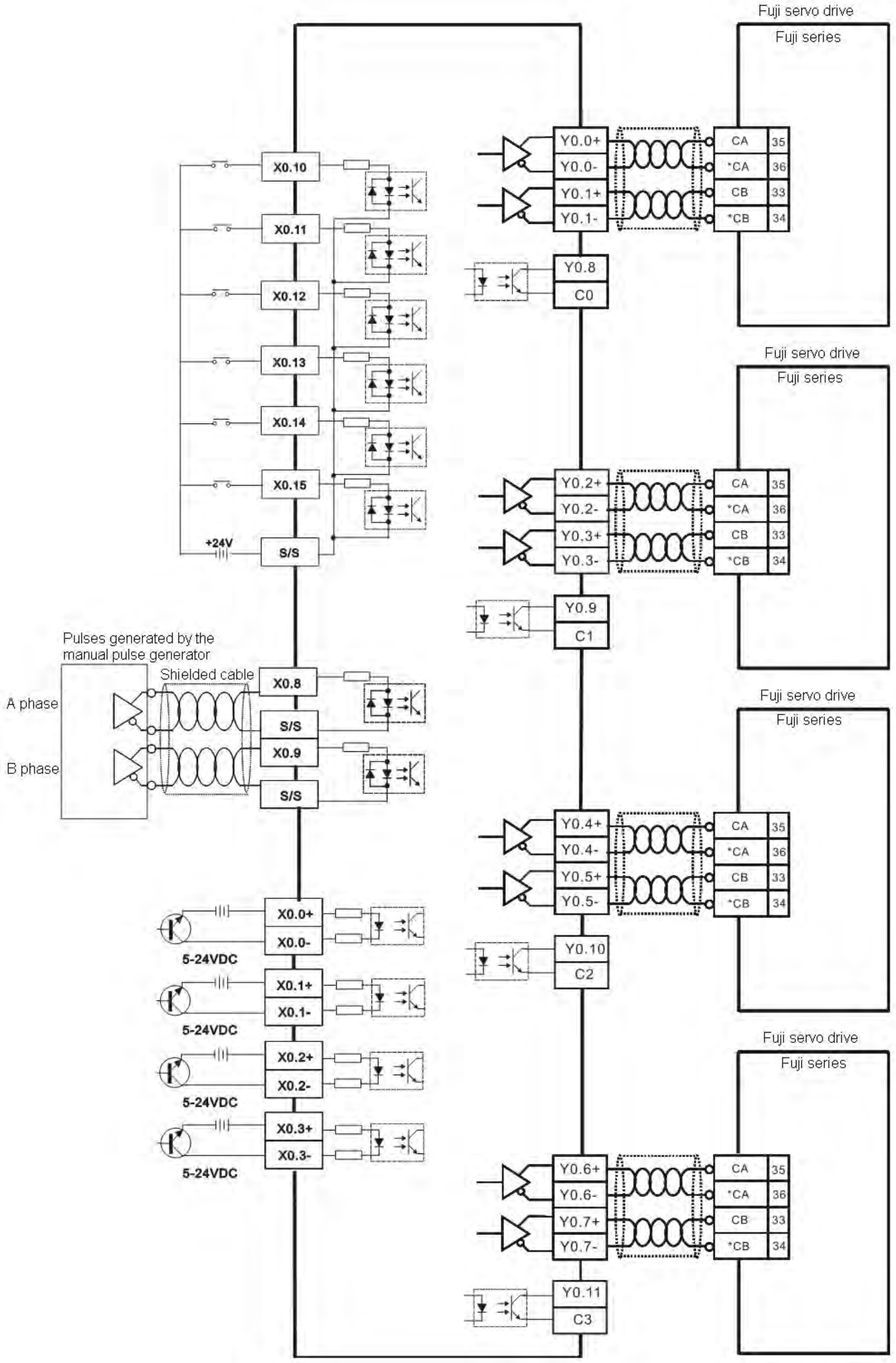
Wiring AH10PM-5A and an SGDV servo drive:



Wiring AH10PM-5A and a Mitsubishi MR-J2 series servo drive:



Wiring AH10PM-5A and a Fuji servo drive:



## 2.3 Communication Ports

AH05PM-5A is equipped with a mini USB port. AH20MC-5A/AH10PM-5A/AH15PM-5A is equipped with a mini USB port and an Ethernet port. Besides, AH20MC-5A is equipped with a DMCNET port. These ports are described below.

**Mini USB**: A mini USB port can function as a slave station. Users download or upload a program through a mini USB port. The communication protocols a mini USB port supports are Modbus ASCII and Modbus RTU.

**Ethernet**: An Ethernet port can function as a master station. The communication protocol it supports is MODBUS TCP/IP.

**DMCNET**: A DMCNET port can be used to control a servo drive.

Communication architecture:

| **Communication port** / **Communication parameter** | **Mini USB** |
|---|---|
| **Serial transmission rate** | 9,600~57,600 bps |
| **Number of data bits** | 7 bits~8 bits |
| **Parity bit** | Even parity bit/Odd parity bit/None |
| **Number of stop bits** | 1 data bit~2 data bits |
| **Register where a communication format is stored** | SR36 |
| **ASCII mode** | Slave stations are supported. |
| **RTU mode** | Slave stations are supported. |
| **Number of data read/written (ASCII mode)** | 100 registers |
| **Number of data read/written (RTU mode)** | 100 registers |
| **Communication port** / **Communication parameter** | **Ethernet** |
| **Transmission rate** | 10/100 Mbps |
| **Communication protocol** | Modbus TCP |
| **Number of data read/written** | 100 registers |
| **Communication port** / **Communication parameter** | **DMCNET** |
| **Serial transmission rate** | 10 Mbps (Channel A and channel B) |
| **Communication protocol** | DMCNET packet format |
| **Number of axes supported** | 12 axes |

Default communication protocol supported by a mini USB port

– Modbus ASCII mode

– 7 data bits

– 1 stop bit

– Even parity bit

– Serial transmission rate: 9600 bps

**Mini USB port**

1. The program in an AH500 series motion control module can be uploaded through the mini USB port on the AH500 series motion control module. Users can download a program to an AH500 series motion control module through the mini USB port on the AH500 series motion control module. The communication protocols that a mini USB port supports are Modbus ASCII and Modbus RTU, and the transmission rate supported is in the range of 9,600 bps to 57,600 bps.

**Ethernet port**

1. An Ethernet port is a communication port which has a RJ45 interface. It can function as a master station. The communication protocol that an Ethernet port supports is Modbus TCP/IP, and the transmission rate supported is 10/100 Mbps.
2. The communication protocol which an Ethernet port supports is Modbus TCP/IP. The program in an AH500 series motion control module can be uploaded through the Ethernet port on the AH500 series motion control module. Users can download a program to an AH500 series motion control module through the Ethernet port on the AH500 series motion control module. The status of AH500 series motion control module can be monitored through the Ethernet port on the AH500 series motion control module.

**DMCNET port (Only for AH20MC-5A)**

1. A DMCNET is a communication port which has a RJ45 interface. It can be connected to a Delta network servo drive. The communication protocol that a DMCNET port supports is DMCNET. The transmission rate of a channel is 10 Mbps. The two channels of an AH500 series motion control module can be simultaneously connected to a servo drive.
2. A DMCNET port can be used to control a Delta network servo drive.
3. Delta DMCNET cables: TAP-CB03/05/10/20/30/100

**MEMO**

# Chapter 3 Devices

3

## Table of Contents

# 3.1 Device List

■ Device list

| Type | Device name | | Number of devices | Range |
|---|---|---|---|---|
| Bit device | Input device | X | 256 | X0.0~X15.15 |
| | Output device | Y | 256 | Y0.0~Y15.15 |
| | Auxiliary relay | M | 4096 | M0~M4095 |
| | Special auxiliary relay | SM | 16384 | SM0~SM16383 |
| | Stepping relay | S | 1024 | S0~S1023 |
| Word device | Input device | X | 16 | X0~X15 |
| | Output device | Y | 16 | Y0~Y15 |
| | Data register | D | 10000 | D0~D9999 |
| | Special data register | SR | 16384 | SR0~SR16383 |
| | Timer | T | 256 | T0~T255 |
| | 16-bit counter | C | 200 | C0~C199 |
| | 32-bit counter | C | 16 | C240~C255 |
| | 32-bit high-speed counter | C | 6 | C200, C204, C208, C212, C216, and C220 (Please refer to section 3.8 for more information.) |
| | Index register | V | 6 | V0~V5 |
| | | Z | 8 | Z0~Z7 |
| Pointer | Pointer | P | 256 | P0~P255 |
| Constant | Decimal system | K | 16-bit operation: -32768~32767<br>32-bit operation: -2147483648~2147483647 | |
| | Hexadecimal system | 16# | 16-bit operation: 16#0~16#FFFF<br>32-bit operation: 16#0~16#FFFFFFFF | |
| | Floating-point number | F | 32-bit operation:<br>±1.17549435$^{-38}$~±3.40282347$^{+38}$ | |

# 3.2 Values, Constants, and Floating-point Numbers

| | | | |
|---|---|---|---|
| **Constant** | **K** | Decimal system | 16-bit operation: -32768~32767<br>32-bit operation: -2147483648~2147483647 |
| | **16#** | Hexadecimal system | 16-bit operation: 16#0~16#FFFF<br>32-bit operation: 16#0~16#FFFFFFFF |
| **Floating-point number** | **F** | 32-bit number | 32-bit operation:<br>±1.17549435-38~±3.40282347+38 |

An AH500 series motion control module performs operations on three types of values according to various control purposes. The functions of the three types of values are described below.

1. Binary number (BIN)

   The values on which an AH500 series motion control module performs operations, and the values stored in the AH500 series motion control module are binary numbers. Binary numbers are described below.

   Bit: A bit is the basic unit of information in the binary system. Its state is either 1 or 0.

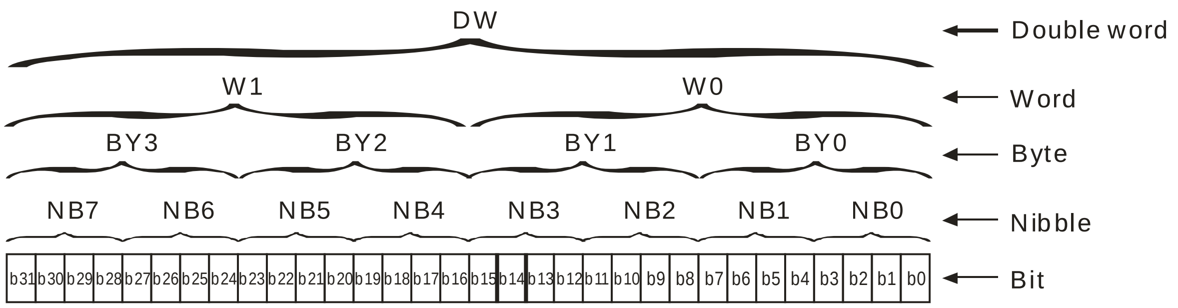
   Nibble: A nibble is composed of four consecutive bits (e.g. b3~b0). Nibbles can be used to represent 0~9 in the decimal system, or 0~F in the hexadecimal system.

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.




2. Decimal number (DEC)
   - A decimal number can be used as the setting value of a timer, or the setting value of a counter, e.g. TMR T0 K50 (K indicates that the value following it is a constant.).
   - A decimal number can be used as the number of an S/M/SM/T/C/D/SR/V/Z/P device, e.g. M10 and T30.
   - A decimal number can be used as an operand in an applied instruction, e.g. MOV K123 D0 (K indicates that the value following it is a constant.).
   - Decimal numbers can be used as external input numbers, e.g. X0.0~X0.15 and X1.0~X 1.15.
   - Decimal number can be used as external output numbers, e.g. Y0.0~Y0.15 and Y1.0~Y1.15.
3. Hexadecimal number (HEX)
   - A hexadecimal number can be used as an operand in an applied instruction, e.g. MOV H1A2B D0 (H indicates that the value following it is a constant.).

Constant (K): A decimal number is generally preceded by K. For example, K100 represents the decimal number 100.

| |
|---|
| Exception:<br>If K is used with an M/S device, a nibble device, a byte device, a word device, or a double word device will be formed.<br>Example:<br>K1M100 represents a device composed of 4 bits, K2M100 represents a device composed of 8 bits, K3M100 represents a device composed of 12 bit, and K4M100 represents a device composed of 16 bits. |

Constant (16#): A hexadecimal number is generally preceded by 16#. For example, the hexadecimal number 16#100 represents the decimal number 256.

Floating-point number (F): A floating-point number is generally preceded by F. For example, the floating-point number F3.123 represents 3.123.

## 3.3 External Input Devices and External Output Devices

- Input device (X):

  Input device numbers are decimal numbers. They start from X0.0. The definition of the number of an input device depends on the input device. An AH500 series motion control module has 256 input devices at most (X0.0~X15.15).

- Output device (Y):

  Output device numbers are decimal numbers. They start from Y0.0. The definition of the number of an output device depends on the output device. An AH500 series motion control module has 256 output devices at most (Y0.0~Y15.15).

- Functions of input devices:

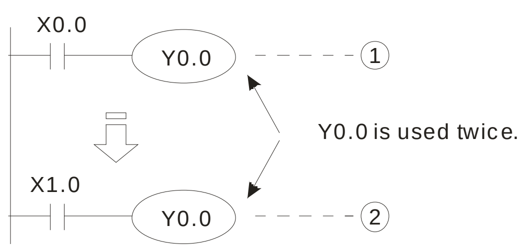
  After X devices in an AH500 series motion control module are connected to an input device, the input signals sent to the AH500 series motion control module will be read. There is no limitation on the number of times the Form A contact/the Form B contact of an X device can be used in a program. The state of an X device varies with the state of the input device to which the X device is connected.

- Users can turn X devices ON/OFF by means of SM304.

  If SM304 is OFF, X devices can not be turned ON/OFF by means of PMSoft. If SM304 is ON, X devices can be turned ON/OFF by means of PMSoft. However, if users use PMSoft to turn ON/OFF X devices in an AH500 series motion control module when SM304 is ON, the function of updating input signals will be disabled.

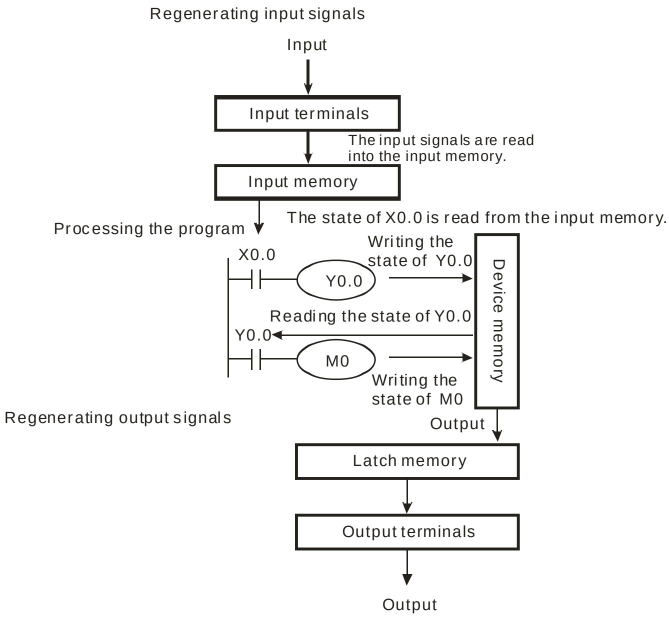
- Functions of output devices:

  A Y device sends a signal to drive the load connected to it. There is no limitation on the number of times the Form A contact/the Form B contact of a Y device can be used in a program. However, it is suggested that a Y device should be used once in a program. If a Y device is used more than once in a program, the state of the Y device depends on the Y device used last time.




The state of Y0.0 depends on circuit 2, that is, the state of X1.0 determines the state of Y0.0.

The procedure for processing a program is described below.




Regenerating an input signal:

1. Before an AH500 series motion control module executes a program, it reads the states of the input signals sent to it into its input memory.
2. If the states of the input signals change during the execution of the program, the states of input signals stored in the input memory will not change until the AH500 series motion control module reads the states of the input signals sent to it next time.
3. The time it takes for an input device in the program to receive the state of an external signal is about 10 milliseconds. (The time it takes for a contact in the program to receive the state of an external signal may be affected by the time it takes for the program to be scanned.)

Processing a program:

After the AH500 series motion control module reads the states of the input signals stored in the input memory, the execution of the instructions in the program will start from the beginning of the program. After the program is executed, the states of the Y devices used in the program will be stored in the device memory in the AH500 series motion control module.

Regenerating an output signal:

1. After M102 is executed, the states of the Y devices stored in the device memory will be sent to the latch memory in the AH500 series motion control module.
2. The time it takes for an output device to be turned form ON to OFF is about 10~20 milliseconds.

## 3.4 Auxiliary Relays

Auxiliary relay (M): Auxiliary relay numbers are decimal numbers. There are 4096 M devices (M0~M4096) in an AH500 series motion control module.

Functions of auxiliary relays: A M device has an output coil and a Form A contact/Form B contact. There is no limitation on the number of times an M device can be used in a program. Users can combine control loops by means of M devices, but can not drive external loads by means of M devices. If a power cut occurs when an AH500 series motion control module runs, the M devices in the AH500 series motion control module will be reset to OFF. When the supply of electricity is restored, the M devices are still OFF.

## 3.5 Special Auxiliary Relays

Special auxiliary relay (SM): Special auxiliary relay numbers are decimal numbers. There are 16384 SM devices (SM0~SM16383) in an AH500 series motion control module.

Functions of special auxiliary relays: A SM device has an output coil and a Form A contact/Form B contact. There is no limitation on the number of times a SM device can be used in a program. Users can combine control loops by means of SM devices, but can not drive external loads by means of SM devices. Every SM device has its own specific function. Please do not the SM devices which are not defined.

## 3.6 Stepping Relays

Stepping relay (S): Stepping relay numbers are decimal numbers. There are 1024 S devices (S0~S1023) in an AH500 series motion control module.

Functions of stepping relays: A S device has an output coil and a Form A contact/Form B contact. There is no limitation on the number of times an S device can be used in a program. Users can combine control loops by means of SM devices, but can not drive external loads by means of S devices. An S device can be used as a general auxiliary relay.

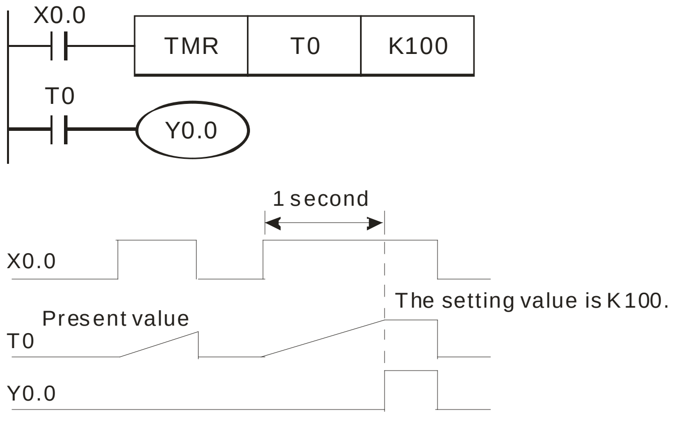
## 3.7 Timers

Timer (T): Timer numbers are decimal numbers. There are 256 timers (T0~T255) in an AH500 series motion control module. Users can change a timer to a latching timer by setting a parameter.

Functions of timers: 10 milliseconds are a unit of measurement for time. A timer counts upwards for measuring time which elapses. If the present timer value is equal to the setting value, the output coil will be ON. The setting value can be a decimal value preceded by K, or the value in a data register.

Actual time measured by a timer= Unit of measurement for time x Setting value

1. If the instruction TMR is executed, a timer will count for measuring time which elapses once. If the timer value matches the setting value, the output coil will be ON.




If X0.0 is ON, the timer T0 will count upwards from the present time value every 10 milliseconds. If the present timer value matches the setting value K100, the output coil T0 will be ON.

If X0.0 is OFF, or there is a power cut, the present value in T0 will become 0, and the output coil T0 will be OFF.

Setting value: Actual time measured by a timer= Unit of measurement for time x Setting value

1. Constant preceded by K: A setting value can be a constant preceded by K.
2. Value in a data register: A setting value can be the value in a data register.

## 3.8 Counters

Counter (C): Counter numbers are decimal numbers.

- **AH20MC-5A**

| Function | Range | | Remark |
|---|---|---|---|
| 16-bit up counter | C0~C199<br>200 counters | 216 counters in total | If the present value of the counter specified by the instruction CNT (DCNT) matches the setting value, the contact represented by the counter will be ON. |
| 32-bit up/down counter | C240~C255<br>16 counters<br>(Accumulation) | | |
| 32-bit high-speed counter | C200, C204, C208, C212, C216, and C220<br>6 counters | 6 counters in total | Input contact of C200: X0.8+/X0.8-/X0.9+/X0.9-<br>Input contact of C204: X0.10+/X0.10-/X0.11+/X0.11-<br>Input contact of C208: X0.12+/X0.12-/X0.13+/X0.13-<br>Input contact of C212: X0.14+/X0.14-/X0.15+/X0.15-<br>Input contact of C216: X0.12+/X0.12-/X0.13+/X0.13-<br>Input contact of C220: X0.14+/X0.14-/X0.15+/X0.15- |

- **AH10PM-5A**

| Function | Range | | Remark |
|---|---|---|---|
| 16-bit up counter | C0~C199<br>200 counters | 216 counters in total | If the present value of the counter specified by the instruction CNT (DCNT) matches the setting value, the contact represented by the counter will be ON. |
| 32-bit up/down counter | C240~C255<br>16 counters<br>(Accumulation) | | |
| 32-bit high-speed counter | C200, C204, C208, C212, C216, and C220<br>6 counters | 6 counters in total | Input contact of C200: X0.8/X0.9<br>Input contact of C204: X0.10/X0.11<br>Input contact of C208: X0.12/X0.13<br>Input contact of C212: X0.14/X0.15<br>Input contact of C216: X0.12/X0.13<br>Input contact of C220: X0.14/X0.15 |

- **AH15PM-5A**

| Function | Range | | Remark |
|---|---|---|---|
| 16-bit up counter | C0~C199<br>200 counters | 216 counters in total | If the present value of the counter specified by the instruction CNT (DCNT) matches the setting value, the contact represented by the counter will be ON. |
| 32-bit up/down counter | C240~C255<br>16 counters<br>(Accumulation) | | |
| 32-bit high-speed counter | C200, C204, C208, C212, C216, and C220<br>6 counters | 6 counters in total | Input contact of C200: X0.8/X0.9<br>Input contact of C204: X0.10/X0.11<br>Input contact of C208: X0.12/X0.13<br>Input contact of C212: X0.14/X0.15<br>Input contact of C216: X0.12/X0.13<br>Input contact of C220: X0.14/X0.15 |

- **AH05PM-5A**

| Function | Range | | Remark |
|---|---|---|---|
| 16-bit up counter | C0~C199<br>200 counters | 216 counters in total | If the present value of the counter specified by the instruction CNT (DCNT) matches the setting value, the contact represented by the counter will be ON |
| 32-bit up/down counter | C240~C255<br>16 counters (Accumulation) | | |
| 32-bit high-speed counter | C200 | 1 counter in total | Input contact of C200: X0.8/X0.9 |

Characteristics of counters:

| Item | 16-bit counter | 32-bit counter | |
|---|---|---|---|
| Type | General counter | General counter | High-speed counter |
| Direction | Counting up | Counting up/down | |
| Setting value | 0~32,767 | -2,147,483,648~+2,147,483,647 | |
| Specification of a setting value | Constant preceded by K, or value stored in a data register | Constant preceded by K, or value stored in two consecutive data registers | |
| Change of the present value | If the present value matches the setting value, the counter will stop counting. | Even if the present value matches the setting value, the counter will keep counting. | |
| Output contact | If the present value matches the setting value, the output contact will be ON. | Counting up: If the present value matches the setting value, the output contact will be ON.<br>Counting down: If the present value matches the setting value, the output contact will be reset to OFF. | |
| Resetting of a contact | If the instruction RST is executed, the present value will becomes zero, and the contact will be reset to OFF. | | |
| Actions of contacts | After the scan of a program is complete, the contacts will act. | After the scan of a program is complete, the contacts will act. | If the present value matches the setting value, the contact will be ON. |

Functions of counters:

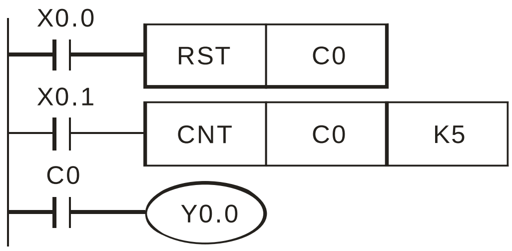
If the input signal of a counter is turned from OFF to ON, and the present value of the counter matches the setting value, the output coil will be ON. A setting value can be a constant preceded by K, or the value stored in a data register.

16-bit counter:

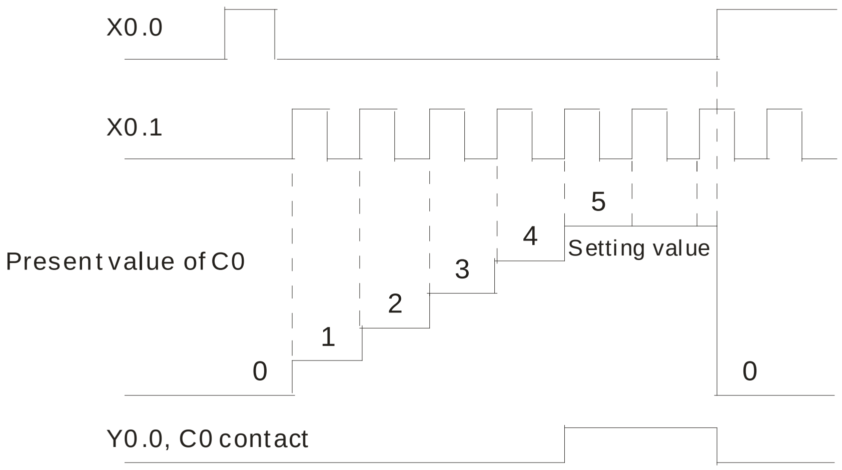
1. The setting value of a 16-bit counter must be in the range of K0 to K32,767. (K0 is equal to K1. If the setting value of a counter is K0 or K1, the output contact will be ON after the counter counts for the first time.)
2. If a value greater than the setting value of C0 is moved to C0 by means of the instruction MOV, the contact C0 will be ON, and the present value of the counter will become the setting value next time X1 is turned from OFF to ON.
3. The setting value of a counter can be a constant preceded by K, or the value stored in a data register.
4. If the setting value of a counter is a value preceded by K, the setting value can only be a positive value. If the setting value of a counter is the value stored in a data register, the setting value can be a positive value or a negative value. If a counter counts up from the present value 32,767, the next value following 32,767 will be -32,768.

Example:

```
LD      X0.0
RST     C0
LD      X0.1
CNT     C0    K5
LD      C0
OUT     Y0.0
```




1. If X0.0 is ON, the instruction RST will be executed, the present value of C0 will become zero, and the output contact will be reset to OFF.
2. If X0.1 is turned from OFF to ON, the present value of the counter will increase by one.
3. If the present value of C0 matches the setting value K5, the contact C0 will be ON (Present value of C0=Setting value=K5). K5 will be retained even if X0.1 is turned from OFF to ON again.

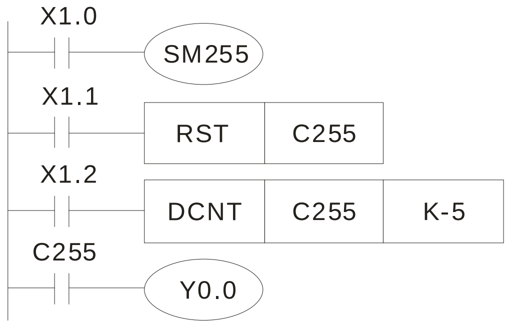



32-bit up/down counter:

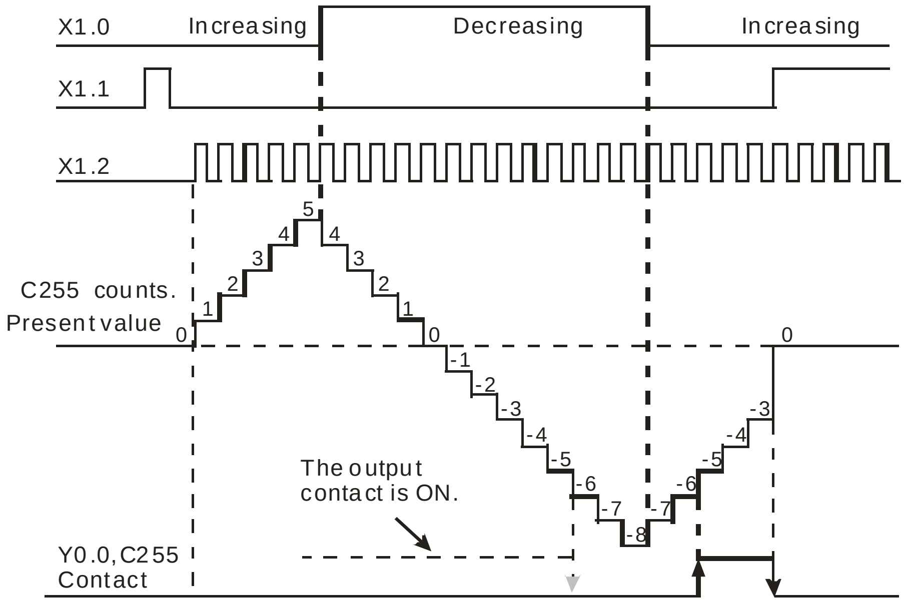
1. The setting value of a 32-bit general up/down counter must be in the range of K-2,147,483,648 to K2,147,483,647. The states of the special auxiliary relays SM240~SM255 determine whether the 32-bit general up/down counters C240~C255 count up or count down. For example, C240 will count up if SM240 is OFF, and C240 will count down if SM240 is ON.
2. The setting value of a 32-bit up/down counter can be a constant preceded by K, or the value stored in two consecutive data registers. A setting value can be a positive value, or a negative value.
3. If a power cut occurs when a general counter counts, the present value of the counter will be cleared. If a power cut occurs when a latching counter counts, the counter value and the state of the contact which are before the power cut will be retained, and the latching counter will not continue to count until power is restored.
4. If a counter counts up from the present value 2,147,483,647, the next value following 2,147,483,647 will be -2,147,483,648. If a counter counts down from the present value -2,147,483,648, the next value following -2,147,483,648 will be 2,147,483,647.

Example:

| | | |
|---|---|---|
| LD | X1.0 | |
| OUT | SM255 | |
| LD | X1.1 | |
| RST | C255 | |
| LD | X1.2 | |
| DCNT | C255 | K-5 |
| LD | C255 | |
| OUT | Y0.0 | |




1. SM255 is driven by X1.0. The state of SM255 determines whether C255 counts up or counts down.
2. If X1.1 is turned form OFF to ON, the instruction RST will be executed, the present value of C255 will become 0, and the contact will be OFF.
3. If X1.2 is turned form OFF to ON, the present value of the counter will increase by one or decrease by one.
4. If the present value of the counter C255 increases from K-6 to K-5, the contact C255 will be turned form OFF to ON. If the present value of the counter C255 decreases from K-5 to K-6, the contact C255 will be turned from ON to OFF.
5. If a value greater than the setting value of C255 is moved to C255 by means of the instruction MOV, the contact C255 will be ON, and the present value of the counter will become the setting value next time X1.1 is turned from OFF to ON.



32-bit high-speed counter:

1. The setting value of s 32-bit high-speed counter must be in the range of K-2,147,483,648 to K2,147,483,647.
2. Mode of counting:

| Counter | Mode of counting | | Resetting of a counter | External reset terminal | External input terminal*1 *2 |
|---|---|---|---|---|---|
| | Device | Setting value*3 | | | |
| C200 | K1SM200 | 0: U/D<br>1: P/D<br>2: A/B (One time the frequency of A/B-phase inputs)<br>3: 4A/B (Four times the frequency of A/B-phase inputs) | SM203 | X0.0+ and X0.0- | X0.8, X0.9, and S/S |
| C204 | K1SM204 | | SM207 | X0.1+ and X0.1- | X0.10, X0.11, and S/S |
| C208 | K1SM208 | | SM211 | X0.2+ and X0.2- | X0.12，X0.13, and S/S |
| C212 | K1SM212 | | SM215 | X0.3+ and X0.3- | X0.14, X0.15, and S/S |
| C216 | K1SM216 | | SM219 | X0.2+ and X0.2- | X0.12，X0.13, and S/S |
| C220 | K1SM220 | | SM223 | X0.3+ and X0.3- | X0.14, X0.15, and S/S |

*1. The input terminals of AH20MC-5A are differential input terminals. X0.8 and X0.9 on AH15PM-5A are differential input terminals. The input terminals of AH05PM-5A/AH10PM-5A are transistors whose collectors are open collectors. X0.10~X0.15 on AH15PM-5A are transistors whose collectors are open collectors.

*2. The terminal S/S on AH05PM-5A/10PM-5A must be connected. X0.10~X0.15 on AH15PM-5A must be connected to the terminal S/S.

*3. U/D: Counting up/Counting down; P/D: Pulse/Direction; A/B: A phase/B phase

- C200: Users can select a mode of counting by setting SM200 and SM201. Input signals are controlled by X0.8 and X0.9. If SM203 is ON, the function of resetting C200 will be enabled. Resetting signals are controlled by X0.0.
- C204: Users can select a mode of counting by setting SM204 and SM205. Input signals are controlled by X0.10 and X0.11. If SM207 is ON, the function of resetting C204 will be enabled. Resetting signals are controlled by X0.1.
- C208: Users can select a mode of counting by setting SM208 and SM209. Input signals are controlled by X0.12 and X0.13. If SM211 is ON, the function of resetting C208 will be enabled. Resetting signals are controlled by X0.2.
- C212: Users can select a mode of counting by setting SM212 and SM213. Input signals are controlled by X0.14 and X0.15. If SM215 is ON, the function of resetting C212 will be enabled. Resetting signals are controlled by X0.3.
- C216: Users can select a mode of counting by setting SM216 and SM217. Input signals are controlled by X0.12 and X0.13. If SM219 is ON, the function of resetting C216 will be enabled. Resetting signals are controlled by X0.2.
- C220: Users can select a mode of counting by setting SM220 and SM221. Input signals are controlled by X0.14 and X0.15. If SM223 is ON, the function of resetting C220 will be enabled. Resetting signals are controlled by X0.3.

3. The setting value of a 32-bit high-speed counter can be a constant preceded by K, or the value stored in two consecutive data registers. A setting value can be a positive value, or a negative value.
4. If a power cut occurs when a general counter counts, the present value of the counter will be cleared. If a power cut occurs when a latching counter counts, the counter value and the state of the contact which are before the power cut will be retained, and the latching counter will not continue to count until power is restored.
5. If a counter counts up from the present value 2,147,483,647, the next value following 2,147,483,647 will be -2,147,483,648. If a counter counts down from the present value -2,147,483,648, the next value following -2,147,483,648 will be 2,147,483,647.

## 3.9 Data Registers and Index Registers

| | | |
|---|---|---|
| **General register (D)** | D0~D9999 (10,000 data registers) | 10,014 registers in total |
| **Index register (V)/(Z)** | V0~V5 and Z0~Z7 (14 index registers) | |

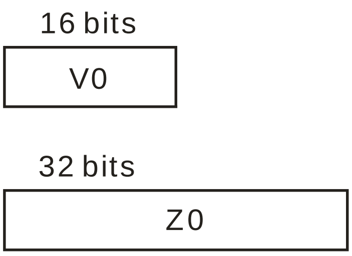
Registers are classified according to character. There are three types of registers.

1. General register: If the STOP/RUN switch on a module is turned from the STOP position to the RUN position, or a module is disconnected, the values in the general registers will become 0. If SM033 in a module is turned ON, the values in the general registers will be retained after the STOP/RUN switch on the module is turned from the RUN position to the STOP position, and will become 0 after the module is disconnected.
2. Latching register: If a module is disconnected, the values in the latching registers will be retained.
   If users want to clear the value in a latching register, they can use the instruction RST or ZRST.
   There are no latching registers in AH500 series motion control modules.
3. Index register (V)/(Z): V devices are 16-bit registers, and Z devices are 32-bit registers. There are 6 V devices (V0~V5), and 8 Z devices (Z0~Z7) in an AH500 series motion control module.

### 3.9.1 Data Registers

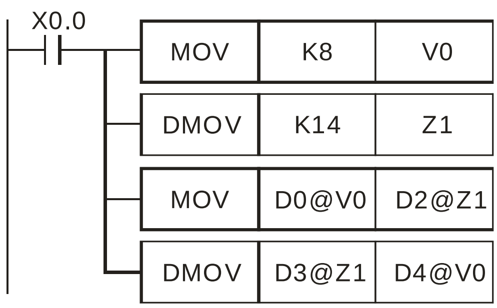
The value in a data register is a 16-bit value. The highest bit in a 16-bit data register represents an algebraic sign. The value stored in a data register must be in the range of -32,768 to +32,767. Two 16-bit data registers can be combined into one 32-bit data register (D+1, D). The highest bit in a 32-bit data register represents an algebraic sign. The value stored in a 32-bit data register must be in the range of -2,147,483,648 to +2,147,483,647.

### 3.9.2 Index Registers




V devices are 16-bit registers. Data can be freely written into a V device, and data can be freely read from a V device. If a V device is used as a general register, it can only be used in a 16-bit instruction.

Z devices are 32-bit registers. If a Z device is used as a general register, it can only be used in a 32-bit instruction.




If X0.0 is ON, the value in V0 will be 8, and the value in Z1 will be 14, the value in D8 will be moved to D16, and the value in D17 will be moved to D12.

If a V device or a Z device is an index register used to modify an operand, the V device or the Z device can be used in a 16-bit instruction and a 32-bit instruction.

Index registers are like general operands in that they can be used in movement instructions and comparison instructions. They can be used to modify word devices (KnM/KnS/T/C/D/SR devices) and bit devices (X/Y/M/S/SM devices).

There are 6 V devices (V0~V5), and 8 Z devices (Z0~Z7) in an AH500 series motion control module.

※Constants and some instructions do not support the use of index registers. Please refer to section 5.4 for more information about using index registers to modify operands.

## 3.10 Special Data Registers

The value in a special data register is a 16-bit value. The highest bit in a 16-bit special data register represents an algebraic sign. The value stored in a special data register must be in the range of -32,768 to +32,767. Two 16-bit special data registers can be combined into one 32-bit special data register (SR+1, SR). The highest bit in a 32-bit special data register represents an algebraic sign. The value stored in a 32-bit special data register must be in the range of -2,147,483,648 to +2,147,483,647.

| | |
|---|---|
| **Special data register (SR)** | SR0~SR16383 (16,384 special data registers) |

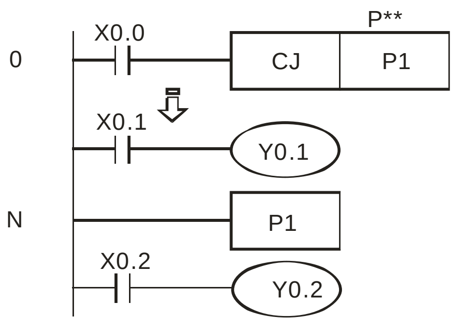
Special data register: Every special data register has its definition and purposes. System states, error messages, and states monitored are stored in special data registers. Please refer to section 3.12 and section 3.13 for more information about special auxiliary relays and special data registers.

## 3.11 Pointers

| | |
|---|---|
| **Pointer (P)** | P0~P255 (256 pointers) |

Pointer: A pointer is used with API 00 CJ, API 256 CJN, or API 257 JMP. Please refer to Chapter 5 for more information about the use of CJ/CJN/JMP.

Conditional jump (CJ):




If X0.0 is ON, the execution of the program will jump from address 0 to address N, and the part of the program between address 0 and address N will not be executed.

If X0.0 is OFF, the execution of the program starts from address 0, and the instruction CJ will not be executed.

# 3.12 Special Auxiliary Relays and Special Data Registers

## 3.12.1 Special Auxiliary Relays

Special auxiliary relays (SM devices) and special data registers (SR devices) are shown in the tables below. Some device numbers in the tables are marked with *. Users can refer to section 3.13 for more information. If the attribute of a device is “R”, the users can only read data from the device. If the attribute of a device is “R/W”, the users can read data from the device, and write data into the device. In addition, “-” indicates that the state of a special auxiliary relay is unchanged, or the value in a special data register is unchanged. “#” indicates that a special auxiliary relay or a special data register in an AH500 series motion controller is set according to the state of the AH500 series motion controller. The users can read a setting value, and refer to the manual for more information.

| SM number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Default |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | |
| 000* | If the module runs, SM000 will be a normally-open contact (Form A contact). When the module runs, SM000 is ON. | ○ | ○ | ○ | ○ | OFF | ON | OFF | R | OFF |
| 001* | If the module runs, SM001 will be a normally-closed contact (Form B contact). When the module runs, SM001 is OFF. | ○ | ○ | ○ | ○ | ON | OFF | ON | R | ON |
| 002* | A positive-going pulse is generated at the time when the module runs. The width of the pulse is equal to the scan cycle. | ○ | ○ | ○ | ○ | OFF | ON | OFF | R | OFF |
| 003* | A negative-going pulse is generated at the time when the module runs. The width of the pulse is equal to the scan cycle. | ○ | ○ | ○ | ○ | ON | OFF | ON | R | ON |
| 008 | The watchdog timer is ON. | ○ | ○ | ○ | ○ | OFF | OFF | - | R | OFF |
| 009 | The low voltage signal has ever occurred. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 011 | 10 millisecond clock pulse (The pulse is ON for 5 milliseconds, and is OFF for 5 milliseconds.) | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 012 | 100 millisecond clock pulse (The pulse is ON for 50 milliseconds, and OFF for 50 milliseconds.) | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 013 | 1 second clock pulse (The pulse is ON for 0.5 seconds, and OFF for 0.5 seconds.) | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 014 | 1 minute clock pulse (The pulse is ON for 30 seconds, and OFF for 30 seconds.) | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |

| SM number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Default |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | |
| 020 | Zero flag (for the instructions SFRD and SFWR) | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 022 | Carry flag (for the instructions SFWR, RCR, and RCL) | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 034 | All the outputs are disabled. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 039* | The scan time for the program is fixed. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 072 | The module is made to run. (Communication) | ○ | ○ | ○ | ○ | OFF | ON | OFF | R/W | OFF |
| 073 | The syntax is checked. | ○ | ○ | ○ | ○ | OFF | ON | OFF | R/W | OFF |
| 087 | The low voltage signal occurs. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 161 | 8-bit mode (ON: 8-bit mode; OFF: 16-bit mode) | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 200* | C200: Selecting a mode of counting | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 201 | | | | | | OFF | - | - | R/W | OFF |
| 203 | Resetting C200 | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 204* | C204: Selecting a mode of counting | X | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 205 | | | | | | OFF | - | - | R/W | OFF |
| 207 | Resetting C204 | X | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 208* | C208: Selecting a mode of counting | X | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 209 | | | | | | OFF | - | - | R/W | OFF |
| 211 | Resetting C208 | X | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 212* | C212: Selecting a mode of counting | X | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 213 | | | | | | OFF | - | - | R/W | OFF |
| 215 | Resetting C212 | X | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 216* | C216: Selecting a mode of counting | X | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 217 | | | | | | OFF | - | - | R/W | OFF |
| 219 | Resetting C216 | X | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 220* | C220: Selecting a mode of counting | X | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 221 | | | | | | OFF | - | - | R/W | OFF |
| 223 | Resetting C220 | X | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 240 | ON: C240 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 241 | ON: C241 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 242 | ON: C242 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 243 | ON: C243 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 244 | ON: C244 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 245 | ON: C245 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 246 | ON: C246 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 247 | ON: C247 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 248 | ON: C248 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 249 | ON: C249 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 250 | ON: C250 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 251 | ON: C251 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 252 | ON: C252 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 253 | ON: C253 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |

| SM number | Function | Applicable model*1 | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Default |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | |
| 254 | ON: C254 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 255 | ON: C255 counts down. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 303 | The high bits in the device specified in the instruction XCH are interchanged with the low bits in the device specified in the instruction XCH. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 304 | The X devices can be forced to be ON/OFF by means of PMSoft. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 920 | Using a radian or a degree in O100 | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 952 | O100 is ready. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 953* | An error occurs in O100. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 968 | Zero flag in O100 | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 969 | Borrow flag in O100 | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 970 | Carry flag in O100 | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 971 | An error occurs in a floating-point operation in O100. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |

*1: 05M=AH05PM-5A; 15M=AH15PM-5A; 10M=AH10PM-5A; 20M=AH20MC-5A

Flags related to an Ox motion subroutine

| SM number | Function | Applicable model*1 | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Default |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | |
| 1016 | Using a radian or a degree in the Ox motion subroutine | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 1049 | Ox motion subroutine error flag (It is reset at the time when the module runs.) | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| 1050* | If an M code in an Ox motion subroutine is executed, SM1050 will be ON. (SM1050 is reset to OFF at the time when the Ox motion subroutine is executed.) | ○ | ○ | ○ | ○ | OFF | - | OFF | R | OFF |
| 1051 | If M00 is executed, SM1051 will be ON. (SM1051 is reset to OFF at the time when the Ox motion subroutine is executed.) | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 1052 | If M02 is executed, SM1052 will be ON. (SM1052 is reset to OFF at the time when the Ox motion subroutine is executed.) | ○ | ○ | ○ | ○ | OFF | ON | - | R | OFF |
| 1064 | Zero flag in the Ox motion subroutine | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |

| SM number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Default |
|---|---|---|---|---|---|---|---|---|---|---|
| 1065 | Borrow flag in the Ox motion subroutine | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 1066 | Carry flag in the Ox motion subroutine | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |
| 1067 | An error occurs in a floating-point operation in the Ox motion subroutine. | ○ | ○ | ○ | ○ | OFF | - | - | R | OFF |

*1: 05M=AH05PM-5A; 15M=AH15PM-5A; 10M=AH10PM-5A; 20M=AH20MC-5A

Special auxiliary relays for motion axis 1~motion axis 16:
SM1000~SM2599 are for motion axis 1~motion axis 16. Every axis uses 100 special auxiliary relays.

- Motion axis~motion axis 16 have the same number of special auxiliary relays. The sixteen groups of special auxiliary relays have the same definitions.
- The special auxiliary relays for motion axis 1~motion axis 16 starts from SM1000. Every axis has 100 special auxiliary relays.

SM mn xx

→ It indicates the definition of a flag for an axis. If **xx** is 48, it is the ready flag for an axis.

→ It represents an axis number. If **mn** is 10, it represents motion axis 1. If **mn** is 25, it represents motion axis 16.

Example: SM1048 is the ready flag for the first axis, SM1148 is the ready axis for the second axis, and SM1548 is the ready flag for the sixth axis.

- The definitions of the special auxiliary relays for motion axis 1~motion axis 16 are shown below.

| Axis number | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Special auxiliary relay | SM1000~ SM1099 (mn=10) | SM1100~ SM1199 (mn=11) | SM1200~ SM1299 (mn=12) | SM1300~ SM1399 (mn=13) | SM1400~ SM1499 (mn=14) | SM1500~ SM1599 (mn=15) |
| Axis number | 7 | 8 | 9 | 10 | 11 | 12 |
| Special auxiliary relay | SM1600~ SM1699 (mn=16) | SM1700~ SM1799 (mn=17) | SM1800~ SM1899 (mn=18) | SM1900~ SM1999 (mn=19) | SM2000~ SM2099 (mn=20) | SM2100~ SM2199 (mn=21) |
| Axis number | 13 | 14 | 15 | 16 | The special auxiliary relays starting from SM2600 are not used. | |
| Special auxiliary relay | SM2200~ SM2299 (mn=22) | SM2300~ SM2399 (mn=23) | SM2400~ SM2499 (mn=24) | SM2500~ SM2599 (mn=25) | | |

| SM number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Default |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | |
| mn04 | The cyclic motion of the electronic cam stops. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |

| SM number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Default |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | |
| mn08 | Setting an offset for the master axis of the electronic cam | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| mn17* | The axis specified stops at the angle specified. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| mn48* | The axis specified is ready. | ○ | ○ | ○ | ○ | ON | ON | ON | R | ON |
| mn49* | Motion error flag (It is reset at the time when the module runs.) | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |
| mn69 | Beginning of the electronic cam cycle | ○ | ○ | ○ | ○ | OFF | - | - | R/W | OFF |

*1: 05M=AH05PM-5A; 15M=AH15PM-5A; 10M=AH10PM-5A; 20M=AH20MC-5A

### 3.12.2 Special Data Registers

| SR number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Latching | Default |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | | |
| 000* | Watchdog timer (Unit: ms) | ○ | ○ | ○ | ○ | 200 | - | - | R/W | NO | 200 |
| 002 | Size of the program | ○ | ○ | ○ | ○ | 65535 | - | - | R | NO | 65535 |
| 003 | Checksum of the program | ○ | ○ | ○ | ○ | - | - | - | R | NO | 0 |
| 004 | Checksum of the program and the cam charts | ○ | ○ | ○ | ○ | - | - | - | R | NO | 0 |
| 005 | Firmware version of the AH500 series motion control module (factory setting) | ○ | ○ | ○ | ○ | # | - | - | R | NO | # |
| 008 | Step address at which the watchdog timer is ON | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 009 | Number of times the low voltage signal occurs | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 010 | Present scan time (Unit: 1 millisecond) | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 011 | Minimum scan time (Unit: 1 millisecond) | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 012 | Maximum scan time (Unit: 1 millisecond) | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 020* | Filtering the inputs (Unit: ms) | ○ | ○ | ○ | ○ | 10 | - | - | R/W | NO | 10 |
| 039* | Fixed scan time (Unit: ms) | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |

| SR number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Latching | Default |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | | |
| 121 | Communication address of the AH500 series motion control module | ○ | ○ | ○ | ○ | - | - | - | R/W | YES | 1 |
| 200 | Special auxiliary relay from which the special auxiliary relays backed up onto the SD card start | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 201 | Special auxiliary relay at which the special auxiliary relays backed up onto the SD card end | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 202 | Timer from which the timers backed up onto the SD card start | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 203 | Timer at which the timers backed up onto the SD card end | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 204 | 16-bit counter from which the 16-bit counters backed up onto the SD card start | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 205 | 16-bit counter at which the 16-bit counters backed up onto the SD card end | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 206 | 32-bit counter from which the 32-bit counters backed up onto the SD card start | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 207 | 32-bit counter at which the 32-bit counters backed up onto the SD card end | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 208 | Stepping relay from which the stepping relays backed up onto the SD card start | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |

| SR number | Function | Applicable model*1 | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Latching | Default |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | | |
| 209 | Stepping relay at which the stepping relays backed up onto the SD card end | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 210 | Data register from which the data registers backed up onto the SD card start | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 211 | Data register at which the data registers backed up onto the SD card end | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 212 | W register from which the W registers backed up onto the SD card start | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 213 | W register at which the W registers backed up onto the SD card end | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 214 | Accessing the SD card | X | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 400* | Enabling the interrupt | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 401* | Cycle of the time interrupt (Unit: ms) | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 700 | Ox motion subroutine which is executed | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 702 | Step address which is executed in the Ox motion subroutine | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 703* | M-code which is executed in the Ox motion subroutine | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 704 | Dwell duration which is set | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 705 | Present dwell duration | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |

| SR number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Latching | Default |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | | |
| 706 | Number of times the instruction RPT in the Ox motion subroutine is executed | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 707 | Number of times the instruction RPT in the Ox motion subroutine has been executed | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 708<br>709 | Compensation value for the x-axis/y-axis | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 710<br>711 | Compensation value for the center | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 712<br>713 | Compensation value for the radius | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 796*<br>797* | Speed to which the speed of the continuous interpolation decreases | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 798* | Percentage for the values of the speed parameters of the G-codes | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 799* | Polarities of the input terminals | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 800* | States of the input terminals | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 802* | O100 error code | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 803* | Step address in O100 at which an error occurs | ○ | ○ | ○ | ○ | 0 | 0 | - | R/W | NO | 0 |
| 804 | Polarities of the input terminals | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 805 | States of the input terminals | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| 806* | Filter coefficient for the input terminals | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| 808 | Ethernet IP address | X | ○ | ○ | ○ | - | - | - | R/W | NO | 100 |
| 809 | Ethernet IP address | X | ○ | ○ | ○ | - | - | - | R/W | NO | 49320 |

*1: 05M=AH05PM-5A; 15M=AH15PM-5A; 10M=AH10PM-5A; 20M=AH20MC-5A

Special data registers related to an Ox motion subroutine

| SR number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Default |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | |
| 1041 | Ox motion subroutine error code | ○ | ○ | ○ | ○ | 0 | - | - | R | 0 |
| 1049 | Mode of stopping Ox0~Ox99 (K1: The execution of Ox0~Ox99 will resume next time Ox0~Ox99 are started. K2: The next instruction will be executed next time Ox0~Ox99 are started. Others: Ox0~Ox99 are executed again. | ○ | ○ | ○ | ○ | OFF | - | - | R/W | 0 |
| 1052* | Setting an Ox motion subroutine number | ○ | ○ | ○ | ○ | OFF | - | OFF | R/W | 0 |
| 1053 | Step address in the Ox motion subroutine at which an error occurs | ○ | ○ | ○ | ○ | OFF | - | - | R | 0 |

*1: 05M=AH05PM-5A; 15M=AH15PM-5A; 10M=AH10PM-5A; 20M=AH20MC-5A

Special data registers for motion axis 1~motion axis 16:
SR1000~SR2599 are for motion axis 1~motion axis 16. Every axis uses 100 special data registers.

- Motion axis~motion axis 16 have the same number of special data registers. The sixteen groups of special data registers have the same definitions.
- The special data registers for motion axis 1~motion axis 16 starts from SR1000. Every axis has 100 special data registers.

SR **mn** **xx**

**xx** → It indicates the definition of a special data register for an axis. If **xx** is 00, it indicates a special data register which users can use to set the parameters of an axis.

**mn** → It represents an axis number. If **mn** is 10, it represents motion axis 1. If **mn** is 25, it represents motion axis 16.

Example: The value in SR1000 indicates the setting of the parameters of the first axis, the value in SR1100 indicates the setting of the parameters of the second axis, and the value in SR1500 indicates the setting of the parameters of the sixth axis.

- The definitions of the special data registers for motion axis 1~motion axis 16 are shown below.

<table>
<tr><th>Axis number</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th></tr>
<tr><td>Special data register</td><td>SR1000~<br>SR1099<br>(mn=10)</td><td>SR1100~<br>SR1199<br>(mn=11)</td><td>SR1200~<br>SR1299<br>(mn=12)</td><td>SR1300~<br>SR1399<br>(mn=13)</td><td>SR1400~<br>SR1499<br>(mn=14)</td><td>SR1500~<br>SR1599<br>(mn=15)</td></tr>
<tr><td>Axis number</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td></tr>
<tr><td>Special data register</td><td>SR1600~<br>SR1699<br>(mn=16)</td><td>SR1700~<br>SR1799<br>(mn=17)</td><td>SR1800~<br>SR1899<br>(mn=18)</td><td>SR1900~<br>SR1999<br>(mn=19)</td><td>SR2000~<br>SR2099<br>(mn=20)</td><td>SR2100~<br>SR2199<br>(mn=21)</td></tr>
<tr><td>Axis number</td><td>13</td><td>14</td><td>15</td><td>16</td><td rowspan="2" colspan="2">The special data registers starting from SR2600 are not used.</td></tr>
<tr><td>Special data register</td><td>SR2200~<br>SR2299<br>(mn=22)</td><td>SR2300~<br>SR2399<br>(mn=23)</td><td>SR2400~<br>SR2499<br>(mn=24)</td><td>SR2500~<br>SR2599<br>(mn=25)</td></tr>
</table>

<table>
<tr><th rowspan="2">SR number</th><th rowspan="2">Function</th><th colspan="4">Applicable model[*1]</th><th rowspan="2">OFF<br>⇩<br>ON</th><th rowspan="2">STOP<br>⇩<br>RUN</th><th rowspan="2">RUN<br>⇩<br>STOP</th><th rowspan="2">Attribute</th><th rowspan="2">Latching</th><th rowspan="2">Default</th></tr>
<tr><th>05M</th><th>15M</th><th>10M</th><th>20M</th></tr>
<tr><td>mn00*</td><td>Setting the parameters of the axis specified</td><td>○</td><td>○</td><td>○</td><td>○</td><td>-</td><td>-</td><td>-</td><td>R/W</td><td>NO</td><td>0</td></tr>
<tr><td>mn01</td><td>Compensation value for the axis specified</td><td>○</td><td>○</td><td>○</td><td>○</td><td>-</td><td>-</td><td>-</td><td>R/W</td><td>NO</td><td>0</td></tr>
<tr><td>mn02</td><td>Number of pulses it takes for the motor of the axis specified to rotate once (A) (Low word)</td><td rowspan="2">○</td><td rowspan="2">○</td><td rowspan="2">○</td><td rowspan="2">○</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">R/W</td><td rowspan="2">NO</td><td rowspan="2">2000</td></tr>
<tr><td>mn03</td><td>Number of pulses it takes for the motor of the axis specified to rotate once (A) (High word)</td></tr>
<tr><td>mn04</td><td>Distance generated after the motor of the axis specified rotate once (B) (Low word)</td><td rowspan="2">○</td><td rowspan="2">○</td><td rowspan="2">○</td><td rowspan="2">○</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">R/W</td><td rowspan="2">NO</td><td rowspan="2">1000</td></tr>
<tr><td>mn05</td><td>Distance generated after the motor of the axis specified rotate once (B) (High word)</td></tr>
</table>

| SR number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Latching | Default |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | | |
| mn06 | Maximum speed ($V_{MAX}$) at which the axis specified rotates (Low word) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 500K |
| mn07 | Maximum speed ($V_{MAX}$) at which the axis specified rotates (High word) | | | | | | | | | | |
| mn08 | Start-up speed ($V_{BIAS}$) at which the axis specified rotates (Low word) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 0 |
| mn09 | Start-up speed ($V_{BIAS}$) at which the axis specified rotates (High word) | | | | | | | | | | |
| mn10 | JOG speed ($V_{JOG}$) at which the axis specified rotates (Low word) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 5000 |
| mn11 | JOG speed ($V_{JOG}$) at which the axis specified rotates (High word) | | | | | | | | | | |
| mn12 | Speed ($V_{RT}$) at which the axis specified returns home (Low word) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 50K |
| mn13 | Speed ($V_{RT}$) at which the axis specified returns home (high word) | | | | | | | | | | |
| mn14 | Speed ($V_{CR}$) to which the speed of the axis specified decreases when the axis returns home (Low word) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 1000 |
| mn15 | Speed ($V_{CR}$) to which the speed of the axis specified decreases when the axis returns home (High word) | | | | | | | | | | |
| mn16 | Number of PG0 pulses for the axis specified | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 0 |
| mn17 | Supplementary pulses for the axis specified | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 0 |
| mn18 | Home position of the axis specified (Low word) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 0 |

| SR number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Latching | Default |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | | |
| mn19 | Home position of the axis specified (High word) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 0 |
| mn20 | Time ($T_{ACC}$) it takes for the axis specified to accelerate | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 500 |
| mn21 | Time ($T_{DEC}$) it takes for the axis specified to decelerate | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 500 |
| mn22 | Target position of the axis specified (P (I)) (Low word) | ○ | ○ | ○ | ○ | ○ | - | - | R/W | NO | 0 |
| mn23 | Target position of the axis specified (P (I)) (High word) | ○ | ○ | ○ | ○ | ○ | - | - | R/W | NO | 0 |
| mn24 | Speed at which the axis specified rotates (V (I)) (Low word) | ○ | ○ | ○ | ○ | 1000 | - | - | R/W | NO | 1000 |
| mn25 | Speed at which the axis specified rotates (V (I)) (High word) | ○ | ○ | ○ | ○ | 1000 | - | - | R/W | NO | 1000 |
| mn26 | Target position of the axis specified (P (II)) (Low word) | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| mn27 | Target position of the axis specified (P (II)) (High word) | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| mn28 | Speed at which the axis specified rotates (V (II)) (Low word) | ○ | ○ | ○ | ○ | 2000 | - | - | R/W | NO | 2000 |
| mn29 | Speed at which the axis specified rotates (V (II)) (High word) | | | | | | | | | | |
| mn30* | Operation command | ○ | ○ | ○ | ○ | 0 | - | 0 | R/W | NO | 0 |
| mn31* | Mode of operation | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| mn32 | Present command position of the axis specified (Pulse) (Low word) | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| mn33 | Present command position of the axis specified (Pulse) (High word) | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |

| SR number | Function | Applicable model*1 | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Latching | Default |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | | |
| mn34 | Present command speed of the axis specified (PPS) (Low word) | ○ | ○ | ○ | ○ | 0 | 0 | 0 | R/W | NO | 0 |
| mn35 | Present command speed of the axis specified (PPS) (High word) | | | | | | | | | | |
| mn36 | Present command position of the axis specified (Unit) (Low word) | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| mn37 | Present command position of the axis specified (Unit) (High word) | | | | | | | | | | |
| mn38 | Present command speed of the axis specified (Unit) (Low word) | ○ | ○ | ○ | ○ | 0 | 0 | 0 | R/W | NO | 0 |
| mn39 | Present command speed of the axis specified (Unit) (High word) | | | | | | | | | | |
| mn40* | State of the axis specified | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| mn41* | Axis error code | ○ | ○ | ○ | ○ | 0 | - | - | R | NO | 0 |
| mn42 | Electronic gear ratio of the axis specified (Numerator) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 1 |
| mn43 | Electronic gear ratio of the axis specified (Denominator) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 1 |
| mn44 | Frequency of pulses generated by the manual pulse generator for the axis specified (Low word) | ○ | ○ | ○ | ○ | 0 | 0 | - | R/W | NO | 0 |
| mn45 | Frequency of pulses generated by the manual pulse generator for the axis specified (High word) | ○ | ○ | ○ | ○ | 0 | 0 | - | R/W | NO | 0 |
| mn46 | Number of pulses generated by the manual pulse generator for the axis specified (Low word) | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |

| SR number | Function | Applicable model[*1] | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Latching | Default |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | | |
| mn47 | Number of pulses generated by the manual pulse generator for the axis specified (High word) | ○ | ○ | ○ | ○ | 0 | - | - | R/W | NO | 0 |
| mn48 | Response speed of the manual pulse generator for the axis specified | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 5 |
| mn50 | Electrical zero of the axis specified (Low word) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 0 |
| mn51 | Electrical zero of the axis specified (High word) | ○ | ○ | ○ | ○ | - | - | - | R/W | NO | 0 |
| mn68 | Present position of the servo encoder for the axis specified on a DMCNET (Low word) | X | X | X | ○ | - | - | - | R | NO | 0 |
| mn69 | Present position of the servo encoder for the axis specified on a DMCNET (High word) | X | X | X | ○ | - | - | - | R | NO | 0 |
| mn72 | Command sent to the servo drive for the axis specified on a DMCNET | X | X | X | ○ | - | - | - | R/W | NO | 0 |
| mn73 | Status of the servo drive for the axis specified on a DMCNET | X | X | X | ○ | - | - | - | R | NO | 0 |
| mn74 | Servo drive error code (Low word) | X | X | X | ○ | - | - | - | R/W | NO | 0 |
| mn75 | Servo drive error code (High word) | X | X | X | ○ | - | - | - | R/W | NO | 0 |
| mn76 | Writing data into the servo drive for the axis specified on a DMCNET/Reading data from the servo drive for the axis specified on a DMCNET (Low word) | X | X | X | ○ | - | - | - | R/W | NO | 0 |

3

| SR number | Function | Applicable model*1 | | | | OFF ⇩ ON | STOP ⇩ RUN | RUN ⇩ STOP | Attribute | Latching | Default |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 05M | 15M | 10M | 20M | | | | | | |
| mn77 | Writing data into the servo drive for the axis specified on a DMCNET/Reading data from the servo drive for the axis specified on a DMCNET (High word) | X | X | X | ○ | - | - | - | R/W | NO | 0 |
| mn78 | Parameter position in the servo drive for the axis specified on a DMCNET | X | X | X | ○ | - | - | - | R/W | NO | 0 |
| | Way in which the axis specified on a DMCNET returns home | X | X | X | ○ | - | - | - | R/W | NO | 0 |

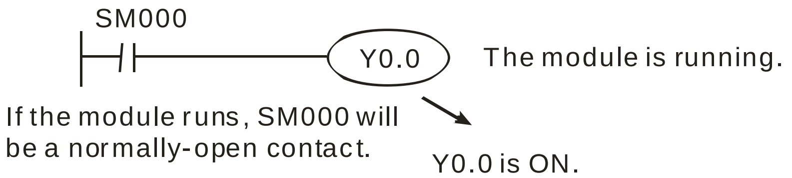
*1: 05M=AH05PM-5A; 10M=AH10PM-5A; 15M=AH15PM-5A; 20M=AH20MC-5A

## 3.13 Functions of Special Auxiliary Relays and Special Data Registers

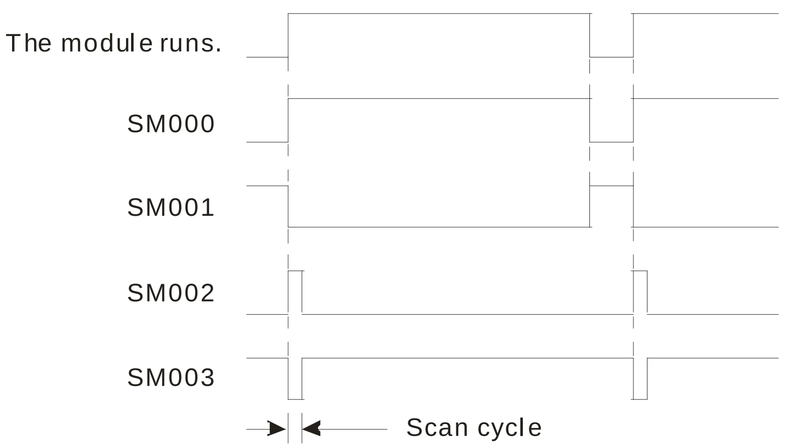
**Operation flags**

SM000~SM003

1. SM000: If the module runs, SM000 will be a normally-open contact (Form A contact). When the module runs, SM000 is ON.



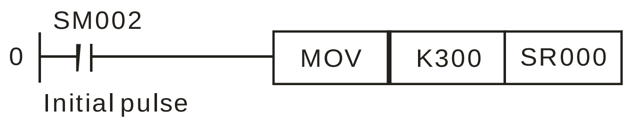

2. SM001: If the module runs, SM001 will be a normally-closed contact (Form B contact). When the module runs, SM001 is OFF.
3. SM002: A positive-going pulse is generated at the time when the module runs. The width of the pulse is equal to the scan cycle. If users want to initialize the module, they can use the contact.
4. SM003: A negative-going pulse is generated at the time when the module runs. The width of the pulse is equal to the scan cycle.



**Watchdog timer**

SR000

1. The watchdog timer is used to monitor a scan cycle. If the scan cycle is greater than the watchdog timer value, the ERROR LED indicator on the module will be turned ON, and all the output devices will be turned OFF.
2. The watchdog timer is initailly set to 200. If the program is long, or the operation is complex, users can change the watchdog timer value by means of the instruction MOV. In the example below, the watchdog timer value is changed to 300.

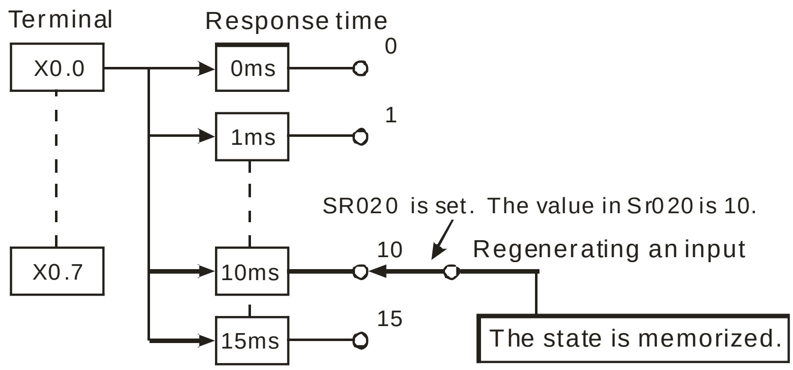



3. The maximum value which can be stored in the watchdog timer is 32,767. However, the larger the watchdog timer value is, the more time it takes to detect any operation error. As a result, if there is no complex operation resulting in a scan cycle longer than 200 milliseconds, it is suggested that the watchdog timer value should be less than 200.
4. If an opration is complex, the scan cycle may be long. Users can check whether the scan cycle is greater than the value stored in SR000 by monitoring SR010~SR012. If the scan cycle is greater than the value stored in SR000, the users can change the value in SR000.

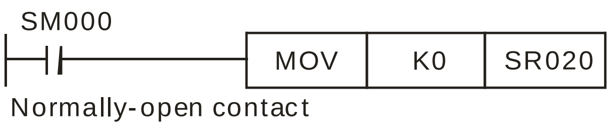
**Input filter**

SR020

1. Users can set the time it takes for the input terminals to respond by setting SR020. The value in SR020 must be in the range of 0 to 20. (Unit: ms)
2. If the module is turned form OFF to ON, the value in SR020 will automatically become 10, and the value in SR021 will automatically become 10.




3. If the program below is executed, the time is takes for the input terminal to respond will be 0 milliseconds. Owing to the fact that the input terminals are connected to resistor–capacitor circuits in series, the shortest time it takes for the input terminals to respond is 50 microseconds.

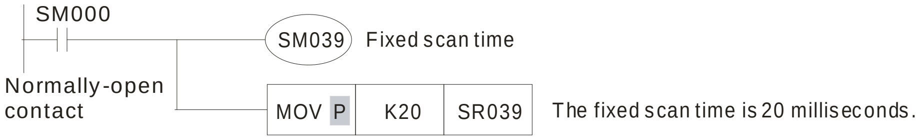



4. If high-spedd counters and interrupts are used in a program, the value in SR020 does not have any effect.

### Fixed scan time

SM039 and SR039

1. If SM039 is ON, the time it takes for the program to be scanned will depend on the value in SR039. If the execution of a program is complete, the program will not be scanned again until the fixed scan time set elapses. If the value in SR039 is less than the time it takes for a program to be scanned, the time it takes for a program to be scanned, will be given priority.




2. The values stored in SR010~SR012 include the value stored in SR039.

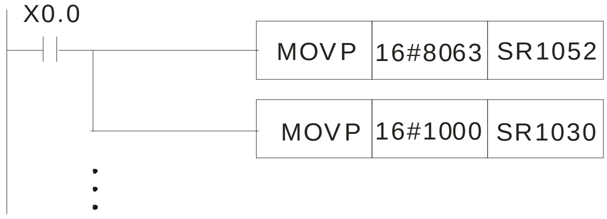
### Setting an OX motion subroutine number

SR1052

Users can specify an Ox motion subroutine by setting SR1052. The steps of setting SR1052 are as follows.

1. The users have to set bit 14 in SR1052 to 1, set bit 15 in SR1052 to 1, or set bit 14 and bit 15 in SR1052 to 1. Besides, the users have to write K99 (16#63) into bit 0~bit 13 in SR1052, that is, the Ox motion subroutine number specified is Ox99. To sum up, the users have to write 16#8063 into SR1052.
2. After the users write 16#1000 to SR1030, the Ox motion subroutine specified by SR1052 will be executed.

The program is shown below.




In the main program O100, X0.0 starts the motion subroutine Ox99.

### High-speed counting

SM200 and C200
SM204 and C204
SM208 and C208
SM212 and C212
SM216 and C216
SM220 and C220

There are six high-speed counters.

| Number | Counter number | Mode of counting | | External resetting terminal | External input terminal*2 |
|---|---|---|---|---|---|
| | | Device | Setting value*1 | | |
| 1 | C200 | K1SM200 | 0: U/D<br>1: P/D<br>2: A/B<br>(One time the frequency of A/B-phase inputs)<br>3: 4A/B<br>(Four times the frequency of A/B-phase inputs) | X0.0+ and X0.0-<br>SM203 | X0.8, X0.9, and S/S |
| 2 | C204 | K1SM204 | | X0.1+ and X0.1-<br>SM207 | X0.10, X0.11, and S/S |
| 3 | C208 | K1SM208 | | X0.2+ and X0.2-<br>SM211 | X0.12, X0.13, and S/S |
| 4 | C212 | K1SM212 | | X0.3+ and X0.3-<br>SM215 | X0.14, X0.15, and S/S |
| 5 | C216 | K1SM216 | | X0.2+ and X0.2-<br>SM219 | X0.12, X0.13, and S/S |
| 6 | C220 | K1SM220 | | X0.3+ and X0.3-<br>SM223 | X0.14, X0.15, and S/S |

*1. U/D: Counting up/Counting down; P/D: Pulse/Direction; A/B: A phase/B phase

*2. The input terminals of AH05PM-5A/AH10PM-5A are transistors whose collectors are open collectors. The input terminals of AH20MC-5A are differential input terminals. X0.8 and X0.9 on AH15PM-5A are differential input terminals. X0.10~X0.15 on AH15PM-5A are transistors whose collectors are open collectors.

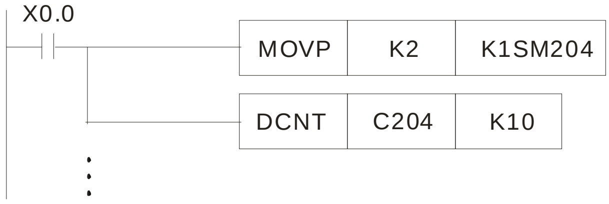
**High-speed counting**

SM200 and C200
SM204 and C204
SM208 and C208
SM212 and C212
SM216 and C216
SM220 and C220

The steps of setting the second counter are as follows.

1. Write K2 into K1SM204.
2. Enable C204.

The program for step 1 and step 2 is shown below.

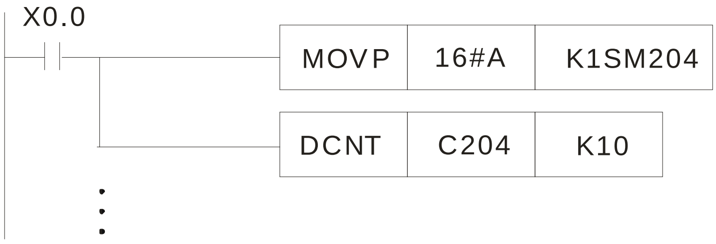



3. If users want to clear the present counter value by means of an external signal, they have to write 16#A into K1SM204.

| SM207 | SM206 | SM205 | SM204 |
|---|---|---|---|
| 1 | 0 | 1 | 0 |

4. C204 is enabled. If X0.1 is ON, the present value of C204 will become zero.

The program for step 3 and step 4 is shown below.




**High-speed timing**

SM200 and C201
SM204 and C205
SM208 and C209
SM212 and C213

There are four high-speed timers.

<table>
<tr><th rowspan="2">Number</th><th rowspan="2">Counter</th><th colspan="2">Mode of measuring time</th><th rowspan="2">External signal</th><th rowspan="2">Storage device</th></tr>
<tr><th>Device</th><th>Setting value</th></tr>
<tr><td>1</td><td>C200</td><td>K1SM200</td><td rowspan="4"><table><tr><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>x</td><td>Enabling a timer</td><td>x</td><td>Selecting a mode</td></tr></table>Bit 2: Enabling a timer<br>Bit 0: (1) 0: General mode (The interval between the rising edge of a pulse and the falling edge of the pulse is measured.)<br>(2) 1: Cyclic mode (The interval between the rising edge of a pulse and the rising edge of the next pulse is measured.)</td><td>X0.0+ and X0.0-</td><td>C201</td></tr>
<tr><td>2</td><td>C204</td><td>K1SM204</td><td>X0.1+ and X0.1-</td><td>C205</td></tr>
<tr><td>3</td><td>C208</td><td>K1SM208</td><td>X0.2+ and X0.2-</td><td>C209</td></tr>
<tr><td>4</td><td>C212</td><td>K1SM212</td><td>X0.3+ and X0.3-</td><td>C213</td></tr>
</table>

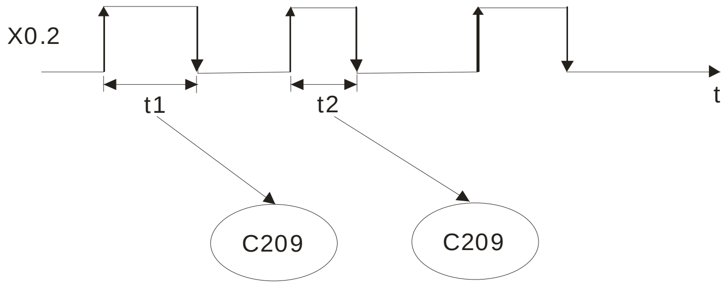
3

**High-speed timing**

SM200 and C201
SM204 and C205
SM208 and C209
SM212 and C213

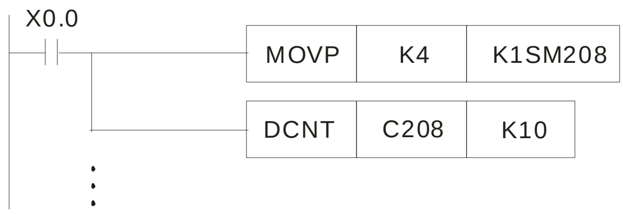
Example 1: Using the third timer in general mode

1. Users have to select the general mode, and enable the timer, that is, they have to write K4 into K1SM208.
2. C208 is enabled. The interval between the rising edge of a pulse received through X0.2 and the falling edge of the pulse is measured. The interval is written into C209. (Unit: 0.01 microseconds)



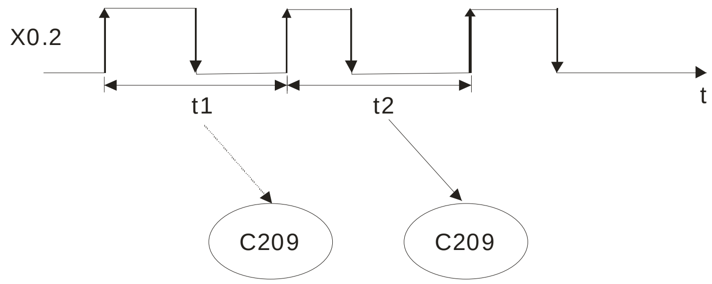

General mode
Unit: 0.01 μs

The program is shown below.



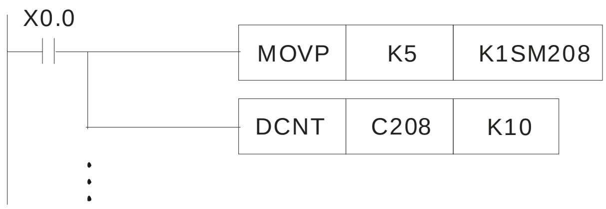

Example 2: Using the third timer in cyclic mode

1. Users have to write K5 into K1SM208.
2. C208 is enabled. The interval between the rising edge of a pulse received through X0.2 and the rising edge of the next pulse is measured. The interval is written into C209. (Unit: 0.01 microseconds)




Cyclic mode
Unit: 0.01 μs

The program is shown below.




3. The cyclic mode is used to measure a frequency.

**Turning the X devices ON/OFF**

SM304

If SM304 in an AH500 series motion control module is ON, the X devices in the AH500 series motion control module can be turned ON/OFF by means of PMSoft.

**Interrupt register**

SR400 and SR401

1. SR400 is an interrupt register. If users set a bit in SR400 to ON, an interrupt will be enabled.

| Bit | Interrupt | Interrupt number |
|---|---|---|
| 0 | Time interrupt | I0 |
| 1 | External terminal X0.8 | I1 |
| 2 | External terminal X0.9 | I2 |
| 3 | External terminal X0.10 | I3 |
| 4 | External terminal X0.11 | I4 |
| 5 | External terminal X0.12 | I5 |
| 6 | External terminal X0.13 | I6 |
| 7 | External terminal X0.14 | I7 |
| 8 | External terminal X0.15 | I8 |

2. If an interrupt enabled is a time interrupt, users can write the cycle of the interrupt into SR401.
3. There are two types of interrupts.
   - External interrupt: If an interrupt is triggered by the rising edge/falling edge of a pulse received through an external terminal, the execution of the present program will stop, and the interrupt will be executed. After an interrupt is executed, the program which is executed before the interrupt is triggered will be executed.
   - Time interrupt: The execution of the present program stops at regular intervals. Whenever the execution of the present program stops, an interrupt is executed.

**Clearing the M-code which is executed**

SM1050 and SR703

- ◆ If users want to clear the M-code in SR703, they have to set SM1050 to OFF.
- ◆ If an M code in an Ox motion subroutine is executed, SM1050 will be ON. The M-code which is executed is stored in SR703.

**Ready flag**

SMmn48 (SM1048, SM1148, ……SM2548)

1. Every motion axis uses a ready flag. The first axis uses SM1048, the second axis uses SM1148, the third axis uses SM1248, the fourth axis uses SM1348, ……, the fifteenth axis uses SM2448, and sixteenth axis uses SM2548. Users can use the ready flags to judge whether the axes operate.
2. Description of the ready flag for the first axis: Before the first axis operates, SM1048 is ON. When the fisrt axis operates, SM1048 is OFF. After the first axis finishes operating, SM1048 is ON.

Stopping the uniaxial motion at the angle specified

SMmn17
(SM1017, SM1117, ……SM2517)

1. Stopping uniaxial motion at an angle specified

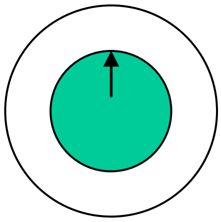
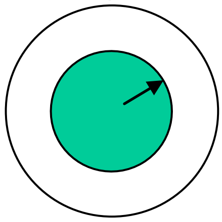
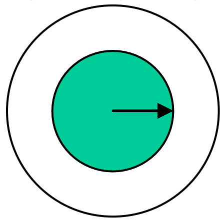
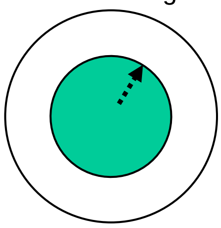
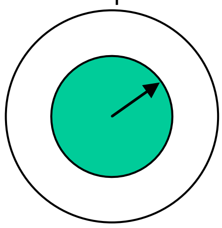
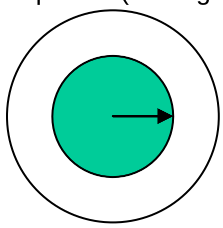
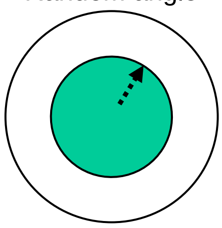
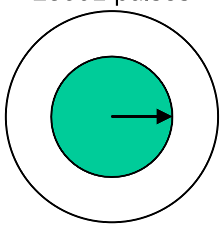
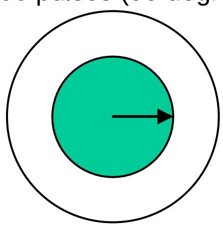
| Parameter | 1st axis | 2nd axis | 3rd axis | 4th axis |
|---|---|---|---|---|
| Stopping at the angle specified | SM1017 | SM1117 | SM1217 | SM1317 |
| Angle | SR1023~ SR1022 | SR1123~ SR1122 | SR1223~ SR1222 | SR1323~ SR1322 |
| Number of pulses per cycle | SR1027~ SR1026 | SR1127~ SR1126 | SR1227~ SR1226 | SR1327~ SR1326 |
| **Parameter** | **5th axis** | **6th axis** | **7th axis** | **8th axis** |
| Stopping at the angle specified | SM1417 | SM1517 | SM1617 | SM1717 |
| Angle | SR1423~ SR1422 | SR1523~ SR1522 | SR1623~ SR1622 | SR1723~ SR1722 |
| Number of pulses per cycle | SR1427~ SR1426 | SR1527~ SR1526 | SR1627~ SR1626 | SR1727~ SR1726 |
| **Parameter** | **9th axis** | **10th axis** | **11th axis** | **12th axis** |
| Stopping at the angle specified | SM1817 | SM1917 | SM2017 | SM2117 |
| Angle | SR1823~ SR1822 | SR1923~ SR1922 | SR2023~ SR2022 | SR2123~ SR2122 |
| Number of pulses per cycle | SR1827~ SR1826 | SR1927~ SR1926 | SR2027~ SR2026 | SR2127~ SR2126 |
| **Parameter** | **13th axis** | **14th axis** | **15th axis** | **16th axis** |
| Stopping at the angle specified | SM2217 | SM2317 | SM2417 | SM2517 |
| Angle | SR2223~ SR2222 | SR2323~ SR2322 | SR2423~ SR2422 | SR2523~ SR2522 |
| Number of pulses per cycle | SR2227~ SR2226 | SR2327~ SR2326 | SR2427~ SR2426 | SR2527~ SR2526 |

2. In JOG+ mode, users can stop the first axis~the sixteenth axis at particular angles.
3. If users want to stop the first axis at a particular angle, they have to set SM1017 to ON, write the number of pulses per cycle into SR1027 and SR1026, and write an angle in SR1023 and SR1022, and the first axis have to be in JOG+ mode.

**Stopping the uniaxial motion at the angle specified**

SMmn17
(SM1017, SM1117, ……SM2517)

4. If the number of pulses it takes for the motor of an axis to rotate once is 20000, and the angle at which users want to stop the axis is 90 degrees, there will be the states shown below.

| Starting position | Stop position of the JOG motion | Final stop position |
|---|---|---|
| Zero degrees  | 4000 pulses  | 5000 pulses (90 degrees)  |
| Random angle  | 63500 pulses  | 65000 pulses (90 degrees)  |
| Random angle  | 25001 pulses  | 45000 pulses (90 degrees)  |

**Clearing the motion error**

SMmn49
(SM1049, SM1149, ……SM2549)
SRmn41
(SR1041, SR1141, ……SR2541)

If errors occur in axis 1~axis 16, the SM devices correspond to the axes will be ON, and the error messages which appear will be stored in the SR devices correspond to the axes.

If users want to eliminate the error occuring in an axis, they have to clear the error code in the SR device corresponding to the axis, and reset the SM device corresponding to the axis.

**Continuous interpolation**

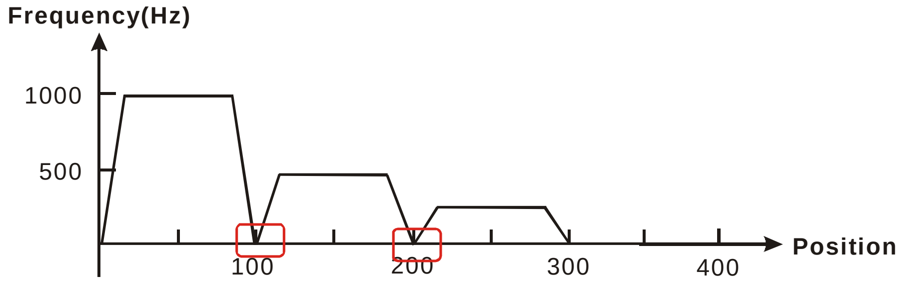
SR797 and SR796

If users set the minimum speed to which the speed of continuous interpolation decreses in (SR797, SR796), the smaller speed will be taken as a turning point after the setting value in (SR797, SR796) is compared with the acutal speed to which the speed of continuous interpolation decreses.

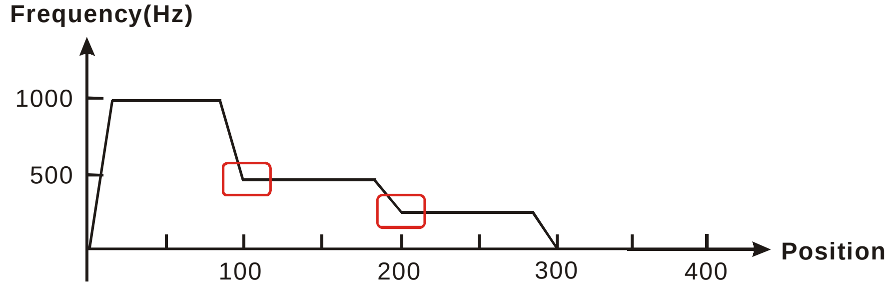
**Continuous interpolation**

SR797 and SR796

If the value in (SR797, SR796) is K0 (there is no continuous interpolation), the speed of motion will decrease to 0 Hz no matter what the actual deceleration is.




The value in (SR797, SR796) is K500. After the value in (SR797, SR796) is compared with the acutal deceleration, the smaller deceleration will be taken as a turning point.

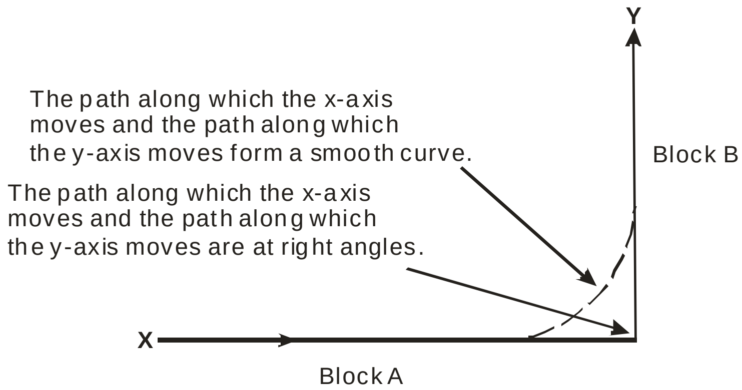



Continuous path:

If (SR797, SR496) is not set, the path along which the x-axis moves and the path along which the y-axis moves will be at right angles. If (SR797, SR496) is set, the path along which the x-axis moves and the path along which the y-axis moves will form a smooth curve.

G01 X100 F1000; (Block A)

Y100; (Block B)




Block A: Path along which the x-axis moves; Block B: Path along which the y-axis moves

**Setting the percentage for the values of the speed parameters of the G-codes**

SR798

1. If the value in SR798 is 100, the speeds of the G-codes used will be the orginial speeds. If the value in SR798 is 1000, the speeds of the G-codes used will be multipled by 10. If the value in SR798 is 50, the speeds of the G-codes used will be half the original speed.
2. If the result gotten from the multiplication of the speed of a G-code by the percentage set in SR798 is greater than 500000 Hz, the G-code will move the axes used at a speed of 500000 Hz.

**Setting the polarities of the input terminals**

SR799

If users want to turn an input terminal into a Form A contact, they have to set the bit corresponding to the input terminal to OFF. If the users want to turn an input terminal into a Form B contact, they have to set the bit corresponding to the input terminal to ON.

| Bit# | Polarity | Bit# | Polarity |
|---|---|---|---|
| 0 | X0.0 | 8 | X0.8 |
| 1 | X0.1 | 9 | X0.9 |
| 2 | X0.2 | 10 | X0.10 |
| 3 | X0.3 | 11 | X0.11 |
| 4 | X0.4[*1] | 12 | X0.12 |
| 5 | X0.5[*1] | 13 | X0.13 |
| 6 | X0.6[*1] | 14 | X0.14 |
| 7 | X0.7[*1] | 15 | X0.15 |

*1: Only AH15PM-5A has X0.4, X0.5, X0.6, and X0.7.

**Reading the states of the input terminals**

SR800

If a bit in SR800 is ON, the input terminal corresponding to the bit receives a signal. If a bit in SR800 is OFF, the input terminal corresponding to the bit does not receive a signal.

| Bit# | State | Bit# | State |
|---|---|---|---|
| 0 | X0.0 | 8 | X0.8 |
| 1 | X0.1 | 9 | X0.9 |
| 2 | X0.2 | 10 | X0.10 |
| 3 | X0.3 | 11 | X0.11 |
| 4 | X0.4[*1] | 12 | X0.12 |
| 5 | X0.5[*1] | 13 | X0.13 |
| 6 | X0.6[*1] | 14 | X0.14 |
| 7 | X0.7[*1] | 15 | X0.15 |

*1: Only AH15PM-5A has X0.4, X0.5, X0.6, and X0.7.

**Setting a filter coefficient for the input terminals**

SR806

1. Users can set the hardware filter for the input terminals by means of SR806.
2. Users can set a filter coefficient for the input terminals of an AH500 series motion control module by setting the low byte in SR806.
3. Filter coefficeint $= \dfrac{85000}{2^{N+4}}$ (kHz); N=1~19

| N | kHz | N | kHz |
|---|---|---|---|
| 1 | 2656.25 | 11 | 2.593994 |
| 2 | 1328.125 | 12 | 1.296997 |
| 3 | 664.0625 | 13 | 0.648499 |
| 4 | 332.0313 | 14 | 0.324249 |
| 5 | 166.0156 | 15 | 0.162125 |
| 6 | 83.00781 | 16 | 0.081062 |
| 7 | 41.50391 | 17 | 0.040531 |
| 8 | 20.75195 | 18 | 0.020266 |
| 9 | 10.37598 | 19 | 0.010133 |
| 10 | 5.187988 | | |

4. If the value in SR806 is 0, no signals will be filtered.
5. If the value in SR806 is 16#000A, the filter coefficient for X0.0, X0.1, X0.2, X0.3, X0.8, X0.9, X0.10, X0.11, X0.12, X0.13, X0.14, and X0.15 $= \dfrac{85000}{2^{10+4}} = 5.187988$, and the signals whose frequencies are higher than 5.187988 kHz will be removed.

O100 error

SM953, SR802, and SR803

1. If an error occurs in O100, SM953 will be ON, the error code corresponding to the error will be stored in SR802, and the step address at which the error occurs will be stored in SR803.
2. Please refer to appendix A for more information about error codes.

Setting the parameters of the axis

SRmn00 (SR1000, SR1100, ……SR2500)

3

SR1000 is for the first axis, SR1100 is for the second axis, SR1200 is for the third axis, SR1300 is for the fourth axis, ……, SR2400 is for the fifteenth axis, and SR2500 is for the sixteenth axis.

| Bit# | Parameter of the axis | Bit# | Parameter of the axis |
|---|---|---|---|
| 0 | Unit*1 | 8 | Direction in which the axis returns home*3 |
| 1 | | 9 | Mode of returning home*3 |
| 2 | - | 10 | Mode of triggering the return to home*3 |
| 3 | - | 11 | Direction in which the motor rotates*3 |
| 4 | Output type*2 | 12 | Relative/Absolute coordinates*3 |
| 5 | | 13 | Mode of triggering the calculation of the target position*3 |
| 6 | - | 14 | Curve*3 |
| 7 | - | 15 | - |

*1:

| b1 | b0 | Unit |
|---|---|---|
| 0 | 0 | Motor unit |
| 0 | 1 | Mechanical unit |
| 1 | 0 | Compound unit |
| 1 | 1 | |

| | Motor unit | Compound unit | Mechanical unit |
|---|---|---|---|
| Position | pulse | μm | |
| | pulse | mdeg | |
| | pulse | $10^{-4}$ inches | |
| Speed | pulse/second | | centimeter/minute |
| | pulse/second | | 10 degrees/minute |
| | pulse/second | | inch/minute |

*2:

| b5 | b4 | Description |
|---|---|---|
| 0 | 0 | Positive-going pulse+Negative-going pulse |
| 0 | 1 | Pulse+Direction |
| 1 | 0 | A/B-phase pulse (two phases and two inputs) |
| 1 | 1 | |

Setting the parameters of the axis

SRmn00
(SR1000, SR1100, ……SR2500)

*3:

| Bit# | Description |
|---|---|
| 8 | Bit 8=0: The value indicating the present position of the axis decreases progressively.<br>Bit 8=1: The value indicating the present position of the axis increases progressively. |
| 9 | Bit 9=0: Normal mode ; bit 9=1: Overwrite mode |
| 10 | Bit 10=0: The return to home is triggered by a transition in DOG's signal from high to low.<br>Bit 10=1: The return to home is triggered by a transition in DOG's signal from low to high. |
| 11 | Bit 11=0: When the motor rotates clockwise, the value indicating the present position of the axis increases.<br>Bit 11=1: When the motor rotates clockwise, the value indicating the present position of the axis decreases. |
| 12 | Bit 12=0: Absolute coordinates<br>Bit 12=1: Relative coordinates |
| 13 | Bit 13=0: The calculation of the target position of the axis is triggered by a transition in DOG's signal from low to high.<br>Bit 13=1: The calculation of the target position of the axis is triggered by a transition in DOG's signal from high to low.<br>(The setting of bit 13 is applicable to the insertion of single-speed motion, and the insertion of two-speed motion.) |
| 14 | Bit 14=0: Trapezoid curve<br>Bit 14=1: S curve |

Operation command

SRmn30
(SR1030, SR1130, ……SR2530)

SR1030 is for the first axis, SR1130 is for the second axis, SR1230 is for the third axis, SR1330 is for the fourth axis, ……, SR2430 is for the fifteenth axis, and SR2530 is for the sixteenth axis.

| Bit# | Operation command | Bit# | Operation command |
|---|---|---|---|
| **0** | The motion of the axis specified is stopped by software. | **8** | A mode of single-speed motion is activated. |
| **1** | - | **9** | A mode of inserting single-speed motion is activated. |
| **2** | The axis specified operates in JOG+ mode. | **10** | A mode of two-speed motion is activated. |
| **3** | The axis specified operates in JOG- mode. | **11** | A mode of inserting two-speed motion is activated. |
| **4** | A mode of variable motion is activated. | **12** | 0: The execution of the Ox motion subroutine set stops.<br>1: The execution of the Ox motion subroutine set starts. |
| **5** | A manual pulse generator is operated. | **13** | - |
| **6** | A mode of triggering the return to home is activated. | **14** | - |
| **7** | - | **15** | - |

The input terminals for the manual pulse generator MPG are X0.8+, X0.8-, X0.9+, and X0.9-.

Mode of operation

SRmn31
(SR1031, SR1131, ……SR2531)

SR1031 is for the first axis, SR1131 is for the second axis, SR1231 is for the third axis, SR1331 is for the fourth axis, ……, SR2431 is for the fifteenth axis, and SR2531 is for the sixteenth axis.

| Bit# | Mode of operation | Bit# | Mode of operation |
|---|---|---|---|
| 0 | - | 8 | - |
| 1 | - | 9 | - |
| 2 | Mode of sending a CLR signal | 10 | - |
| 3 | - | 11 | - |
| 4 | - | 12 | - |
| 5 | - | 13 | - |
| 6 | - | 14 | - |
| 7 | - | 15 | Restoring the module to the factory settings |

| Bit# | Description |
|---|---|
| 2 | Bit 2=0: After the axis returns home, the CLR output will send a 130 millisecond signal to the servo drive, and the present position of the servo drive which is stored in a register in the servo drive will be cleared.<br>Bit 2=1: The CLR output functions as a general output. Its state is controlled by bit 3. |
| 15 | Bit 15=1: The values of parameters are restored to factory settings. |

State of the axis

SRmn40
(SR1040, SR1140, ……SR2540)

SR1040 is for the first axis, SR1140 is for the second axis, SR1240 is for the third axis, SR1340 is for the fourth axis, ……, SR2440 is for the fifteenth axis, and SR2540 is for the sixteenth axis.

| Bit# | State of the axis |
|---|---|
| 0 | Positive pulses are being output. |
| 1 | Negative pulses are being output. |
| 2 | The axis is being operating. |
| 3 | An error occurs. |
| 4 | The axis pauses. |
| 5 | - |
| 6 | - |
| 7 | - |

## 3.14 Special Data Registers for Motion Axes

The special data registers for motion axis 1~motion axis 16 are described below. Please refer to Chapter 7 for more information about the setting of the special data registers.

| SR number*1 | | Special data register | Setting range | Default value |
|---|---|---|---|---|
| HW | LW | | | |
| - | mn00 | Setting the parameters of the axis specified | Bit 0~bit 15 | 16#0 |
| - | mn01 | Compensation value for the axis specified | Users have to set a value according to their needs. | 16#0 |
| mn03 | mn02 | Number of pulses it takes for the motor of the axis specified to rotate once (A) | 1~2,147,483,647 pulses/revolution | K2,000 |
| mn05 | mn04 | Distance generated after the motor of the axis specified rotate once (B) | 1~2,147,483,647*2 | K1,000 |

| SR number*1 | | Special data register | Setting range | Default value |
|---|---|---|---|---|
| HW | LW | | | |
| mn07 | mn06 | Maximum speed (**$V_{MAX}$**) at which the axis specified rotates | 0~2,147,483,647*3 | K500,000 |
| mn09 | mn08 | Start-up speed (**$V_{BIAS}$**) at which the axis specified rotates | 0~2,147,483,647*3 | K0 |
| mn11 | mn10 | JOG speed (**$V_{JOG}$**) at which the axis specified rotates | 0~2,147,483,647*3 | K5,000 |
| mn13 | mn12 | Speed (**$V_{RT}$**) at which the axis specified returns home | 0~2,147,483,647*3 | K50,000 |
| mn15 | mn14 | Speed (**$V_{CR}$**) to which the speed of the axis specified decreases when the axis returns home | 0~2,147,483,647*3 | K1,000 |
| - | mn16 | Number of zero signals for the axis specified | 0~32,767 pulses | K0 |
| - | mn17 | Supplementary pulses for the axis specified | -32,768~+32,767 PLS | K0 |
| mn19 | mn18 | Home position of the axis specified | 0~±999,999 | K0 |
| - | mn20 | Time ($T_{ACC}$) it takes for the axis specified to accelerate | 10~32,767 ms | K100 |
| - | mn21 | Time ($T_{DEC}$) it takes for the axis specified to decelerate | 10~32,767 ms | K100 |
| mn23 | mn22 | Target position of the axis specified (P (I)) | -2,147,483,648~+2,147,483,647 | K0 |
| mn25 | mn24 | Speed at which the axis specified rotates (V (I)) | 0~2,147,483,647 | K1000 |
| mn27 | mn26 | Target position of the axis specified (P (II)) | -2,147,483,648~+2,147,483,647 | K0 |
| mn29 | mn28 | Speed at which the axis specified rotates (V (II)) | 0~2,147,483,647*2 | K2,000 |
| - | mn30 | Operation command | Bit 0~bit 15 | 16#0 |
| - | mn31 | Mode of operation | Bit 0~bit 15 | 16#0 |
| mn33 | mn32 | Present command position of the axis specified (Pulse) | -2,147,483,648~+2,147,483,647 | K0 |
| mn35 | mn34 | Present command speed of the axis specified (PPS) | 0~2,147,483,647 PPS | K0 |
| mn37 | mn36 | Present command position of the axis specified (unit *3) | -2,147,483,648~+2,147,483,647 | K0 |
| mn39 | mn38 | Present command speed of the axis specified (unit *3) | 0~2,147,483,647 PPS | K0 |
| - | mn40 | State of the axis specified | Bit 0~bit 15 | 16#0 |
| - | mn41 | Axis error code | Please refer to appendix A for more information. | 16#0 |
| - | mn42 | Electronic gear of the axis specified (Numerator) | 1~32,767 | K1 |
| - | mn43 | Electronic gear of the axis specified (Denominator) | 1~32,767 | K1 |
| mn45 | mn44 | Frequency of pulses generated by the manual pulse generator for the axis specified | Frequency of pulses generated by the manual pulse generator for the axis specified | K0 |
| mn47 | mn46 | Number of pulses generated by the manual pulse generator for the axis specified | Number of pulses generated by the manual pulse generator for the axis specified | K0 |

| SR number*1 | | Special data register | Setting range | Default value |
|---|---|---|---|---|
| HW | LW | | | |
| - | mn48 | Response speed of the manual pulse generator for the axis specified | Response speed of the manual pulse generator for the axis specified | K5 |
| mn51 | mn50 | Electrical zero of the axis specified | Users have to set a value according to their needs. | K0 |
| mn69 | mn68 | Present position of the servo encoder for the axis specified on a DMCNET*4 | The values displayed in SRmn68 and SRmn59 vary with the setting of Delta ASDA-A2 series servo drive. | K0 |
| - | mn72 | Command sent to the servo drive for the axis specified on a DMCNET *4 | Users have to set a value according to their needs. | K0 |
| - | mn73 | Status of the servo drive for the axis specified on a DMCNET *4 | Users have to set a value according to their needs. | K0 |
| mn75 | mn74 | Servo drive error code *4 | Users have to set a value according to their needs. | K0 |
| mn77 | mn76 | Writing data into the servo drive for the axis specified on a DMCNET/Reading data from the servo drive for the axis specified on a DMCNET *4 | Users have to set a value according to their needs. | K0 |
| - | mn78 | Parameter position in the servo drive for the axis specified on a DMCNET *4 | Users have to set a value according to their needs. | K0 |

*1. HW: High word; LW: Low word; mn=10 (the first axis)~25 (the sixteenth axis)

*2. Unit: µm/rev, mdeg/rev, and $10^{-4}$ inches/rev

*3. The unit used varies with the setting of bit 0 and bit 1 in SRmn00.

*4. Only AH20MC-5A is supported. AH05PM-5A, AH15PM-5A and AH10PM-5A are not supported.

Special data registers for motion axis 1~motion axis 16:

Please refer to Chapter 7 for more information about the functions of the special data registers for motion axis 1~motion axis 16, and usage of the special data registers for motion axis 1~motion axis 16.

The Please refer to Chapter 14 for more information about the functions of the special data registers related to DMCNET in AH20MC-5A, and the usage of the special data registers related to DMCNET in AH20MC-5A.

# Chapter 4 Basic Instructions

## Table of Contents

# 4.1 Table of Basic Instructions

**General instructions**

| Instruction code | Function | Operand | Execution speed (us) | | | | Step | Page number |
|---|---|---|---|---|---|---|---|---|
| | | | 20MC*1 | 10PM*1 | 05PM*1 | 15PM*1 | | |
| LD | Loading a Form A contact | Xn.n, Yn.n, M, S, T, C | 0.22 | 0.22 | 0.22 | 0.11 | 3 | 4-4 |
| LDI | Loading a Form B contact | Xn.n, Yn.n, M, S, T, C | 0.22 | 0.22 | 0.22 | 0.11 | 3 | 4-4 |
| AND | Connecting a Form A contact in series | Xn.n, Yn.n, M, S, T, C | 0.22 | 0.22 | 0.22 | 0.11 | 3 | 4-5 |
| ANI | Connecting a Form B contact in series | Xn.n, Yn.n, M, S, T, C | 0.22 | 0.22 | 0.22 | 0.11 | 3 | 4-5 |
| OR | Connecting a Form A contact in parallel | Xn.n, Yn.n, M, S, T, C | 0.22 | 0.22 | 0.22 | 0.11 | 3 | 4-6 |
| ORI | Connecting a Form B contact in parallel | Xn.n, Yn.n, M, S, T, C | 0.22 | 0.22 | 0.22 | 0.11 | 3 | 4-6 |
| ANB | Connecting circuit blocks in series | None | 0.12 | 0.12 | 0.12 | 0.25 | 3 | 4-7 |
| ORB | Connecting circuit blocks in parallel | None | 0.12 | 0.12 | 0.12 | 0.25 | 3 | 4-7 |

4

**Output instructions**

| Instruction code | Function | Operand | Execution speed (us) | | | | Step | Page number |
|---|---|---|---|---|---|---|---|---|
| | | | 20MC*1 | 10PM*1 | 05PM*1 | 15PM*1 | | |
| OUT | Driving a coil | Yn.n, M, S | 0.17 | 0.17 | 0.17 | 0.11 | 3 | 4-8 |
| SET | Keeping a device ON | Yn.n, M, S | 0.39 | 0.39 | 0.39 | 0.23 | 3 | 4-8 |
| RST | Resetting a contact or a register | Yn.n, M, S, T, C, D, V, Z, W | 0.47 | 0.47 | 0.47 | 0.32 | 3 | 4-9 |

**Timer and counters**

| Instruction code | Function | Operand | Execution speed (us) | | | | Step | Page number |
|---|---|---|---|---|---|---|---|---|
| | | | 20MC*1 | 10PM*1 | 05PM*1 | 15PM*1 | | |
| TMR | 16-bit timer | T-K, T-D, T-W | 5.6 | 5.6 | 5.6 | 5.2 | 5 | 4-9 |
| CNT | 16-bit counter | C-K, C-D, C-W (16 bits) | 3.2 | 3.2 | 3.2 | 2.8 | 5 | 4-10 |
| DCNT | 32-bit counter | C-K, C-D, C-W (32 bits) | 3.8 | 3.8 | 3.8 | 2.8 | 6 | 4-10 |

### Rising-edge/Falling-edge detection instructions

| Instruction code | Function | Operand | Execution speed (us) | | | | Step | Page number |
|---|---|---|---|---|---|---|---|---|
| | | | 20MC*1 | 10PM*1 | 05PM*1 | 15PM*1 | | |
| LDP | Starting rising-edge detection | Xn.n, Yn.n, M, S, T, C | 0.39 | 0.39 | 0.39 | 0.23 | 3 | 4-11 |
| LDF | Starting falling-edge detection | Xn.n, Yn.n, M, S, T, C | 0.39 | 0.39 | 0.39 | 0.23 | 3 | 4-11 |
| ANDP | Connecting rising-edge detection in series | Xn.n, Yn.n, M, S, T, C | 0.39 | 0.39 | 0.39 | 0.23 | 3 | 4-12 |
| ANDF | Connecting falling-edge detection in series | Xn.n, Yn.n, M, S, T, C | 0.39 | 0.39 | 0.39 | 0.23 | 3 | 4-12 |
| ORP | Connecting rising-edge detection in parallel | Xn.n, Yn.n, M, S, T, C | 0.37 | 0.37 | 0.37 | 0.04 | 3 | 4-13 |
| ORF | Connecting falling-edge detection in parallel | Xn.n, Yn.n, M, S, T, C | 0.37 | 0.37 | 0.37 | 0.04 | 3 | 4-13 |

### Rising-edge/Falling-edge output instruction

| Instruction code | Function | Operand | Execution speed (us) | | | | Step | Page number |
|---|---|---|---|---|---|---|---|---|
| | | | 20MC*1 | 10PM*1 | 05PM*1 | 15PM*1 | | |
| PLS | Rising-edge output | Yn.n, M | 0.37 | 0.37 | 0.37 | 0.23 | 3 | 4-14 |
| PLF | Falling-edge output | Yn.n, M | 0.37 | 0.37 | 0.37 | 0.23 | 3 | 4-15 |

### Other instructions

| Instruction code | Function | Operand | Execution speed (us) | | | | Step | Page number |
|---|---|---|---|---|---|---|---|---|
| | | | 20MC*1 | 10PM*1 | 05PM*1 | 15PM*1 | | |
| P | Pointer | P0~P255 | – | – | – | – | 1 | 4-15 |

*1. 05PM=AH05PM-5A; 10PM=AH10PM-5A; 15PM=AH15PM-5A; 20MC=AH20MC-5A

## 4.2 Descriptions of the Basic Instructions

| Instruction code | Operand | Function |
|---|---|---|
| LD | **S** | Loading a Form A contact |

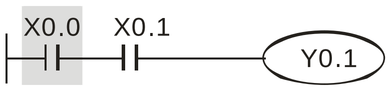
| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

Explanation

◆ The instruction LD applies to the Form A contact which starts from a busbar or the Form A contact which is the start of a circuit. It reserves the present contents, and stores the state which is gotten in the accumulation register.

Example

Ladder diagram:




| Instruction code: | | Description: |
|---|---|---|
| **LD** | **X0.0** | Loading the Form A contact X0.0 |
| AND | X0.1 | Connecting the Form A contact X0.1 in series |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| LDI | **S** | Loading a Form B contact |

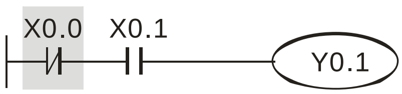
| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

Explanation

◆ The instruction LDI applies to the Form B contact which starts from a busbar or the Form B contact which is the start of a circuit. It reserves the present contents, and stores the state which is gotten in an accumulation register.

Example

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| **LDI** | **X0.0** | Loading the Form B contact X0.0 |
| AND | X0.1 | Connecting the Form A contact X0.1 in series |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| AND | **S** | Connecting a Form A contact in series |

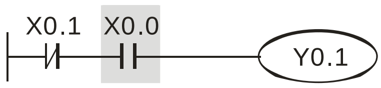
| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

**Explanation**

◆ The instruction AND is used to connect a Form A contact in series. It reads the state of a contact which is connected in series, and performs the AND operation on the previous logical operation result. The final result is stored in an accumulation register.

**Example**

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LDI | X0.1 | Loading the Form B contact X0.1 |
| **AND** | **X0.0** | Connecting the Form A contact X0.0 |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| ANI | **S** | Connecting a Form B contact in series |

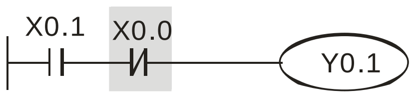
| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

**Explanation**

◆ The instruction ANI is used to connect a Form B contact in series. It reads the state of a contact which is connected in series, and performs the AND operation on the previous logical operation result. The final result is stored in an accumulation register.

**Example**

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.1 | Loading the Form A contact X0.1 |
| **ANI** | **X0.0** | Connecting the Form B contact X0.0 in series |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| OR | **S** | Connecting a Form A contact in parallel |

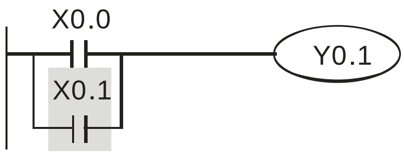
| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

Explanation

◆ The instruction OR is used to connect a Form A contact in parallel. It reads the state of a contact which is connected in parallel, and performs the OR operation on the previous logical operation result. The final result is stored in an accumulation register.

Example

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| **OR** | **X0.1** | Connecting the Form A contact X0.1 in parallel |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| ORI | **S** | Connecting a Form B contact in parallel |

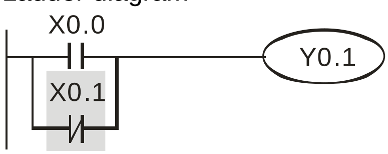
| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

Explanation

◆ The instruction ORI is used to connect a Form B contact in parallel. It reads the state of a contact which is connected in parallel, and performs the OR operation on the previous logical operation result. The final result is stored in an accumulation register.

Example

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| **ORI** | **X0.1** | Connecting the Form B contact X0.1 in parallel |
| OUT | Y0.1 | Driving the coil Y0.1 |

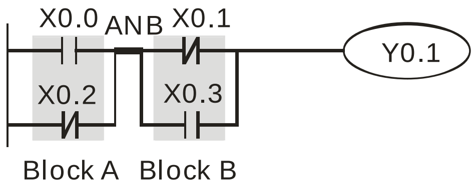
| Instruction code | Operand | Function |
|---|---|---|
| ANB | – | Connecting circuit blocks in series |

**Explanation**

- The instruction ANB is used to perform the AND operation on the logical operation result reserved previously and the contents of the present accumulation register.

**Example**

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| ORI | X0.2 | Connecting the Form B contact X0.2 in parallel |
| LDI | X0.1 | Loading the Form B contact X0.1 |
| OR | X0.3 | Connecting the Form A contact X0.3 in parallel |
| **ANB** | | Connecting the circuit blocks in series |
| OUT | Y0.1 | Driving the coil Y0.1 |

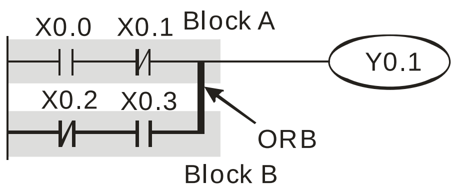
| Instruction code | Operand | Function |
|---|---|---|
| ORB | – | Connecting circuit blocks in parallel |

**Explanation**

- The instruction ORB is used to perform the OR operation on the logical operation result reserved previously and the contents of the present accumulation register.

**Example**

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| ANI | X0.1 | Connecting the Form B contact X0.1 in series |
| LDI | X0.2 | Loading the Form B contact X0.2 |
| AND | X0.3 | Connecting the Form A contact X0.3 in series |
| **ORB** | | Connecting the circuit blocks in parallel |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| OUT | S | Driving a coil |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | | ● | ● | ● | | | | | | | | | | | |

Explanation

- The logical operation result prior to the application of the instruction OUT is sent to the device specified.
- Action of a coil

| Operation result | OUT | | |
|---|---|---|---|
| | Coil | Contact | |
| | | Form A contact (Normally-open contact) | Form B contact (Normally-closed contact) |
| False | OFF | OFF | ON |
| True | ON | ON | OFF |

Example

Ladder diagram


| Instruction code: | | Description: |
|---|---|---|
| LDI | X0.0 | Loading the Form B contact X0.0 |
| AND | X0.1 | Connecting the Form A contact X0.1 in series |
| **OUT** | **Y0.1** | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| SET | S | Keeping a device ON |

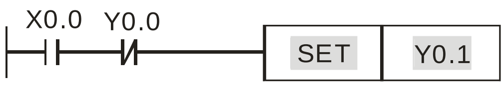
| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | | ● | ● | ● | | | | | | | | | | | |

Explanation

- When the instruction SET is driven, the device specified is set to ON. Whether the instruction SET is still driven or not, the device specified remains ON. Users can set the device specified to OFF by means of the instruction RST.

Example

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| ANI | Y0.0 | Connecting the Form B contact Y0.0 in series |
| **SET** | **Y0.1** | Y0.1 remains ON. |

| Instruction code | Operand | Function |
|---|---|---|
| RST | **S** | Resetting a contact or a register |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | ● | ● | ● | | | | | | ● | ● | ● | ● | ● | ● |

Explanation

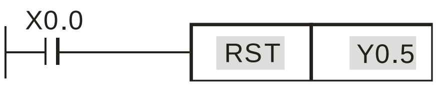
◆ When the instruction RST is driven, the device specified acts in the way described below.

| Device | State |
|---|---|
| S, Yn.n, M | The coil and the contact are set to OFF. |
| T, C | The present timer value or the present counter value becomes 0. The coil and the contact are set to OFF. |
| D, V, Z | The value becomes 0. |

◆ If the instruction RST is not executed, the state of the device specified will remain unchanged.

Example

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| **RST** | **Y0.5** | Resetting Y0.5 |

| Instruction code | Operand | Function |
|---|---|---|
| TMR | **$S_1$, $S_2$** | 16-bit timer |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | | | | | | ● | | | |
| **$S_2$** | | | | | ● | | | | | ● | ● | | | | |

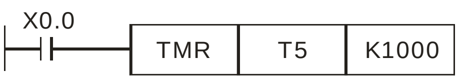
Explanation

◆ When the instruction TMR is executed, the coil specified is ON, and the timer specified begins to count. If the timer value matches the setting value (timer value≧setting value), the contact specified will act in the way described below.

| | |
|---|---|
| NO (Normally-open) contact | OFF |
| NC (Normally-closed) contact | ON |

Example

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| **TMR** | **T5 K1000** | The setting value in the timer T5 is K1000. |

Additional remark

◆ Please refer to the specifications for the model used for more information about the timer range which can be used.

| Instruction code | Operand | Function |
|---|---|---|
| CNT | $S_1$, $S_2$ | 16-bit counter |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | | | | | | | | | | | ● | | |
| $S_2$ | | | | | ● | | | | | ● | ● | | | | |

Explanation

- When the counter coil specified by the instruction CNT is turned from OFF to ON, the counter value increases by 1. If the counter value matches the setting value (counter value=setting value), the contact specified will act in the way described below.

| | |
|---|---|
| NO (Normally-open) contact | OFF |
| NC (Normally-closed) contact | ON |

- If there are pulses sent to the counter specified by the instruction CNT after the counter value matches the setting value, the state of the contact specified and the counter value will remain unchanged.

4

Example

Ladder diagram


| Instruction code: | Description: |
|---|---|
| LD X0.0 | Loading the Form A contact X0.0 |
| **CNT C20 K100** | The setting value in the counter C20 is K100. |

| Instruction code | Operand | Function |
|---|---|---|
| DCNT | $S_1$, $S_2$ | 32-bit counter |

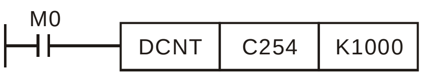
| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | | | | | | | | | | | ● | | |
| $S_2$ | | | | | ● | | | | | ● | ● | | | | |

Explanation

- DCNT is an instruction which is used to enable the 32-bit counters C200~C255.
- C200~C220 are special counters. Please refer to Chapter 10 for more information.
- C221~C2255 are general up/down counters. When the counter coil specified by the instruction DCNT is turned from OFF to ON, the counter value increases or decreases by one according to the setting of SM221~SM255.

Example

Ladder diagram




| Instruction code: | Description: |
|---|---|
| LD M0 | Loading the Form A contact M0 |
| **DCNT C254 K1000** | The setting value in the counter C254 is K1000. |

| Instruction code | Operand | Function |
|---|---|---|
| LDP | **S** | Starting rising-edge detection |

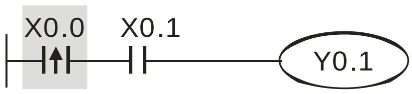
| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

**Explanation**

◆ The usage of LDP is similar to that of LD, but the action of LDP is different from that of LD. LDP reserves the present contents, and stores the state of the rising edge-triggered contact specified to an accumulation register.

**Example**

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| **LDP** | **X0.0** | Starting the detection of the state of the rising edge-triggered contact X0.0 |
| AND | X0.1 | Connecting the Form A contact X0.1 in series |
| OUT | Y0.1 | Driving the coil Y0.1 |

**Additional remark**

◆ Please refer to the specifications for the model used for more information about the operand ranges which can be used.

◆ If the state of a rising edge-triggered contact in an AH500 series motion controller is ON before the AH500 series motion controller is powered, it is TRUE after the AH500 series motion controller is powered.

| Instruction code | Operand | Function |
|---|---|---|
| LDF | **S** | Starting falling-edge detection |

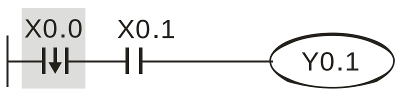
| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

**Explanation**

◆ The usage of LDF is similar to that of LD, but the action of LDP is different from that of LD. LDF reserves the present contents, and stores the state of the falling edge-triggered contact specified to an accumulation register.

**Example**

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| **LDF** | **X0.0** | Starting the detection of the state of the falling edge-triggered contact X0.0 |
| AND | X0.1 | Connecting the Form A contact X0.1 in series |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| ANDP | **S** | Connecting rising-edge detection in series |

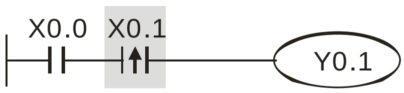
| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

Explanation

◆ The instruction ANDP is used to connect a rising edge-triggered contact in series.

Example

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| **ANDP** | **X0.1** | Connecting the rising edge-triggered contact X0.1 in series |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| ANDF | **S** | Connecting falling-edge detection in series |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

Explanation

◆ The instruction ANDF is used to connect a falling edge-triggered contact in series.

Example

Ladder diagram


| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| **ANDF** | **X0.1** | Connecting the falling edge-triggered contact X0.1 in series |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| ORP | **S** | Connecting rising-edge detection in parallel |

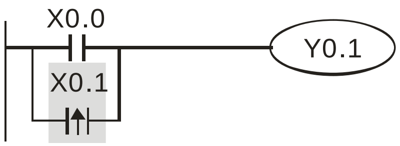
| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

Explanation

◆ The instruction ORP is used to connect a rising edge-triggered contact in parallel.

Example

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| **ORP** | **X0.1** | Connecting the rising edge-triggered contact X0.1 in parallel |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| ORF | **S** | Connecting falling-edge detection in parallel |

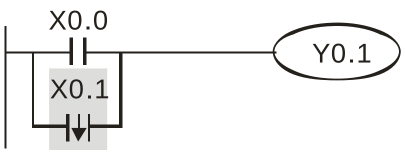
| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | ● | ● | | |

Explanation

◆ The instruction ORF is used to connect a falling edge-triggered contact in parallel.

Example

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| **ORF** | **X0.1** | Connecting the falling edge-triggered contact X0.1 |
| OUT | Y0.1 | Driving the coil Y0.1 |

| Instruction code | Operand | Function |
|---|---|---|
| PLS | **S** | Rising-edge output |

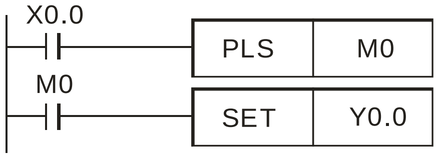
| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | ● | ● | | | | | | | | | | | | |

**Explanation**

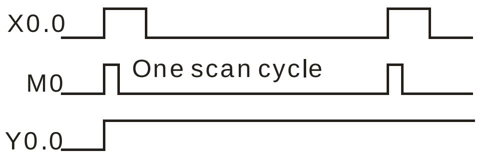
- PLS is a rising-edge output instruction. When X0.0 is turned from OFF to ON, the instruction PLS is executed. M0 sends a pulse for a scan cycle.

**Example**

Ladder diagram




Timing diagram:




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| **PLS** | **M0** | M0 is rising edge-triggered. |
| LD | M0 | Loading the Form A contact M0 |
| SET | Y0.0 | Y0.0 remains ON. |

| Instruction code | Operand | Function |
|---|---|---|
| PLF | **S** | Falling-edge output |

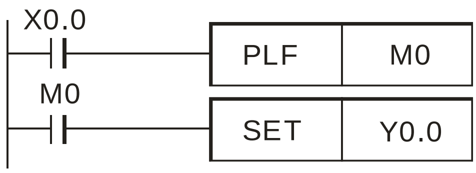
| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | ● | ● | | | | | | | | | | | | |

**Explanation**

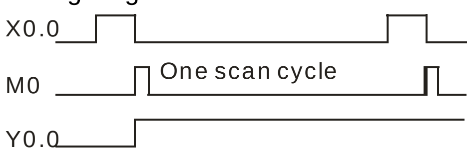
- PLF is a falling-edge output instruction. When X0.0 is turned from ON to OFF, the instruction PLF is executed. MO sends a pulse for a scan cycle.

**Example**

Ladder diagram




Timing diagram:




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| **PLF** | **M0** | M0 is falling edge-triggered. |
| LD | M0 | Loading the Form A contact M0 |
| SET | Y0.0 | Y0.0 remains ON. |

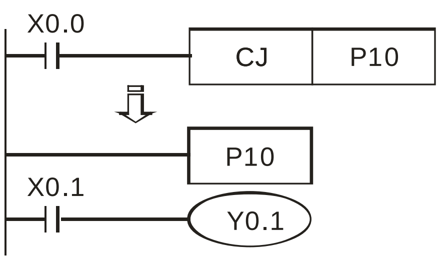
| Instruction code | Operand | Function |
|---|---|---|
| P | – | Pointer |

**Explanation**

- A pointer indicates a subroutine. It is in the range of P0 to P255.
- A pointer can be used by API 00 CJ, API 01 CALL, API 256 CJN, and API 257 JMP. The pointers used do not have to start from P0. A pointer number can not be used repeatedly, otherwise an unexpected error will occur.

**Example**

Ladder diagram




| Instruction code: | | Description: |
|---|---|---|
| LD | X0.0 | Loading the Form A contact X0.0 |
| CJ | P10 | The jump instruction CJ specifies P10. |
| : | | |
| **P10** | | Pointer P10 |
| LD | X0.1 | Loading the Form A contact X0.0 |
| OUT | Y0.1 | Driving the coil Y0.1 |

**MEMO**

# Chapter 5 Applied Instructions

## Table of Contents

## 5.1 Table of Applied Instructions

| Type | API | Instruction code | | Pulse instruction | Function | Model | | | Step | | Page number |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16-bit | 32-bit | | | 20MC | 10PM/ 15PM | 05PM | 16-bit | 32-bit | |
| Loop control | **00** | CJ | – | ✓ | Conditional jump | ✓ | ✓ | ✓ | 3 | – | 5-19 |
| | **01** | CALL | – | ✓ | Calling a subroutine | ✓ | ✓ | ✓ | 3 | – | 5-22 |
| | **02** | SRET | – | – | Indicating that a subroutine ends | ✓ | ✓ | ✓ | 1 | – | 5-23 |
| | **07** | WDT | – | ✓ | Watchdog timer | ✓ | ✓ | ✓ | 1 | – | 5-25 |
| | **08** | RPT | – | – | Start of a nested loop (only one loop) | ✓ | ✓ | ✓ | 3 | – | 5-26 |
| | **09** | RPE | – | – | End of a nested loop | ✓ | ✓ | ✓ | 1 | – | 5-27 |
| Transfer and comparison | **10** | CMP | DCMP | ✓ | Comparing values | ✓ | ✓ | ✓ | 7 | 9 | 5-28 |
| | **11** | ZCP | DZCP | ✓ | Zonal comparison | ✓ | ✓ | ✓ | 9 | 12 | 5-29 |
| | **12** | MOV | DMOV | ✓ | Transferring a value | ✓ | ✓ | ✓ | 5 | 6 | 5-30 |
| | **14** | CML | DCML | ✓ | Inverting bits | ✓ | ✓ | ✓ | 5 | 6 | 5-31 |
| | **15** | BMOV | – | ✓ | Transferring values | ✓ | ✓ | ✓ | 7 | – | 5-32 |
| | **16** | FMOV | DFMOV | ✓ | Transferring a value to several devices | ✓ | ✓ | ✓ | 7 | 8 | 5-34 |
| | **17** | XCH | DXCH | ✓ | Interchanging values | ✓ | ✓ | ✓ | 5 | 9 | 5-35 |
| | **18** | BCD | DBCD | ✓ | Converting a binary number into a binary-coded decimal number | ✓ | ✓ | ✓ | 5 | 5 | 5-36 |
| | **19** | BIN | DBIN | ✓ | Converting a binary-coded decimal number into a binary number | ✓ | ✓ | ✓ | 5 | 5 | 5-37 |
| Arithmetic | **20** | ADD | DADD | ✓ | Binary addition | ✓ | ✓ | ✓ | 7 | 9 | 5-38 |
| | **21** | SUB | DSUB | ✓ | Binary subtraction | ✓ | ✓ | ✓ | 7 | 9 | 5-40 |
| | **22** | MUL | DMUL | ✓ | Binary multiplication | ✓ | ✓ | ✓ | 7 | 9 | 5-41 |
| | **23** | DIV | DDIV | ✓ | Binary division | ✓ | ✓ | ✓ | 7 | 9 | 5-42 |

| Type | API | Instruction code 16-bit | Instruction code 32-bit | Pulse instruction | Function | Model 20MC | Model 10PM/15PM | Model 05PM | Step 16-bit | Step 32-bit | Page number |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Arithmetic | **24** | INC | DINC | ✓ | Adding one to a binary number | ✓ | ✓ | ✓ | 3 | 3 | 5-43 |
| | **25** | DEC | DDEC | ✓ | Subtracting one from a binary number | ✓ | ✓ | ✓ | 3 | 3 | 5-44 |
| | **26** | WAND | DWAND | ✓ | Logical AND operation | ✓ | ✓ | ✓ | 7 | 9 | 5-45 |
| | **27** | WOR | DWOR | ✓ | Logical OR operation | ✓ | ✓ | ✓ | 7 | 9 | 5-46 |
| | **28** | WXOR | DWXOR | ✓ | Logical exclusive OR operation | ✓ | ✓ | ✓ | 7 | 9 | 5-47 |
| | **29** | NEG | DNEG | ✓ | Taking the two's complement of a number | ✓ | ✓ | ✓ | 3 | 3 | 5-48 |
| Rotation and move | **30** | ROR | DROR | ✓ | Rotating bits rightwards | ✓ | ✓ | ✓ | 5 | 6 | 5-50 |
| | **31** | ROL | DROL | ✓ | Rotating bits leftwards | ✓ | ✓ | ✓ | 5 | 6 | 5-51 |
| | **32** | RCR | DRCR | ✓ | Rotating bits rightwards with a carry flag | ✓ | ✓ | ✓ | 5 | 6 | 5-52 |
| | **33** | RCL | DRCL | ✓ | Rotating bits leftwards with a carry flag | ✓ | ✓ | ✓ | 5 | 6 | 5-53 |
| | **34** | SFTR | – | ✓ | Moving the states of bit devices rightwards | ✓ | ✓ | ✓ | 9 | – | 5-54 |
| | **35** | SFTL | – | ✓ | Moving the states of bit devices leftwards | ✓ | ✓ | ✓ | 9 | – | 5-55 |
| | **36** | WSFR | – | ✓ | Moving the values in word devices rightwards | ✓ | ✓ | ✓ | 9 | – | 5-56 |
| | **37** | WSFL | – | ✓ | Moving the values in word devices leftwards | ✓ | ✓ | ✓ | 9 | – | 5-57 |
| | **38** | SFWR | – | ✓ | Moving a value and writing it into a word device | ✓ | ✓ | ✓ | 7 | – | 5-58 |
| | **39** | SFRD | – | ✓ | Moving a value and reading it from a word device | ✓ | ✓ | ✓ | 7 | – | 5-59 |

| Type | API | Instruction code | | Pulse instruction | Function | Model | | | Step | | Page number |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16-bit | 32-bit | | | 20MC | 10PM/ 15PM | 05PM | 16-bit | 32-bit | |
| Data | 40 | ZRST | – | ✓ | Resetting a zone | ✓ | ✓ | ✓ | 5 | – | 5-60 |
| | 41 | DECO | – | ✓ | Decoder | ✓ | ✓ | ✓ | 7 | – | 5-61 |
| | 42 | ENCO | – | ✓ | Encoder | ✓ | ✓ | ✓ | 7 | – | 5-62 |
| | 43 | SUM | DSUM | ✓ | Number of bits which are ON | ✓ | ✓ | ✓ | 5 | 5 | 5-64 |
| | 44 | BON | DBON | ✓ | Checking the state of a bit | ✓ | ✓ | ✓ | 7 | 8 | 5-65 |
| | 45 | MEAN | DMEAN | ✓ | Mean | ✓ | ✓ | ✓ | 7 | 8 | 5-66 |
| | 46 | ANS | – | – | Driving an annunciator | ✓ | ✓ | ✓ | 7 | – | 5-67 |
| | 47 | ANR | – | ✓ | Resetting an annunciator | ✓ | ✓ | ✓ | 1 | – | 5-68 |
| | 48 | SQR | DSQR | ✓ | Square root of a binary value | ✓ | ✓ | ✓ | 5 | 6 | 5-69 |
| | 49 | – | DFLT | ✓ | Converting a binary integer into a binary floating-point value | ✓ | ✓ | ✓ | – | 6 | 5-70 |
| High-speed processing | 50 | REF | – | ✓ | Refreshing the states of I/O devices | ✓ | ✓ | ✓ | 5 | – | 5-71 |
| Convenience | 61 | SER | DSER | ✓ | Searching data | ✓ | ✓ | ✓ | 9 | 11 | 5-72 |
| | 66 | ALT | – | ✓ | Alternating between ON and OFF | ✓ | ✓ | ✓ | 3 | – | 5-74 |
| I/O | 78 | FROM | DFROM | ✓ | Reading data from a control register in a special module | ✓ | ✓ | ✓ | 9 | 12 | 5-75 |
| | 79 | TO | DTO | ✓ | Writing data into a control register in a special module | ✓ | ✓ | ✓ | 9 | 13 | 5-76 |
| | 87 | ABS | DABS | ✓ | Absolute value | ✓ | ✓ | ✓ | 3 | 3 | 5-77 |

5

| Type | API | Instruction code | | Pulse instruction | Function | Model | | | Step | | Page number |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16-bit | 32-bit | | | 20MC | 10PM/ 15PM | 05PM | 16-bit | 32-bit | |
| Floating-point number | 110 | – | DECMP | ✓ | Comparing binary floating-point numbers | ✓ | ✓ | ✓ | – | 9 | 5-78 |
| | 111 | – | DEZCP | ✓ | Binary floating-point zonal comparison | ✓ | ✓ | ✓ | – | 12 | 5-79 |
| | 112 | – | DMOVR | ✓ | Transferring a floating-point value | ✓ | ✓ | ✓ | – | 6 | 5-80 |
| | 116 | – | DRAD | ✓ | Converting a degree to a radian | ✓ | ✓ | ✓ | – | 6 | 5-81 |
| | 117 | – | DDEG | ✓ | Converting a radian to a degree | ✓ | ✓ | ✓ | – | 6 | 5-82 |
| | 120 | – | DEADD | ✓ | Binary floating-point addition | ✓ | ✓ | ✓ | – | 9 | 5-83 |
| | 121 | – | DESUB | ✓ | Binary floating-point subtraction | ✓ | ✓ | ✓ | – | 9 | 5-84 |
| | 122 | – | DEMUL | ✓ | Binary floating-point multiplication | ✓ | ✓ | ✓ | – | 9 | 5-85 |
| | 123 | – | DEDIV | ✓ | Binary floating-point division | ✓ | ✓ | ✓ | – | 9 | 5-86 |
| | 124 | – | DEXP | ✓ | Exponent of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-87 |
| | 125 | – | DLN | ✓ | Natural logarithm of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-88 |
| | 126 | – | DLOG | ✓ | Logarithm of a binary floating-point number | ✓ | ✓ | ✓ | – | 9 | 5-89 |
| | 127 | – | DESQR | ✓ | Square root of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-90 |
| | 128 | – | DPOW | ✓ | Power of a floating-point number | ✓ | ✓ | ✓ | – | 9 | 5-91 |
| | 129 | – | DINT | ✓ | Converting a binary floating-point number into a binary integer | ✓ | ✓ | ✓ | – | 5 | 5-92 |

| Type | API | Instruction code | | Pulse instruction | Function | Model | | | Step | | Page number |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16-bit | 32-bit | | | 20MC | 10PM/ 15PM | 05PM | 16-bit | 32-bit | |
| Floating-point number | 130 | – | DSIN | ✓ | Sine of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-93 |
| | 131 | – | DCOS | ✓ | Cosine of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-95 |
| | 132 | – | DTAN | ✓ | Tangent of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-97 |
| | 133 | – | DASIN | ✓ | Arcsine of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-99 |
| | 134 | – | DACOS | ✓ | Arccosine of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-100 |
| | 135 | – | DATAN | ✓ | Arctangent of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-101 |
| | 136 | – | DSINH | ✓ | Hyperbolic sine of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-102 |
| | 137 | – | DCOSH | ✓ | Hyperbolic cosine of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-103 |
| | 138 | – | DTANH | ✓ | Hyperbolic tangent of a binary floating-point number | ✓ | ✓ | ✓ | – | 6 | 5-104 |
| | 172 | – | DADDR | ✓ | Floating-point addition | ✓ | ✓ | ✓ | – | 9 | 5-105 |
| | 173 | – | DSUBR | ✓ | Floating-point subtraction | ✓ | ✓ | ✓ | – | 9 | 5-106 |
| | 174 | – | DMULR | ✓ | Floating-point multiplication | ✓ | ✓ | ✓ | – | 9 | 5-107 |
| | 175 | – | DDIVR | ✓ | Floating-point division | ✓ | ✓ | ✓ | – | 9 | 5-108 |

5

| Type | API | Instruction code | | Pulse instruction | Function | Model | | | Step | | Page number |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16-bit | 32-bit | | | 20MC | 10PM/ 15PM | 05PM | 16-bit | 32-bit | |
| Logical operation | **215** | LD& | DLD& | – | S1&S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-109 |
| | **216** | LD\| | DLD\| | – | S1\|S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-109 |
| | **217** | LD^ | DLD^ | – | S1^S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-109 |
| | **218** | AND& | DAND& | – | S1&S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-110 |
| | **219** | AND\| | DAND\| | – | S1\|S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-110 |
| | **220** | AND^ | DAND^ | – | S1^S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-110 |
| | **221** | OR& | DOR& | – | S1&S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-111 |
| | **222** | OR\| | DOR\| | – | S1\|S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-111 |
| | **223** | OR^ | DOR^ | – | S1^S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-111 |
| Comparison instruction | **224** | LD= | DLD= | – | S1 = S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-112 |
| | **225** | LD> | DLD> | – | S1 > S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-112 |
| | **226** | LD< | DLD< | – | S1 < S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-112 |
| | **228** | LD<> | DLD<> | – | S1≠S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-112 |
| | **229** | LD<= | DLD<= | – | S1≦S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-112 |
| | **230** | LD>= | DLD>= | – | S1≧S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-112 |
| | **232** | AND= | DAND= | – | S1 = S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-113 |
| | **233** | AND> | DAND> | – | S1 > S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-113 |
| | **234** | AND< | DAND< | – | S1 < S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-113 |
| | **236** | AND<> | DAND<> | – | S1≠S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-113 |
| | **237** | AND<= | DAND<= | – | S1≦S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-113 |
| | **238** | AND>= | DAND>= | – | S1≧S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-113 |
| | **240** | OR= | DOR= | – | S1 = S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-114 |
| | **241** | OR> | DOR> | – | S1 > S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-114 |
| | **242** | OR< | DOR< | – | S1 < S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-114 |
| | **244** | OR<> | DOR<> | – | S1≠S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-114 |
| | **245** | OR<= | DOR<= | – | S1≦S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-114 |
| | **246** | OR>= | DOR>= | – | S1≧S2 | ✓ | ✓ | ✓ | 5 | 7 | 5-114 |
| Other instructions | **152** | SWAP | DSWAP | ✓ | Interchanging the high byte in a device with the low byte in the device | ✓ | ✓ | ✓ | 3 | 3 | 5-115 |
| | **154** | RAND | DRAND | ✓ | Random value | ✓ | ✓ | ✓ | 7 | 9 | 5-116 |
| | **202** | SCAL | – | ✓ | Scale | ✓ | ✓ | ✓ | 7 | – | 5-117 |
| | **203** | SCLP | DSCLP | ✓ | Parameter scale | ✓ | ✓ | ✓ | 7 | 9 | 5-119 |
| | **256** | CJN | – | ✓ | Negated conditional jump | ✓ | ✓ | ✓ | 3 | – | 5-123 |
| | **257** | JMP | – | – | Unconditional jump | ✓ | ✓ | ✓ | 3 | – | 5-124 |

| Type | API | Instruction code | | Pulse instruction | Function | Model | | | Step | | Page number |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16-bit | 32-bit | | | 20MC | 10PM/ 15PM | 05PM | 16-bit | 32-bit | |
| Other instructions | **258** | BRET | – | – | Returning to a busbar | ✓ | ✓ | ✓ | 1 | – | 5-125 |
| | **259** | MMOV | – | ✓ | Converting a 16-bit value into a 32-bit value | ✓ | ✓ | ✓ | 6 | – | 5-126 |
| | **260** | RMOV | – | ✓ | Converting a 32-bit value into a 16-bit value | ✓ | ✓ | ✓ | 6 | – | 5-127 |

Additional remark: 05PM=AH05PM-5A; 10PM=AH10PM-5A; 15PM=AH15PM-5A; 20MC=AH20MC-5A

## 5.2 Structure of an Applied Instruction

- An applied instruction is composed of an instruction name and operands.
  Instruction name: An instruction name represents a function.
  Operand: An operand is the object of an operation.
  An instruction name occupie one step. The number of steps an operand occupies can be two or three, depending on the instruction used is a 16-bit instruciton or a 32-bit instruction. K/16#/F used by a 32-bit instruction occupies three steps, and the other operands occupy two steps.
- Descriptions of the applied instructions
  1. A PLC instruction is assigned an instruction code and an API number. The API number in the table below is 012, and the instruction code in the table below is MOV. MOV is used to transfer data.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 12 | D | MOV | P | **S, D** | Transferring a value |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

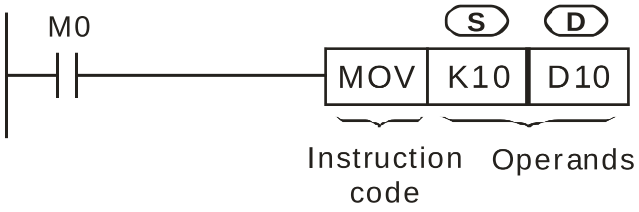
  2. The devices used by an instruction are shown in a device table. S, D, n, and m are used as operands according to their functions. If more than one operand is used, and these operands have the same function, they will be suffixed with numbers, e.g. $\mathbf{S_1}$ and $\mathbf{S_2}$.
  3. If an instruction is used as a pulse instruction, "P" will be added to the back of its instruction code. If an instruction is used as a 32-bit instruction, "D" will be added to the front of its instruction code. For example, "***" in "D***P" is an instruction code.
  4. A 32-bit floating-point number is notated by "F".
  5. The devices marked with "●" in the table above can be modified by V devices and Z devices, and the devices marked with "○" in the table above can not be modified by V devices and Z devices. For example, the D device specified by the operand **S** can be modified by a V device or a Z device.
  6. A V device can only be used by a 16-bit instruction, and a Z device can only be used by a 32-bit instruction.
  7. "✓" in the table above indicates that AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A are supported, and "–" in the table above indicates that AH05PM, AH10PM-5A AH15PM-5A, and AH20MC-5A are not supported. Users can check whether the instruction

can be used as a pulse instruction, a 16-bit instruction, and a 32-bit instruction according to the information in the table.

**Typing an applied instruction**

Some applied instructions are composed of instruction names, e.g. BRET and SRET, but most applied instructions are composed of instruction names and operands.

The applied instructions that a module can use are assigned the instruction numbers API 00~API 260. Besides, every applied instruction is assigned a mnemonic. For example, the mnemonic of API 12 is MOV. If users want to type an instruction by means of PMSoft, they can type the mnemonic assigned to the instruction. Every applied instruction specifies operands. Take the instruction MOV for instance.




The instruction is used to move the value in the operand **S** to the operand **D**.

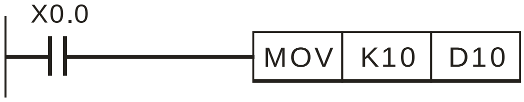
| | |
|---|---|
| **S** | Source operand<br>If there is more one source operand, the source operands will be represented by $\mathbf{S_1}$, $\mathbf{S_2}$, and etc. |
| **D** | Destination operand<br>If there is more than one destination operand, the destination operands will be represented by $\mathbf{D_1}$, $\mathbf{D_2}$, and etc. |
| If operands are constants, they will be represented by **m**, $\mathbf{m_1}$, $\mathbf{m_2}$, **n**, $\mathbf{n_1}$, $\mathbf{n_2}$, and etc. | |

5

- Length of an operand (16-bit instruction or 32-bit instruction)

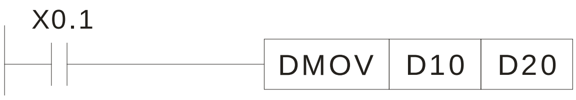
  The values in operands can be grouped into 16-bit values and 32-bit values. In order to process values of difference lengths, some applied instructions are grouped into 16-bit instructions and 32-bit instructions. After "D" is added to the front of a 16-bit instruction, the instruction becomes a 32-bit instruction.

  The instruction MOV is a 16-bit instruction.




  When X0.0 is ON, K10 is moved to D10.

  The instruction DMOV is a 32-bit instruction.

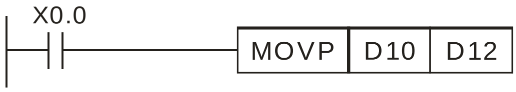



  When X0.1 is ON, the value in (D11, D10) is moved to (D21, D20).

- Continuity instruction/Pulse instruction

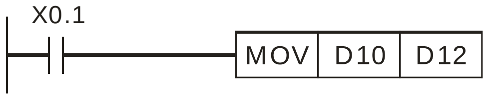
  The applied instructions can be grouped into continuity instructions and pulse instructions in terms of the ways the applied instructions are executed. If an instruction in a program is not executed, the execution of the program will take less time. As a result, if there are pulse instructions in a program, the scan cycle will be shorter. If "P" is added to the back of an instruction, the instruction becomes a pulse instruction. Some instructions are mostly used as pulse instructions.

  Pulse instruction




  When X0.0 is turned from OFF to ON, the instruction MOVP is executed once. MOVP will not be executed again during the scan cycle, and therefore it is a pulse instruction.

Continuity instruction




Whenever X0.1 is ON, the instruction MOV is executed once. MOV is a continuity instruction.

When the contacts X0.0 and X0.1 are OFF, the instructions are not executed, and the values in the destation operands are not changed.

◆ Operand

1. A word device can consist of bit devices. Applied instructions can use KnM and KnS. Values can be stored in KnM and KnS.
2. Data registers, timers, counters, and index registers can be used as general operands.
3. A data register is a 16-bit register. If users want to use a 32-bit data register, they have to specify two consecutive data registers.
4. If a 32-bit instruction uses D0 as an operand, the 32-bit data register composed of D1 and D0 will be used. D1 occupies the high 16 bits, and D0 occupy the low 16 bits. Timers and the 16-bit counters C0~C199 can be used in the same way.
5. If the 32-bit counters C240~C255 are used as data registers, they can be operands used by 32-bit instructions.

◆ SM/SR devices are like M/D devices.

| **Device** | Xn.n | Yn.n | **M** | S | K | 16# | F | KnM | KnS | **D** | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

Owing to the fact that SM/SR devices are like M/D devices, users can refer to **M**/**D** columns.

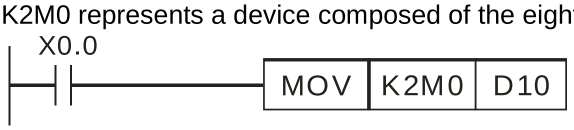
◆ X/Y devices

| **Device** | **Xn.n** | **Yn.n** | M | S | K | 16# | F | KnM | KnS | **D** | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

Xn.n (X0.0~X15.15) and Yn.n (Y0.0~Y15.15) are bit devices.
If an instruction supports D devices, it will also support Xn and Yn. The usage of Xn/Yn is the same as the usage of D devices.

◆ Operand type

1. X devices, Y devices, M devices, and S devices can only be turned ON or OFF. They are bit devices.
2. 16-bit (or 32-bit) T devices, C device, D devices, V devices, and Z devices are word devices.
3. If Kn is added to the front of an M/S device, a word device will be formed. For example, K2M0 represents a device composed of the eight bit devices M0~M7.




When X0.0 is ON, the values of M0~M7 are moved to bit 0~bit 7 in D10, and bit 8~bit 15 are set to 0.

- Values in word device composed of bit devices

| 16-bit instruction | |
|---|---|
| A 16-bit value is in the range of K-32,768 to K32,767. | |
| Value in a word device composed of bit devices | |
| K1 (4 bits) | 0~15 |
| K2 (8 bits) | 0~255 |
| K3 (12 bits) | 0~4,095 |
| K4 (16 bits) | -32,768~+32,767 |

| 32-bit instruction | |
|---|---|
| A 32-bit value is in the range of K-2,147,483,648 to K2,147,483,647. | |
| Value in a word device composed of bit devices | |
| K1 (4 bits) | 0~15 |
| K2 (8 bits) | 0~255 |
| K3 (12 bits) | 0~4,095 |
| K4 (16 bits) | 0~65,535 |
| K5 (20 bits) | 0~1,048,575 |
| K6 (24 bits) | 0~167,772,165 |
| K7 (28 bits) | 0~268,435,455 |
| K8 (32 bits) | -2,147,483,648~+2,147,483,647 |

- General flags

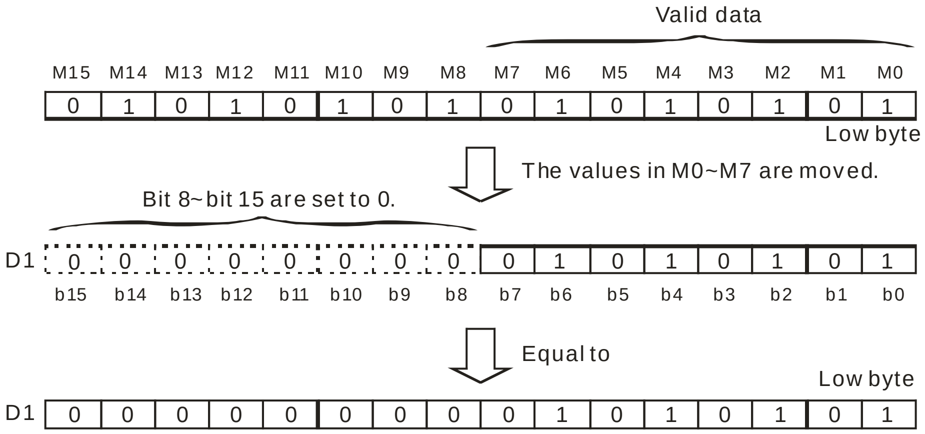
  Every flag in a module corresponds to an operation result.

  Example: SM968 is a zero flag, SM969 is a borrow flag, and SM970 is a carry flag

  The state of a flag varies with an operation result. For example, if the instruction ADD/SUB/MUL/DIV is used in the main program O100~M102, the operation result gotten will affect the states of SM968~SM970. However, if the instruciton is not executed, the states of the flags will remain unchanged. The states of flags are related to instructions. Please refer to the explanations of instructions for more information.

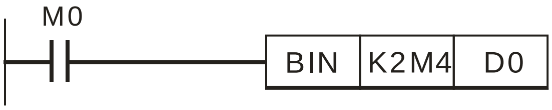
## 5.3 Processing Values

- X devices, Y devices, M devices, and S devices can only be turned ON or OFF. They are bit devices. Values can be stored in T device, C devices, D devices, V devices, and Z devices. They are word devices. If Kn is added to the front of an M/S device, a word device will be formed.
- If Kn is added to the front of an M/S device, a word device will be formed. For example, K2M0 represents a device composed of the eight bit devices M0~M7.




- The value in K1M0 is moved to a 16-bit register, and bit 4~bit 15 in the register are set to 0. The value in K2M0 is moved to a 16-bit register, and bit 8~bit 15 in the register are set to 0. The value in K3M0 is moved to a 16-bit register, and bit 12~bit 15 in the register are set to 0. The value in K1M0 is moved to a 32-bit register, and bit 4~bit 31 in the register are set to 0. The value in K2M0 is moved to a 32-bit register, and bit 8~bit 31 in the register are set to 0. The value in K3M0 is moved to a 32-bit register, and bit 12~bit 31 in the register are set to 0. The value in K4M0 is moved to a 32-bit register, and bit 16~bit 31 in the register are set to 0. The value in K5M0 is moved to a 32-bit regiser, and bit 20~bit 31 in the register are set to 0. The vlaue in K6M0 is moved to a 32-bit register, and bit 24~bit 31 in the register are set to 0. The value in K7M0 is moved to a 32-bit register, and bit 28~bit 31 in the register are set to 0.

- If Kn is in the range of K1~K3 (or K4~K7), the bits which are not assigned values in the 16-bit register (the 32-bit register) to which a value is moved will be set to 0. As a result, operations will be performed on positive numbers if Kn is in the range of K1~K3 (or K4~K7).




☞ The binary-coded decimal number in M4~M11 is converted into a binary number, and the binary number is stored in D0.

- Users can specify bit device numbers freely. It is suggested that M device numbers/S device numbers should start from a number which is a multiple of 8.
- Consecutive devices

  Take data registers for instances. D0, D1, D2, D3, and D4 are consecutive data registers.

  The consecutive word devices composed of bit devices are shown below.

| | | | |
|---|---|---|---|
| K1M0 | K1M4 | K1M8 | K1M12 |
| K2M0 | K2M8 | K2M16 | K2M24 |
| K3M0 | K3M12 | K3M24 | K3M36 |
| K4M0 | K4M16 | K4M32 | K4M48 |

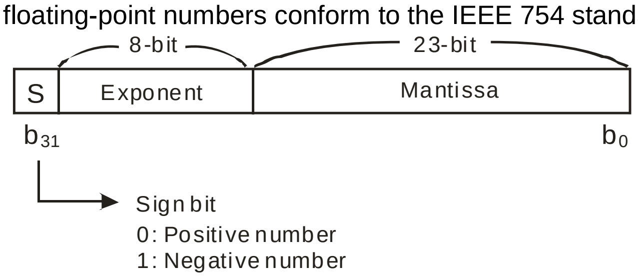
  The consecutive word devices composed of bit devices are shown above. To avoid confusion, please do not skip any word device composed of bit devices. Beisdes, if a 32-bit operation is performed on K4M0, the high 16 bits in the 32-bit register to which the value in K4M0 is moved will be set to 0. If a 32-bit value is required, please use K8M0.

After an operation is performed, the binary integer gotten will be given priority. For example, 40÷3=13, and the remainder 1 is dropped. The integer part of the square root of an integer is retained, and the fractional part of the square root is dropped. However, if a decimal instruiction is used, a decimal will be gotten. The applied intructions listed below are decimal instructions.

| | | | |
|---|---|---|---|
| API 110 (D ECMP) | API 111 (D EZCP) | API 116 (D RAD) | API 117 (D DEG) |
| API 120 (D EADD) | API 121 (D ESUB) | API 122 (D EMUL) | API 123 (D EDIV) |
| API 124 (D EXP) | API 125 (D LN) | API 126 (D LOG) | API 127 (D ESQR) |
| API 128 (D POW) | API 129 (D INT) | API 130 (D SIN) | API 131 (D COS) |
| API 132 (D TAN) | API 133 (D ASIN) | API 134 (D ACOS) | API 135 (D ATAN) |
| API 136 (D SINH) | API 137 (D COSH) | API 138 (D TANH) | |

**Representations of binary floating-point numbers**

The floating-point numbers in a motion control module are 32-bit floating-point numbers, and the representations of the floating-point numbers conform to the IEEE 754 standard.




Representation of a floating-point number:

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

A 32-bit floating-point number is in the range of $\pm 2^{-126}$ to $\pm 2^{+128}$, that is, a 32-bit floating-point number is in the range of $\pm 1.1755\times 10^{-38}$ to $\pm 3.4028\times 10^{+38}$.

Example 1: 23 is represented by a 32-bit floating-point number.

Step 1: Converting 23 into a binary number: 23.0=10111

5

Step 2: Normalizing the binary number: $10111=1.0111\times2^4$, (0111 is a mantissa, and 4 is an exponent)

Step 3: Getting the exponent which is stored

$\because E-B=4 \rightarrow E-127=4 \therefore E=131=10000011_2$

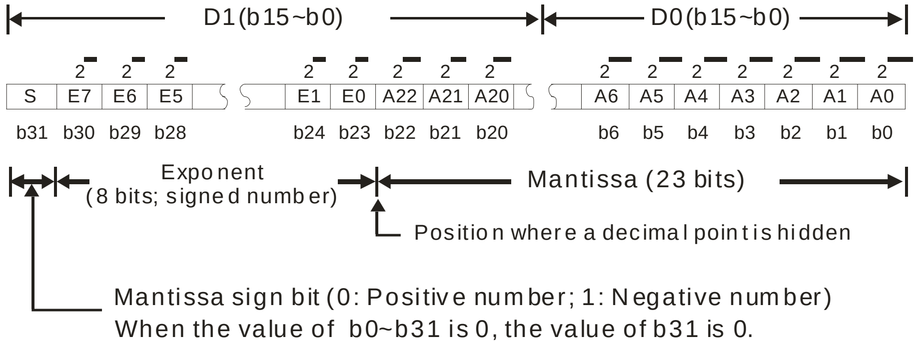
Step 4: Combining the sign bit, the exponent, and the mantissa to form a floating-point number.

0 100**00**11 011**1000**0000**0000**0000**0000**$_2$=41B80000$_{16}$

Example 2: -23.0 is represented by a 32-bit floating-point number.

-23.1 is converted in the same way as 23.0. Users only need to change the sign bit to 1.

A motion control module uses two consecutive registers to form a 32-bit floating-point numbers. Take (D1, D0) in which a bianry floating-point number is stored for instance.




**Decimal floating-point number**

- Since binary floating-point numbers are not widely accepted by people, they can be converted into decimal floating-point numbers. However, the decimals on which operations are performed in a motion control module are still binary floating-point numbers.
- A decimal floating-point number is stored in two consecutive registers. The constant part is stored in the register whose device number is smaller, and the exponent part is stored in the register whose device number is bigger.

  Take (D1, D0) for instance.

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

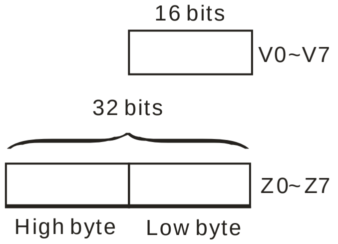
  Base: D0=±1,000~±9,999

  Exponent: D1=-41~+35

  Besides, the base 100 does not exist in D0 because 100 is represented by $1,000\times10^{-1}$. A decimal floating-point number is in the range of $\pm1,175\times10^{-41}$ to $\pm3,402\times10^{+35}$.
- If the instruction ADD/SUB/MUL/DIV is used in the main program O100~M102, the operation result gotten will affect the states of SM968~SM970. If a floating-point operation instruction is used, the result gotten will also affect the state of the zero flag SM968, the state of the borrow flag SM969, and the state of the carry flag SM970.
  - Zero flag: If the operation result gotten is 0, SM968 will be ON.
  - Borrow flag: If the absolute value of the operation result gotten is less than the minimum value allowed, SM969 will be ON.
  - Carry flag: If the absolute value of the operaiton result gotten is greater than the maximum value allowed, SM970 will be ON.

## 5.4 Using Index Registers to Modify Operands

V devices are 16-bit index registers. There are 6 V devices (V0~V5). Z devices are 32-bit index registers. There are 8 Z devices (Z0~Z7).




V devices are 16-bit registers. Data can be freely written into a V device, and data can be freely read from a V device. If a 32-bit value is required, please use a Z device.

Index registers can be used to modify X/Y/M/S/KnM/KnS/T/C/D/SM/SR devices, but they can not be used to modify index registers, constants, and Kn. For example, K4@Z0 is invalid, K4M0@Z0 is valid, and K0@Z0M0 is invalid. The devices marked with "●" in the table in the explanation of an applied instruction can be modified by V devices and Z devices.

## 5.5 Instruction Index

- Arranging applied instructions in alphabetical order

| Type | API | Instruction code | | Pulse instruction | Function | Step | | Page number |
|---|---|---|---|---|---|---|---|---|
| | | 16-bit | 32-bit | | | 16-bit | 32-bit | |
| A | 87 | ABS | DABS | ✓ | Absolute value | 3 | 3 | 5-77 |
| | 20 | ADD | DADD | ✓ | Binary addition | 7 | 9 | 5-38 |
| | 134 | – | DACOS | ✓ | Arccosine of a binary floating-point number | – | 6 | 5-100 |
| | 172 | – | DADDR | ✓ | Floating-point addition | – | 9 | 5-105 |
| | 66 | ALT | – | ✓ | Alternating between ON and OFF | 3 | – | 5-74 |
| | 218 | AND& | DAND& | – | S1&S2 | 5 | 7 | 5-110 |
| | 220 | AND^ | DAND^ | – | S1^S2 | 5 | 7 | 5-110 |
| | 219 | AND\| | DAND\| | – | S1\|S2 | 5 | 7 | 5-110 |
| | 234 | AND< | DAND< | – | S1 < S2 | 5 | 7 | 5-113 |
| | 237 | AND<= | DAND<= | – | S1≦S2 | 5 | 7 | 5-113 |
| | 236 | AND<> | DAND<> | – | S1≠S2 | 5 | 7 | 5-113 |
| | 232 | AND= | DAND= | – | S1 = S2 | 5 | 7 | 5-113 |
| | 233 | AND> | DAND> | – | S1 > S2 | 5 | 7 | 5-113 |
| | 238 | AND>= | DAND>= | – | S1≧S2 | 5 | 7 | 5-113 |
| | 47 | ANR | – | ✓ | Resetting an annunciator | 1 | – | 5-68 |
| | 46 | ANS | – | – | Driving an annunciator | 7 | – | 5-67 |
| | 133 | – | DASIN | ✓ | Arcsine of a binary floating-point number | – | 6 | 5-99 |
| | 135 | – | DATAN | ✓ | Arctangent of a binary floating-point number | – | 6 | 5-101 |
| B | 18 | BCD | DBCD | ✓ | Converting a binary number into a binary-coded decimal number | 5 | 5 | 5-36 |
| | 19 | BIN | DBIN | ✓ | Converting a binary-coded decimal number into a binary number | 5 | 5 | 5-37 |
| | 15 | BMOV | – | ✓ | Transferring values | 7 | – | 5-32 |
| | 44 | BON | DBON | ✓ | Checking the state of a bit | 7 | 8 | 5-65 |
| | 258 | BRET | – | – | Returning to a busbar | 1 | – | 5-125 |
| C | 01 | CALL | – | ✓ | Calling a subroutine | 3 | – | 5-22 |

| Type | API | Instruction code 16-bit | Instruction code 32-bit | Pulse instruction | Function | Step 16-bit | Step 32-bit | Page number |
|---|---|---|---|---|---|---|---|---|
| C | 131 | – | DCOS | ✓ | Cosine of a binary floating-point number | – | 6 | 5-95 |
| | 137 | – | DCOSH | ✓ | Hyperbolic cosine of a binary floating-point number | – | 6 | 5-103 |
| | 00 | CJ | – | ✓ | Conditional jump | 3 | – | 5-19 |
| | 256 | CJN | – | ✓ | Negated conditional jump | 3 | – | 5-123 |
| | 14 | CML | DCML | ✓ | Inverting bits | 5 | 6 | 5-31 |
| | 10 | CMP | DCMP | ✓ | Comparing values | 7 | 9 | 5-28 |
| D | 25 | DEC | DDEC | ✓ | Subtracting one from a binary number | 3 | 3 | 5-44 |
| | 41 | DECO | – | ✓ | Decoder | 7 | – | 5-61 |
| | 117 | – | DDEG | ✓ | Converting a radian to a degree | – | 6 | 5-82 |
| | 23 | DIV | DDIV | ✓ | Binary division | 7 | 9 | 5-42 |
| | 175 | – | DDIVR | ✓ | Floating-point division | – | 9 | 5-108 |
| | 120 | – | DEADD | ✓ | Binary floating-point addition | – | 9 | 5-83 |
| | 110 | – | DECMP | ✓ | Comparing binary floating-point numbers | – | 9 | 5-78 |
| | 123 | – | DEDIV | ✓ | Binary floating-point division | – | 9 | 5-86 |
| | 122 | – | DEMUL | ✓ | Binary floating-point multiplication | – | 9 | 5-85 |
| E | 42 | ENCO | – | ✓ | Encoder | 7 | – | 5-62 |
| | 127 | – | DESQR | ✓ | Square root of a binary floating-point number | – | 6 | 5-90 |
| | 121 | – | DESUB | ✓ | Binary floating-point subtraction | – | 9 | 5-84 |
| | 124 | – | DEXP | ✓ | Exponent of a binary floating-point number | – | 6 | 5-87 |
| | 111 | – | DEZCP | ✓ | Binary floating-point zonal comparison | – | 12 | 5-79 |
| F | 49 | – | DFLT | ✓ | Converting a binary integer into a binary floating-point value | – | 6 | 5-70 |
| | 16 | FMOV | DFMOV | ✓ | Transferring a value to several devices | 7 | 8 | 5-34 |
| | 78 | FROM | DFROM | ✓ | Reading data from a control register in a special module | 9 | 12 | 5-75 |
| I | 24 | INC | DINC | ✓ | Adding one to a binary number | 3 | 3 | 5-43 |
| | 129 | – | DINT | ✓ | Converting a binary floating-point number into a binary integer | – | 5 | 5-92 |
| J | 257 | JMP | – | – | Unconditional jump | 3 | – | 5-124 |
| L | 215 | LD& | DLD& | – | S1&S2 | 5 | 7 | 5-109 |
| | 217 | LD^ | DLD^ | – | S1^S2 | 5 | 7 | 5-109 |
| | 216 | LD\| | DLD\| | – | S1\|S2 | 5 | 7 | 5-109 |
| | 226 | LD< | DLD< | – | S1 < S2 | 5 | 7 | 5-112 |
| | 229 | LD<= | DLD<= | – | S1≦S2 | 5 | 7 | 5-112 |
| | 228 | LD<> | DLD<> | – | S1≠S2 | 5 | 7 | 5-112 |
| | 224 | LD= | DLD= | – | S1 = S2 | 5 | 7 | 5-112 |
| | 225 | LD> | DLD> | – | S1 > S2 | 5 | 7 | 5-112 |
| | 230 | LD>= | DLD>= | – | S1≧S2 | 5 | 7 | 5-112 |
| | 125 | – | DLN | ✓ | Natural logarithm of a binary floating-point number | – | 6 | 5-88 |

| Type | API | Instruction code | | Pulse instruction | Function | Step | | Page number |
|---|---|---|---|---|---|---|---|---|
| | | 16-bit | 32-bit | | | 16-bit | 32-bit | |
| L | 126 | – | DLOG | ✓ | Logarithm of a binary floating-point number | – | 9 | 5-89 |
| M | 45 | MEAN | DMEAN | ✓ | Mean | 7 | 8 | 5-66 |
| | 259 | MMOV | – | ✓ | Converting a 16-bit value into a 32-bit value | 6 | – | 5-126 |
| | 12 | MOV | DMOV | ✓ | Transferring a value | 5 | 6 | 5-30 |
| | 112 | – | DMOVR | ✓ | Transferring a floating-point value | – | 6 | 5-80 |
| | 22 | MUL | DMUL | ✓ | Binary multiplication | 7 | 9 | 5-41 |
| | 174 | – | DMULR | ✓ | Floating-point multiplication | – | 9 | 5-107 |
| N | 29 | NEG | DNEG | ✓ | Taking the two's complement of a number | 3 | 3 | 5-48 |
| O | 221 | OR& | DOR& | – | S1&S2 | 5 | 7 | 5-111 |
| | 223 | OR^ | DOR^ | – | S1^S2 | 5 | 7 | 5-111 |
| | 222 | OR\| | DOR\| | – | S1\|S2 | 5 | 7 | 5-111 |
| | 242 | OR< | DOR< | – | S1 < S2 | 5 | 7 | 5-114 |
| | 245 | OR<= | DOR<= | – | S1≦S2 | 5 | 7 | 5-114 |
| | 244 | OR<> | DOR<> | – | S1≠S2 | 5 | 7 | 5-114 |
| | 240 | OR= | DOR= | – | S1 = S2 | 5 | 7 | 5-114 |
| | 241 | OR> | DOR> | – | S1 > S2 | 5 | 7 | 5-114 |
| | 246 | OR>= | DOR>= | – | S1≧S2 | 5 | 7 | 5-114 |
| P | 128 | – | DPOW | ✓ | Power of a floating-point number | – | 9 | 5-91 |
| R | 116 | – | DRAD | ✓ | Converting a degree to a radian | – | 6 | 5-81 |
| | 154 | RAND | DRAND | ✓ | Random value | 7 | 9 | 5-116 |
| | 33 | RCL | DRCL | ✓ | Rotating bits leftwards with a carry flag | 5 | 6 | 5-53 |
| | 32 | RCR | DRCR | ✓ | Rotating bits rightward with a carry flag | 5 | 6 | 5-52 |
| | 50 | REF | – | ✓ | Refreshing the states of I/O devices | 5 | – | 5-71 |
| | 260 | RMOV | – | ✓ | Converting a 32-bit value into a 16-bit value | 6 | – | 5-127 |
| | 31 | ROL | DROL | ✓ | Rotating bits leftwards | 5 | 6 | 5-51 |
| | 30 | ROR | DROR | ✓ | Rotating bits rightwards | 5 | 6 | 5-50 |
| | 09 | RPE | – | – | End of a nested loop | 1 | – | 5-27 |
| | 08 | RPT | – | – | Start of a nested loop (only one loop) | 3 | – | 5-26 |
| S | 202 | SCAL | – | ✓ | Scale | 7 | – | 5-117 |
| | 203 | SCLP | DSCLP | ✓ | Parameter scale | 7 | 9 | 5-119 |
| | 61 | SER | DSER | ✓ | Searching data | 9 | 11 | 5-72 |
| | 39 | SFRD | – | ✓ | Moving a value and reading it from a word device | 7 | – | 5-59 |
| | 35 | SFTL | – | ✓ | Moving the states of bit devices leftwards | 9 | – | 5-55 |
| | 34 | SFTR | – | ✓ | Moving the states of bit devices rightwards | 9 | – | 5-54 |
| | 38 | SFWR | – | ✓ | Moving a value and writing it into a word device | 7 | – | 5-58 |
| | 130 | – | DSIN | ✓ | Sine of a binary floating-point number | – | 6 | 5-93 |

| Type | API | Instruction code | | Pulse instruction | Function | Step | | Page number |
|---|---|---|---|---|---|---|---|---|
| | | 16-bit | 32-bit | | | 16-bit | 32-bit | |
| S | **136** | – | DSINH | ✓ | Hyperbolic sine of a binary floating-point number | – | 6 | 5-102 |
| | **173** | – | DSUBR | ✓ | Floating-point subtraction | – | 9 | 5-106 |
| | **48** | SQR | DSQR | ✓ | Square root of a binary value | 5 | 6 | 5-69 |
| | **2** | SRET | – | – | Indicating that a subroutine ends | 1 | – | 5-23 |
| | **21** | SUB | DSUB | ✓ | Binary subtraction | 7 | 9 | 5-40 |
| | **43** | SUM | DSUM | ✓ | Number of bits which are ON | 5 | 5 | 5-64 |
| | **152** | SWAP | DSWAP | ✓ | Interchanging the high byte in a device with the low byte in the device | 3 | 3 | 5-115 |
| T | **132** | – | DTAN | ✓ | Tangent of a binary floating-point number | – | 6 | 5-97 |
| | **138** | – | DTANH | ✓ | Hyperbolic tangent of a binary floating-point number | – | 6 | 5-104 |
| | **79** | TO | DTO | ✓ | Writing data into a control register in a special module | 9 | 13 | 5-76 |
| W | **26** | WAND | DWAND | ✓ | Logical AND operation | 7 | 9 | 5-45 |
| | **07** | WDT | – | ✓ | Watchdog timer | 1 | – | 5-25 |
| | **27** | WOR | DWOR | ✓ | Logical OR operation | 7 | 9 | 5-46 |
| | **37** | WSFL | – | ✓ | Moving the values in word devices leftwards | 9 | – | 5-57 |
| | **36** | WSFR | – | ✓ | Moving the values in word devices rightwards | 9 | – | 5-56 |
| | **28** | WXOR | DWXOR | ✓ | Logical exclusive OR operation | 7 | 9 | 5-47 |
| X | **17** | XCH | DXCH | ✓ | Interchanging values | 5 | 9 | 5-35 |
| Z | **11** | ZCP | DZCP | ✓ | Zonal comparison | 9 | 12 | 5-29 |
| | **40** | ZRST | – | ✓ | Resetting a zone | 5 | – | 5-60 |

## 5.6 Descriptions of the Applied Instructions

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 00 | | CJ | P | **S** | Conditional jump |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | | | | | | | | |

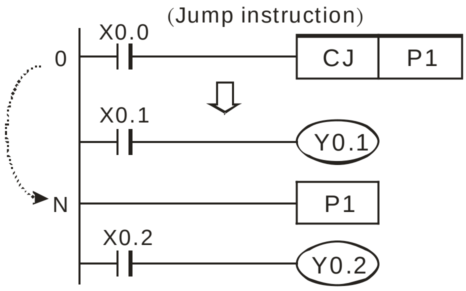
| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

Explanation

- **S**: Pointer which points to a jump destination
- A pointer is in the range of P0~P255.
- If some part of the main program O100 does not need to be executed, users can use CJ or CJP to shorten the scan time. Besides, if a dual output is used, users can use CJ or CJP.
- If the program specified by a pointer is prior to the instruction CJ, a watchdog timer error will occur, and the main program will not be executed. Please use the instruction carefully.
- The instruction CJ can specify the same pointer repeatedly. The pointer specified by CJ can not be the same as the pointer specified by CALL, otherwise an error will occur.
- When the instruction CJ/CJP in a program is executed, the actions of the devices in the program are as follows.
  - The states of the Y devices, the states of the M devices, and the states of the S devices in the program remain the same as those before the execution of the jump.
  - The 10 millisecond timers in the program stop counting.
  - The general counters in the program stop counting, and the general applied instructions in the program are not executed.
  - If the instructions which are used to reset the timers in the program are driven before the jump is executed, the timers will still be reset during the execution of the jump.

Example 1

- When X0.0 is ON, the execution of the program jumps from address 0 to address N (P1).
- When X0.0 is OFF, the execution of the program starts from address 0, and the instruction CJ is not executed.



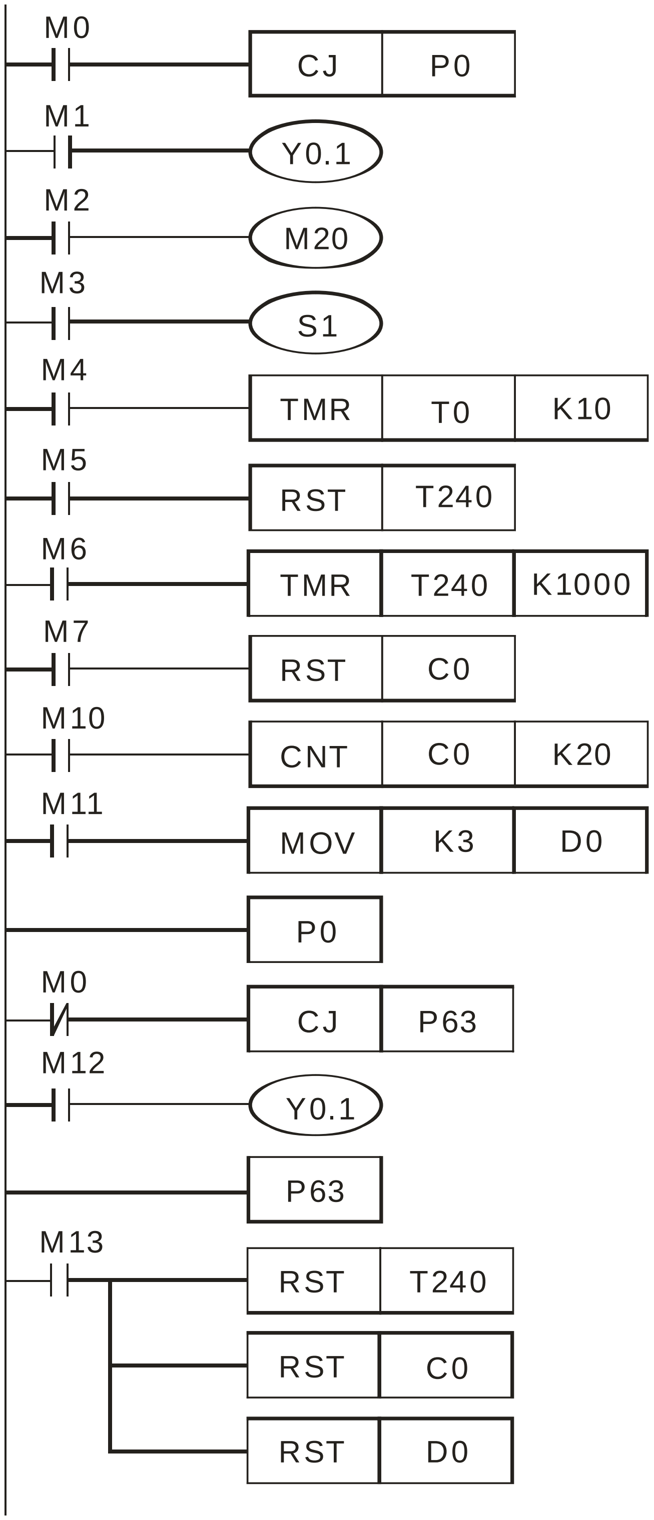
Example 2

◆ States of devices

| Device | States of contacts before the execution of CJ | States of contacts during the execution of CJ | States of output coils during the execution of CJ |
|---|---|---|---|
| Y devices, M devices, S devices | M1, M2, and M3 are OFF. | M1, M2, and M3 are turned from OFF to ON. | Y0.1[*1], M20, and S1 are OFF. |
| | M1, M2, and M3 are ON. | M1, M2, and M3 are turned from ON to OFF. | Y0.1[*1], M20, and S1 are ON. |
| 10 millisecond timers | M4 is OFF. | M4 is turned from OFF to ON. | The timer T0 does not count. |
| | M4 is ON. | M4 is turned from ON to OFF. | The timer T0 stops counting immediately. When M0 is turned from ON to OFF, the timer T0 is reset to 0. |
| | M6 is OFF. | M6 is turned from OFF to ON. | The timer T240 does not count. |
| | M6 is ON. | M6 is turned from ON to OFF. | The timer T240 stops counting immediately. When M0 is turned from ON to OFF, the timer T240 is reset to 0. |
| C0~C234 | M7 and M10 are OFF. | M10 is ON/OFF. | The counter C0 does not count. |
| | M7 is OFF. M10 is ON/OFF. | M10 is ON/OFF. | C0 stops counting. After M0 is turned OFF, C0 will resume counting. |
| Applied instructions | M11 OFF | M11 is turned from OFF to ON. | The applied instructions are not executed. |
| | M11 ON | M11 is turned from ON to OFF. | The applied instructions which are skipped are not executed, but API 53~API 59 and API 157~API 159 are still 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.

- 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.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 01 | | CALL | P | **S** | Calling a subroutine |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | | | | | | | | |

| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

Explanation

- **S**: Pointer which points to a subroutine
- A pointer is in the range of P0~P255.
- The subroutine to which a pointer points should be written after M102, M2 and the instruction SRET.
- The pointer used by the instruction CALL can not be the same as the pointers used by the instructions CJ, CJN, and JMP.
- If only the instruction CALL is used, the same subroutine can be called repeatedly.

| API | Instruction code | Operand | Function |
|---|---|---|---|
| 02 | SRET | – | Indicating that a subroutine ends |

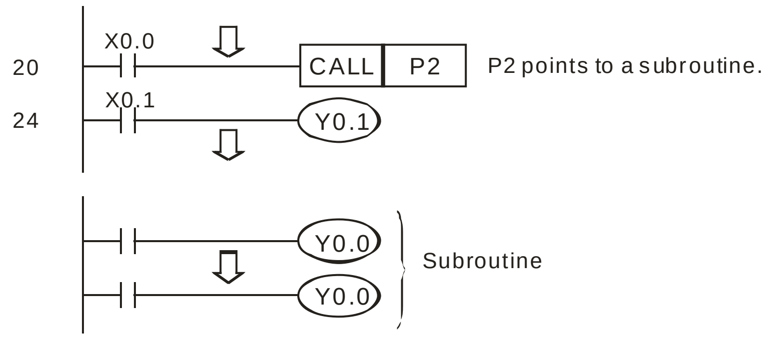
| Pulse instruction | 16-bit instruction (1 step) | 32-bit instruction |
|---|---|---|
| – | ✓ | – |

**Explanation**

- The instruction SRET indicates that a soubroutine ends. After the execution of a subroutine in a program is complete, the instruction following CALL which calls the subroutine in the main program O100 will be executed.

**Example 1**

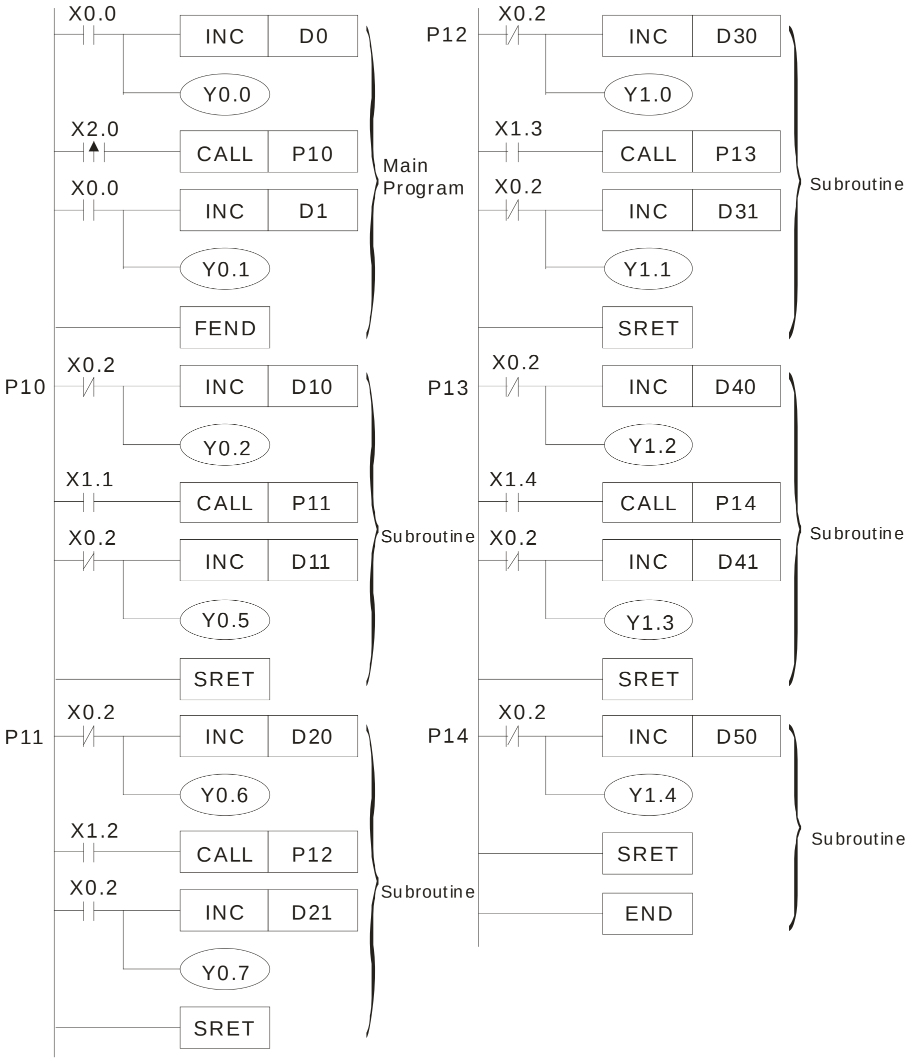
- When X0.0 is ON, the instruction CALL is executed, and the execution of the program jumps to the subroutine to which P2 points. When the instruction SRET is executed, the execution of the program returns to address 24.




**Example 2**

- When X2.0 is turned from OFF to ON, the instruction CALL P10 is executed, and the execution of the program jumps to the subroutine to which P10 points.
- When X1.1 is ON, the instruction CALL P11 is executed, and the execution of the program jumps to the subroutine to which P11 points.
- When X1.2 is ON, the instruction CALL P12 is executed, and the execution of the program jumps to the subroutine to which P12 points.
- When X1.3 is ON, the instruction CALL P13 is executed, and the execution of the program jumps to the subroutine to which P13 points.
- When X1.4 is ON, the instruction CALL P14 is executed, and the execution of the program jumps to the subroutine to which P14 points. When the instruction SRET is executed, the execution of the program returns to the previous subroutine.
- When the instruction SRET in the subroutine to which P10 points is executed, the execution of the program returns to the main program.

5

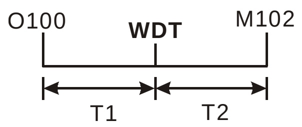

X0.0
INC D0
Y0.0
X2.0
CALL P10
X0.0
INC D1
Y0.1
FEND
Main Program
P10
X0.2
INC D10
Y0.2
X1.1
CALL P11
X0.2
INC D11
Y0.5
SRET
Subroutine
P11
X0.2
INC D20
Y0.6
X1.2
CALL P12
X0.2
INC D21
Y0.7
SRET
Subroutine
P12
X0.2
INC D30
Y1.0
X1.3
CALL P13
X0.2
INC D31
Y1.1
SRET
Subroutine
P13
X0.2
INC D40
Y1.2
X1.4
CALL P14
X0.2
INC D41
Y1.3
SRET
Subroutine
P14
X0.2
INC D50
Y1.4
SRET
END
Subroutine

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 07 | | WDT | P | – | Watchdog timer |

| Pulse instruction | 16-bit instruction (1 step) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

**Explanation**

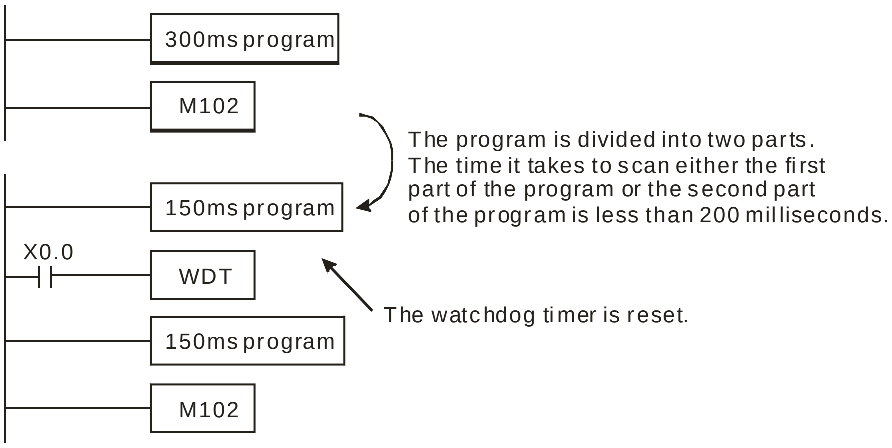
- The instruction WDT is used to reset the watchdog timer in a motion control module. If the scan time in a motion control module exceeds 200 milliseconds, the error LED indicator of the motion control module will be ON, and users will have to disconnect the motion control module. After the users connect the motion control module again, the module control module will judge its state according to the setting of the STOP/RUN switch. If there is no STOP/RUN switch, the module will stop running automatically.
- The points when a watchdog timer acts are as follows.
  - The system is abnormal.
  - The execution of a program takes much time, and therefore the scan time is greater than the setting value in SR0. There are two ways users can use to improve the situation.
    1. Using the instruction WDT




    2. Changing the value in SR0 (The default setting is 200 milliseconds.)

**Example**

- Suppose the scan 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 will be less than 200 milliseconds.




**Additional remark**

- The instruction WDT is executed when a condition is met. Users can make the instruction WDT executed only in one scan cycle by writing a program. They can use the pulse instruction WDTP.
- The default setting of a watchdog timer is 200 milliseconds. Users can set a watchdog timer by means of SR0.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 08 | | RPT | | S | Start of a nested loop |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | |

| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction |
|---|---|---|
| – | ✓ | – |

Explanation

- **S**: Number of times a loop is executed
- There is only one RPT-RPE loop in a program. If there is more than one RPT-RPE loop in a program, an error will occur.

5

| PI | Instruction code | Operand | Function |
|---|---|---|---|
| 09 | RPE | – | End of a nested loop |

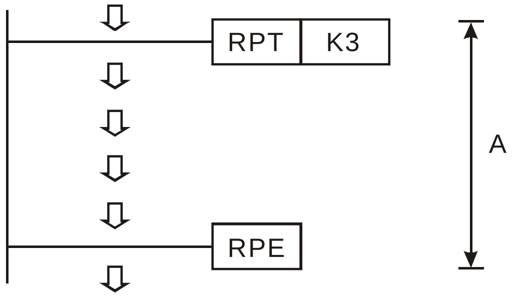
| Pulse instruction | 16-bit instruction (1step) | 32-bit instruction |
|---|---|---|
| – | ✓ | – |

**Explanation**

- RPT in a program specifies that the RPT-RPE loop in the program must be executed N times.
- N is in the range of K1 to K32,767. If N≦K1, N will be regarded as K1.
- Users can skip the execution of the RPT-RPE loop in a program by means of the instruction CJ.
- An error will occur if
  - the instruction RPE is before the instruction RPT
  - there is RPT, but there is no RPE
  - the number of times RPT is used is not the same as the number of times RPE is used
- There is only one RPT-RPE loop in a program. If there is more than one RPT-RPE loop in a program, an error will occur.

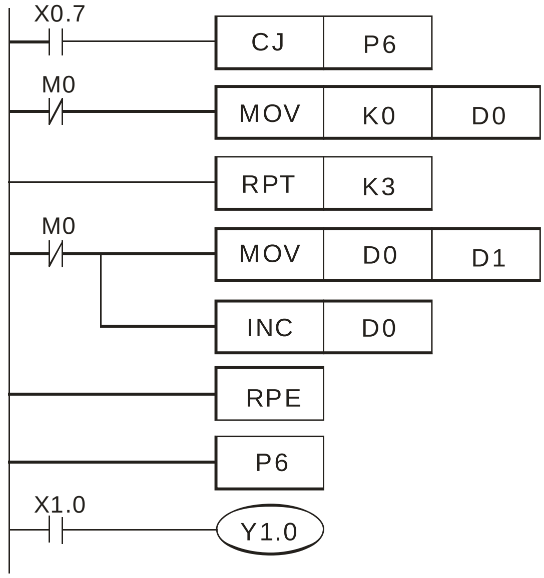
**Example 1**

- Part A can be executed three times by means of a RPT-RPE loop.




**Example 2**

- When X0.7 is OFF, the program between RPT and RPE is executed. When X0.7 is ON, the instruction CJ is executed, the subroutine to which P6 points is executed, and the program between RPT and RPE is skipped.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 10 | D | CMP | P | **$S_1$, $S_2$, D** | Comparing values |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | ○ | ○ | ○ | | | | | | | | | | | |

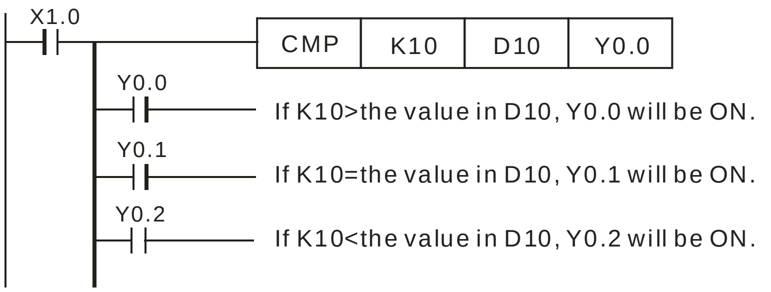
| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **$S_1$**: Comparison value 1; **$S_2$**: Comparison value 2; **D**: Comparison result
- The instruction is used to compare the value in **$S_1$** with that in **$S_2$**. The comparison result is stored in **D**.
- The operand **D** occupies three consecutive devices.
- If the operand **D** is Y0.0, Y0.0, Y0.1, and Y0.2 will be occupied automatically.

## Example

- When X1.0 is ON, the instruction CMP is executed, and Y0.0, Y0.1, or Y0.2 is ON. When X1.0 is OFF, the execution of the instruction CMP stops, and the states of Y0.0, Y0.1, and Y0.2 remain unchanged.
- If users want to get the result that K10≧the value in D10, they have to connect Y0.0 and Y0.1 in series. If users want to get the result that K10≦the value in D10, they have to connect Y0.1 and Y0.2 in series. If users want to get the result that K10≠the value in D10, they have to connect Y0.0, Y0.1, and Y0.2 in series.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 11 | D | ZCP | P | **$S_1$, $S_2$, S, D** | Zonal comparison |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | ○ | ○ | ○ | | | | | | | | | | | |

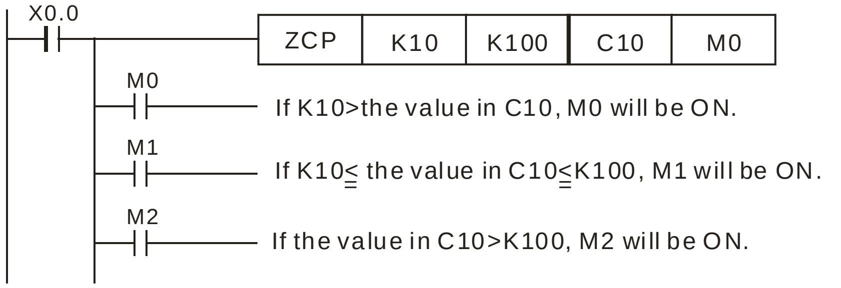
| Pulse instruction | 16-bit instruction (9 steps) | 32-bit instruction (12 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **$S_1$**: Minimum value; **$S_2$**: Maximum value; **S**: Comparison value; **D**: Comparison result
- 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 comparison result is stored in **D**.
- The value in **$S_2$** must be greater than that in **$S_1$**.
- The operand **D** occupies three consecutive devices.
- If the operand **D** is M0, M0, M1, and M2 will be occupied automatically.

## Example

- When X0.0 is ON, the instruction ZCP is executed, and M0, M1, or M2 is ON. When X0.0 is OFF, the execution of the instruction ZCP stops, and the states of M0, M1, and M2 remain unchanged.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 12 | D | MOV | P | **S, D** | Transferring a value |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

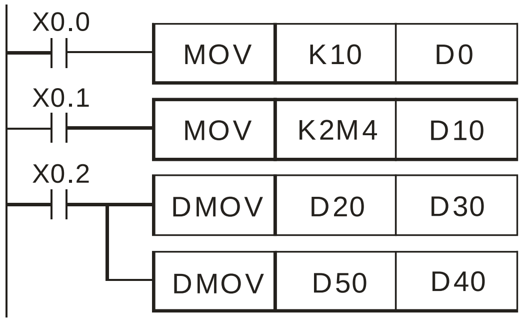
5

Explanation

- **S**: Source; **D**: Destination
- When the instruction is executed, the value in **S** is transferred to **D**. When the instruction is not executed, the value in **D** is unchanged.
- If an operation result gotten is a 32-bit value, users can only move the operation result by means of the instruction DMOV.

Example

- If users want to move a 16-bit value, they have to use the instruction MOV.
  1. When X0.0 is OFF, the value in D0 is unchanged. When X0.0 is ON, the value K10 is transferred to the data register D0.
  2. When X0.1 is OFF, the value in D10 is unchanged. When X0.1 is ON, the value in K2M4 is transferred to the data register D10.
- If users want to move a 32-bit value, they have to use the instruction DMOV.
  When X0.2 is OFF, the values in (D31, D30) and (D41, D40) are unchanged. When X0.2 is ON, the value in (D21, D20) is transferred to (D31, D30), and the value in (D51, D50) is transferred to (D41, D40).



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 14 | D | CML | P | **S, D** | Inverting bits |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

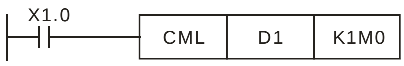
| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

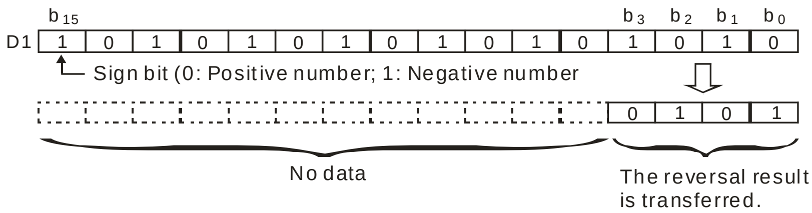
## Explanation

- **S**: Source; **D**: Destination
- The instruction is used to invert the bits in **S** (0→1 and 1→0), and transfer the inversion result to **D**.

## Example

- When X1.0 is ON, bit 0~bit 3 in D1 are inverted, and the inversion result is transferred to M0~M3.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 15 | | BMOV | P | **S, D, n** | Transferring values |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | ● | ● | ● | ● | ● | ● | | |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | | |
| **n** | | | | | ○ | ○ | | | | | | | | | |

| Pulse instruction | 16-bit instruction (7steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

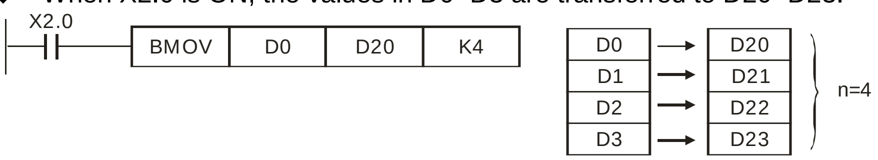


## Explanation

- **S**: Source; **D**: Destination; **n**: Length
- The instruction is used to transfer the values in registers to new registers. The values in the **n** registers starting from **S** are transferred to the **n** registers starting from **D**. If **n** is not in the range available, only the values in registers available will be transferred.
- **n** is in the range of 1 to 512.

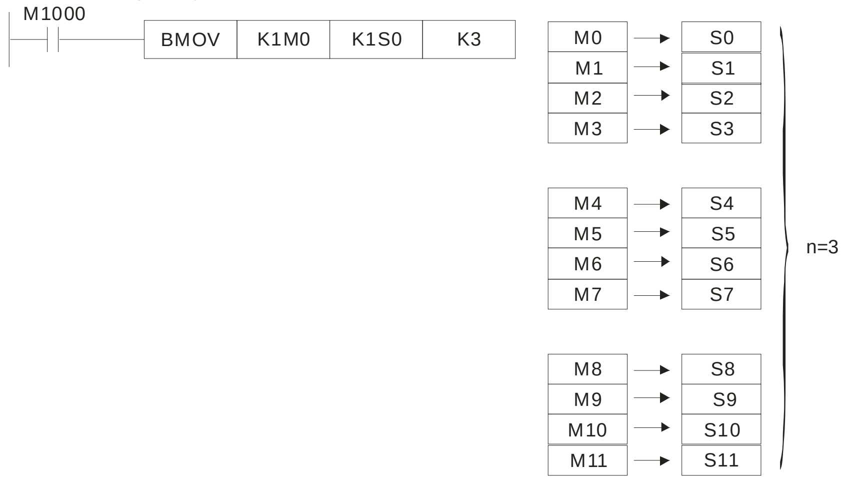
## Example 1

- When X2.0 is ON, the values in D0~D3 are transferred to D20~D23.




## Example 2

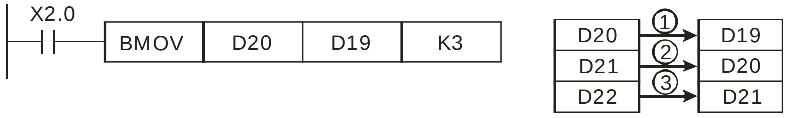
- If users specify KnM and KnS, n in KnM must be the same as n in KnS.



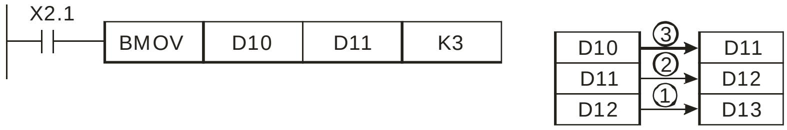
**Example 3**

- In order to prevent the error which results from the overlap between source devices and destination devices, the values in the source devices are transferred in the following way.

1. The device number of **S** is greater than the device number of **D**. The values in D20~D22 are transferred in the order ①→②→③.




2. The device number of **S** is less than the device number of **D**. The values in D10~D12 are transferred in the order ③→②→①. The values in D11~D13 are the same as the value in D10.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 16 | | FMOV | P | **S, D, n** | Transferring a value to several devices |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | | |
| **n** | | | | | ○ | ○ | | | | | | | | | |

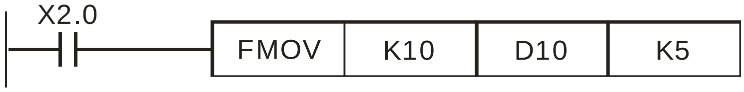
| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (8 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

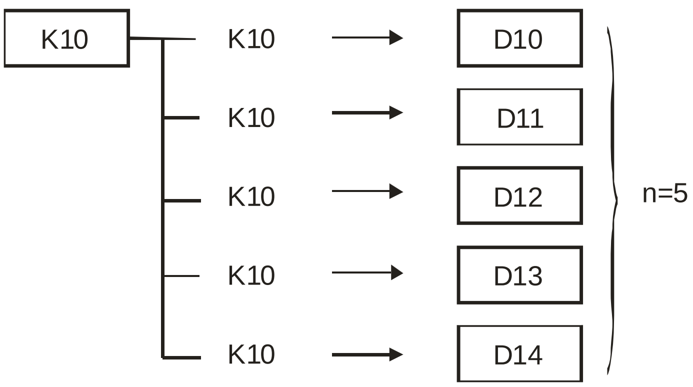
**Explanation**

- **S**: Source; **D**: Destination; **n**: Length
- The value in **S** is transferred to the n registers starting from **D**. If **n** is not in the range available, a value will only be transferred to registers available.
- **n** is in the range of 1 to 512.

**Example**

- When X2.0 is ON, K10 is transferred to the 5 registers starting from D10 (D10~D14).



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 17 | D | XCH | P | $D_1$, $D_2$ | Interchanging values |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| $D_2$ | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

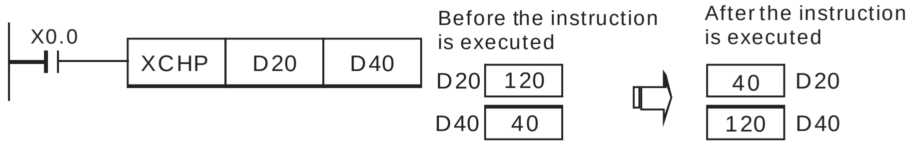
| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (5 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

**Explanation**

- **$D_1$**: Value which is interchanged; **$D_2$**: Value which is interchanged
- The instruction is used to interchange the value in **$D_1$** with the value in **$D_2$**.
- It is suggested that users should use the pulse instruction XCHP.

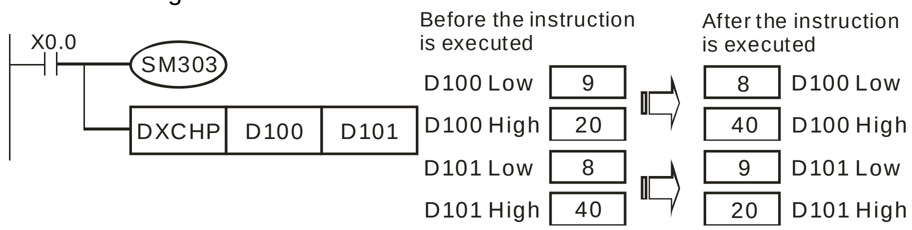
**Example**

- When X0.0 is turned from OFF to ON, the value in D20 is interchanged with the value in D40.




**Additional remark**

- 16-bit instruction: If **$D_1$** is the same as **$D_2$**, and SM303 is ON, the high 8 bits are interchanged with the low 8 bits.
- 32-bit instruction: If **$D_1$** is the same as **$D_2$**, and SM303 is ON, the high 16 bits are interchanged with the low 16 bits.
- When X0.0 is ON, and SM303 is ON, the high 8 bits in D100 are interchanged with the high 8 bits in D101, and the low 8 bits in D100 are interchanged with the low 8 bits in D101.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 18 | D | BCD | P | **S, D** | Converting a binary number into a binary-coded decimal number |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

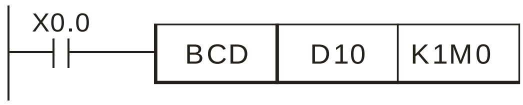
| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

Explanation

- The binary value in **S** is converted into a binary-coded decimal value, and the conversion result is transferred to **D**.
- If a binary number is converted to a binary-coded decimal number which is not in the range of 0 to 9,999, the instruction BCD will not be executed. If a binary number is converted to a binary-coded decimal number which is not in the range of 0 to 99,999,999, the instruction DBCD will not be executed.
- BCD can be used to convert the binary value in a positioning unit to a binary-coded decimal value, and transfer the conversion result to an external device, e.g. a seven-segment display.
- SM1049 is an Ox motion subroutine error flag, and SM953 is an O100 error flag.

Example

- When X0.0 is ON, the binary value in D10 is converted into a binary-coded decimal value, and the digit in the ones place of the conversion result is stored in K1M0 (M0~M3).




If D10=001E (hexadecimal number)=0030 (decimal number), M0~M3=0000 (binary number).

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 19 | D | BIN | P | **S, D** | Converting a binary-coded decimal number into a binary number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

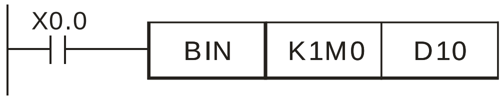
| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (5 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **S**: Source; **D**: Conversion result
- The binary-coded decimal value in **S** is converted into a binary value, and the conversion result is transferred to **D**.
- The 16-bit binary-coded decimal value in **S** should be in the range of 0 to 9,999, and the 32-bit binary-coded decimal value in **S** should be in the range of 0 to 99,999,999.
- Decimal constants and hexadecimal constants are converted into binary numbers automatically. Users do not need to use the instruction.
- SM1049 is an Ox motion subroutine error flag, and SM953 is an O100 error flag.

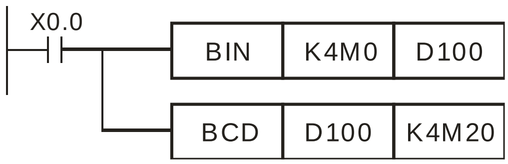
## Example

- When X0.0 is ON, the binary-coded decimal value in K1M0 is converted into a binary value, and the conversion result is stored in D10.




## Additional remark

- Applications of the instructions BCD and BIN:
  1. If a motion control module wants to read a binary-coded decimal value created by a DIP switch, users have to use the instruction BIN to convert the value into a binary value, and store the conversion result in the motion control module.
  2. If users want to display a value stored in a motion control module on a seven-segment display on which binary-coded decimal numbers can be displayed, they have to use the instruction BCD to convert the value into a binary-coded decimal value, and transfer the conversion result to the seven-segment display.
  3. When X0.0 is ON, the binary-coded decimal value in K4M0 is converted into a binary value, and the conversion result is stored in D100. Subsequently, the binary value in D100 is converted into a binary-coded decimal value, and the conversion result is stored in K4M20.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 20 | D | ADD | P | **$S_1$, $S_2$, D** | Binary addition |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

Explanation

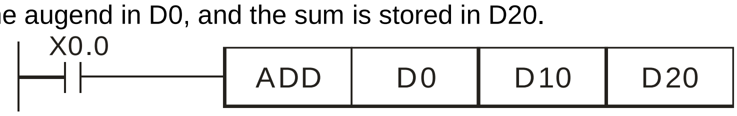
- **$S_1$**: Augend; **$S_2$**: Addend; **D**: Sum
- The binary value in **$S_2$** is added to the binary value in **$S_1$**, and the sum is stored in **D**.
- The highest bit in **$S_1$** and the highest bit in **$S_2$** are sign bits. If the sign bit in a register is 0, the value in the register is a positive value. If the sign bit in a register is 1, the value in the register is a negative value.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.
- The flags related to 16-bit binary addition and 32-bit binary addition are listed below.

  16-bit binary addition:
  1. If the operation result gotten is 0, a zero flag will be ON.
  2. If the operation result gotten is less than –32,768, a borrow flag will be ON.
  3. If the operation result gotten is greater than 32,767, a carry flag will be ON.

  32-bit binary addition:
  1. If the operation result gotten is 0, a zero flag will be ON.
  2. If the operation result gotten is less than –2,147,483,648, a borrow flag will be ON.
  3. If the operation result gotten is greater than 2,147,483,647, a carry flag will be ON.

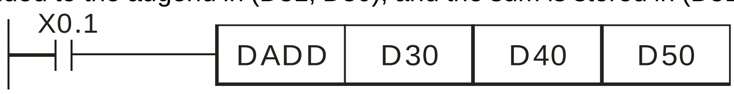
Example 1

- 16-bit binary addition: When X0.0 is ON, the addend in D10 is added to the augend in D0, and the sum is stored in D20.



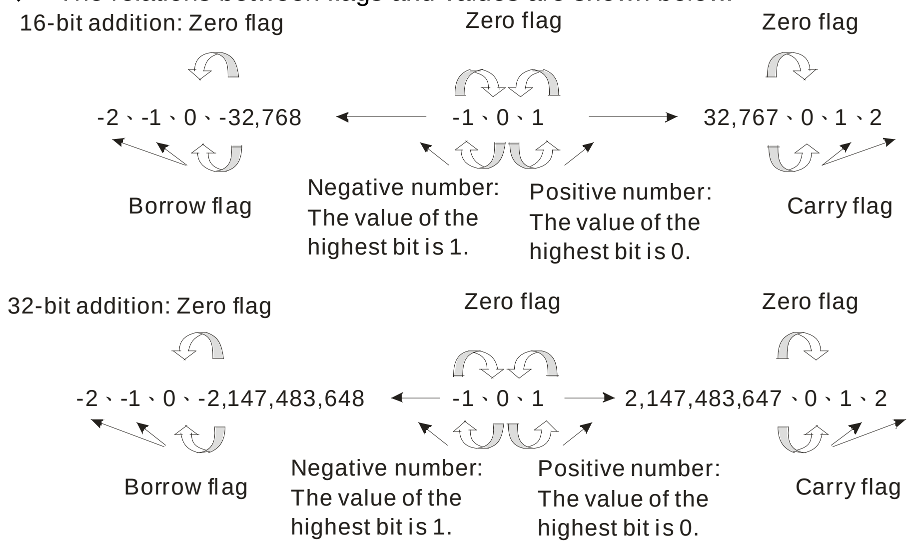

Example 2

- 32-bit binary addition: When X0.1 is ON, the value in (D41, D40) is added to the augend in (D31, D30), and the sum is stored in (D51, D50).



## Additional remark

◆ The relations between flags and values are shown below.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 21 | D | SUB | P | **$S_1$, $S_2$, D** | Binary subtraction |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **$S_1$**: Minuend; **$S_2$**: Subtrahend; **D**: Difference
- The binary value in **$S_2$** is subtracted from the binary value in **$S_1$**, and the difference is stored in **D**.
- The highest bit in **$S_1$** and the highest bit in **$S_2$** are sign bits. If the sign bit in a register is 0, the value in the register is a positive value. If the sign bit in a register is 1, the value in the register is a negative value.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.
- The flags related to 16-bit binary subtraction and 32-bit binary subtraction are listed below.

  16-bit binary subtraction:

  1. If the operation result gotten is 0, a zero flag will be ON.
  2. If the operation result gotten is less than –32,768, a borrow flag will be ON.
  3. If the operation result gotten is greater than 32,767, a carry flag will be ON.

  32-bit binary subtraction:

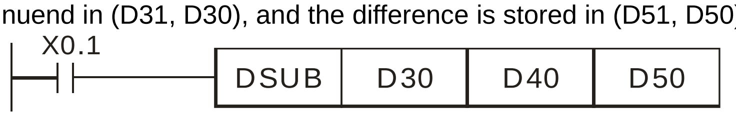
  1. If the operation result gotten is 0, a zero flag will be ON.
  2. If the operation result gotten is less than –2,147,483,648, a borrow flag will be ON.
  3. If the operation result gotten is greater than 2,147,483,647, a carry flag will be ON.
- Please refer to the additional remark on the instruction ADD for more information about the relations between flags and values.

## Example 1

- 16-bit binary subtraction: When X0.0 is ON, the subtrahend in D10 is subtracted from the minuend in D0, and the difference is stored in D20.


## Example 2

- When X0.1 is ON, the subtrahend in (D41, D40) is subtracted from the minuend in (D31, D30), and the difference is stored in (D51, D50).



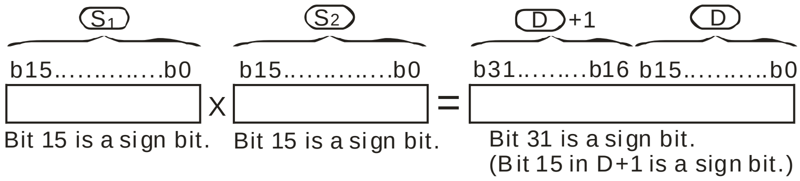
| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 22 | D | MUL | P | **$S_1$, $S_2$, D** | Binary multiplication |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | | | ● | ● | ● | ● | | |

| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

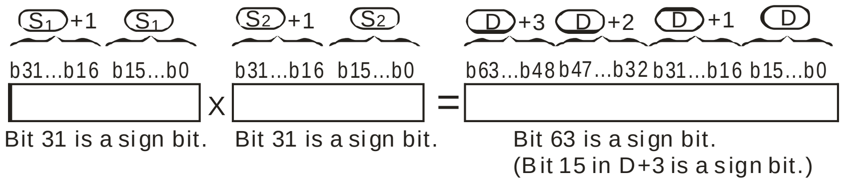
## Explanation

- **$S_1$**: Multiplicand; **$S_2$**: Multiplier; **D**: Product
- The signed binary value in **$S_1$** is multiplied by the singed binary value in **$S_2$**, and the product is stored in **D**. Users have to notice the sign bits in **$S_1$**, **$S_2$**, and **D** when 16-bit binary multiplication or 32-bit binary multiplication is done.
- 16-bit binary multiplication




Sign bit=0 (Positive sign); Sign bit=1 (Negative sign)

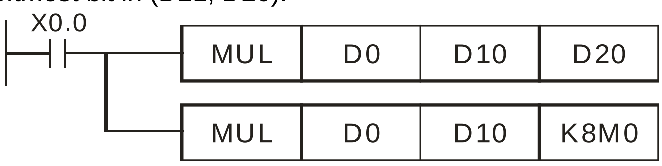
- 32-bit binary multiplication




Sign bit=0 (Positive sign); Sign bit=1 (Negative sign)

## 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 bits in D21 is the high 16 bits in (D21, D20), whereas the bits in D20 is the low 16 bits in (D21, D20). Whether the product is a positive value or a negative value depends on the leftmost bit in (D21, D20).



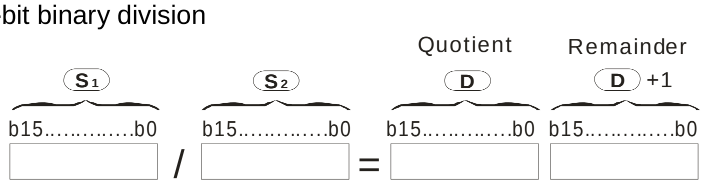
| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 23 | D | DIV | P | **$S_1$, $S_2$, D** | Binary division |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | | | ● | ● | ● | ● | | |

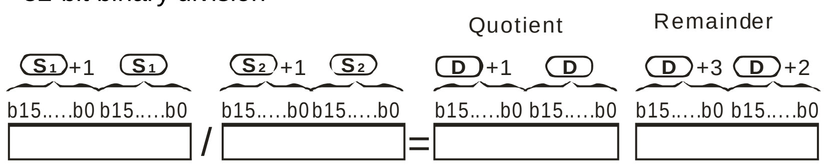
| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **$S_1$**: Dividend; **$S_2$**: Divisor; **D**: Quotient and remainder
- 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**. Users have to notice the sign bits in **$S_1$**, **$S_2$**, and **D** when 16-bit binary division or 32-bit binary division is done.
- If the divisor in **$S_2$** is 0, the instruciton will not be executed.
- 16-bit binary division

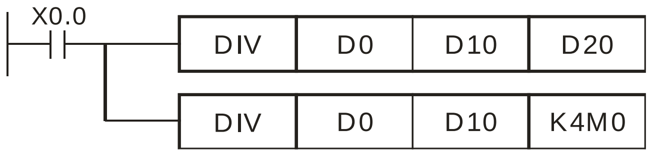



- 32-bit binary division




## 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 quotient and the remainder are positive values or negative values depends on the leftmost bit in D20 and the leftmost bit in D21.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 24 | D | INC | P | **D** | Adding one to a binary number |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

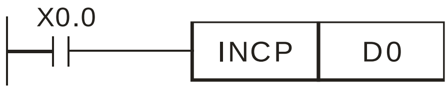
| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction (3 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **D**: Destination device
- If the instruction used is not a pulse instruction, the value in **D** used by the instruction increases by one whenever the instruction is executed.
- Generally, the pulse instructions INCP and DINCP are used.
- If a 16-bit operation is performed, 32,767 plus 1 equals -32,768. If a 32-bit operation is performed, 2,147,483,647 plus 1 equals -2,147,483,648.

## Example

- When X0.0 is turned from OFF to ON, the value in D0 increases by one.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 25 | D | DEC | P | **D** | Subtracting one from a binary number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

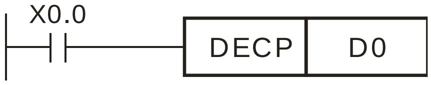
| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction (3 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

Explanation

- **D**: Destination device
- If the instruction used is not a pulse instruction, the value in **D** used by the instruction decreases by one whenever the instruction is executed.
- Generally, the pulse instructions DECP and DDECP are used.
- If a 16-bit operation is performed, -32,768 minus 1 leaves 32,767. If a 32-bit operation is performed, -2,147,483,648 minus 1 leaves 2,147,483,647.

Example

- When X0.0 is turned from OFF to ON, the value in D0 decreases by one.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 26 | D | WAND | P | $S_1$, $S_2$, D | Logical AND operation |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| $S_2$ | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| D | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

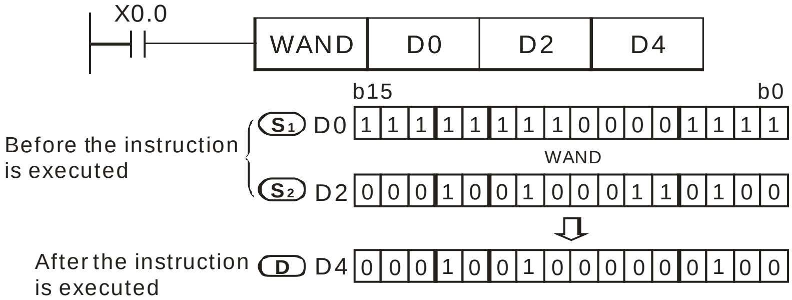
| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **$S_1$**: Source device 1; **$S_2$**: Source device 2; **D**: Operation result
- A logical AND operator 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**.
- The result in each position is 1 if the first bit is 1 and the second bit is 1. Otherwise, the result is 0.

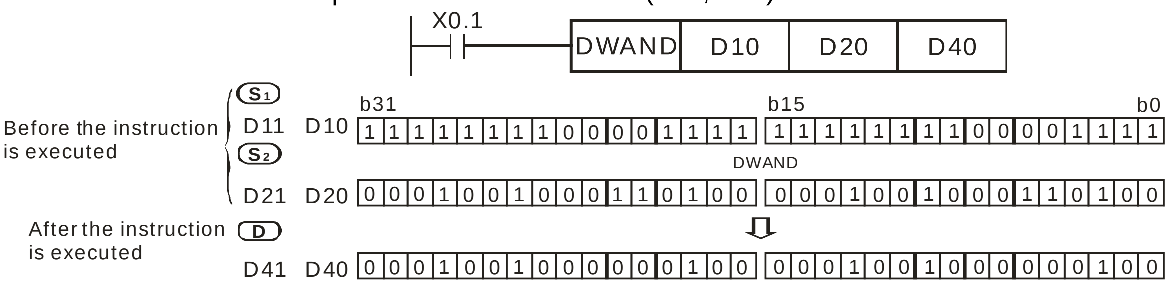
## Example 1

- When X0.0 is ON, a logical AND operator takes the values in the 16-bit device D0 and the 16-bit device D2, and performs the logical AND operation on each pair of corresponding bits, and the operation result is stored in Y4.




## Example 2

- When X0.1 is ON, a logical AND operator takes the values in the 32-bit device (D11, D10) and the 32-bit device (D21, D20), and performs the logical AND operation on each pair of corresponding bits, and the operation result is stored in (D41, D40).



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 27 | D | WOR | P | **$S_1$, $S_2$, D** | Logical OR operation |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **$S_1$**: Source device 1; **$S_2$**: Source device 2; **D**: Operation result
- A logical OR operator 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**.
- 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, a logical OR operator takes the values in the 16-bit device D0 and the 16-bit device D2, and performs the logical inclusive OR operation on each pair of corresponding bits, and the operation result is stored in D4.

X0.0 ─┤├─ | WOR | D0 | D2 | D4 |

| | | b15 | | | | | | | | | | | | | | | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Before the instruction is executed | S1 D0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | WOR | | | | | | | | | | | | | | | | |
| | S2 D2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| After the instruction is executed | D D4 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

## Example 2

- When X0.1 is ON, a logical OR operator takes the values in the 32-bit device (D11, D10) and the 32-bit device (D21, D20), and performs the logical inclusive OR operation on each pair of corresponding bits, and the operation result is stored in (D41, D40).

X0.1 ─┤├─ | DWOR | D10 | D20 | D40 |

| | | | b31 |
|---|---|---|---|
| Before the instruction is executed | S1 | D11 D10 | 1111 1111 0000 1111 (b15) 1111 1111 0000 1111 (b0) |
| | | DWOR | |
| | S2 | D21 D20 | 0001 0010 0011 0100 0001 0010 0011 0100 |
| After the instruction is executed | D | D41 D40 | 1111 1111 0011 1111 1111 1111 0011 1111 |

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 28 | D | WXOR | P | **$S_1$, $S_2$, D** | Logical exclusive OR operation |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

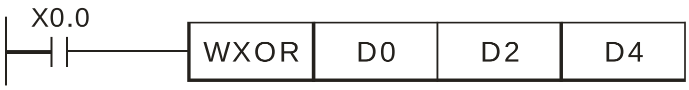
| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **$S_1$**: Source device 1; **$S_2$**: Source device 2; **D**: Operation result
- A logical XOR operator 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**.
- The result in each position is 1 if the two bits are different, and 0 if they are the same.

## Example 1

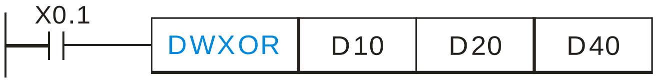
- When X0.0 is ON, a logical XOR operator takes the values in the 16-bit device D0 and the 16-bit device D2, and performs the exclusive OR operation on each pair of corresponding bits, and the operation result is stored in D4.




| | | b15 | | | | | | | | | | | | | | | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Before the instruction is executed | $S_1$ D0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | WXOR | | | | | | | | | | | | | | | | |
| | $S_2$ D2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| After the instruction is executed | D D4 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

## Example 2

- When X0.1 is ON, a logical XOR operator takes the values in the 32-bit device (D11, D10) and the 32-bit device (D21, D20), and performs the logical exclusive OR operation on each pair of corresponding bits, and the operation result is stored in (D41, D40).




| | | b31 … | b15 … b0 |
|---|---|---|---|
| Before the instruction is executed | $S_1$ D11 D10 | 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 | 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 |
| | DWXOR | | |
| | $S_2$ D21 D20 | 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 | 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 |
| After the instruction is executed | D D41 D40 | 1 1 1 0 1 1 0 1 0 0 1 1 1 0 1 1 | 1 1 1 0 1 1 0 1 0 0 1 1 1 0 1 1 |

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 29 | D | NEG | P | **D** | Taking the two's complement of a number |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction (3 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

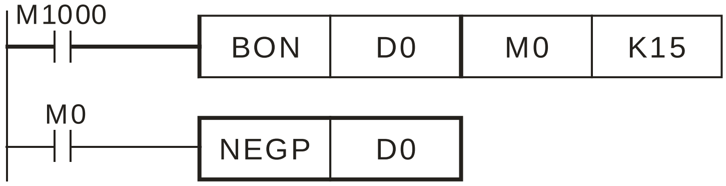
- **D**: Device in which the two's complement of the value in the device is stored
- The instructions can be used to convert a negative binary value into an absolute value.
- Generally, the pulse instructions NEGP and DNEGP are used.

## Example 1

- When X0.0 is turned from OFF to ON, all the bits in D0 are inverted (0 becomes 1, and 1 becomes 0), 1 is added to the result, and the final value is stored in the original register D10.

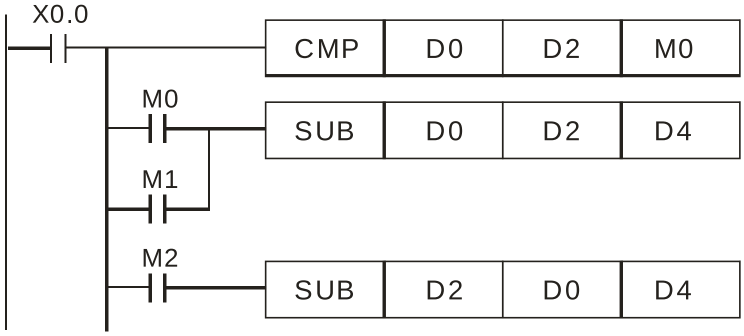

## Example 2

- Getting the aboluste value of a negative number
  1. When bit 15 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 take the two's complement of the negative value in D0. (The corresponding positive value is gotten.)




## Example 3

- Getting 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 a negative value and its absolute value are described below.
  1. Whether the value in a register is a positive value or a negative value depends on the leftmost bit in the register. If the leftmost bit in a register is 0, the value in the register is a positive value. If the leftmost bit in a register is 1, the value in the register is a negative value.
  2. The negative value in a register 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

(D0)=-1 → $(\overline{D0})+1=1$

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 → 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

(D0)=-2 → $(\overline{D0})+1=2$

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 → 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

(D0)=-3 → $(\overline{D0})+1=3$

1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 → 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1

(D0)=-4 → $(\overline{D0})+1=4$

1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 → 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

(D0)=-5 → $(\overline{D0})+1=5$

1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 → 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1

⋮

(D0)=-32,765 → $(\overline{D0})+1=32{,}765$

1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 → 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1

(D0)=-32,766 → $(\overline{D0})+1=32{,}766$

1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 → 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0

(D0)=-32,767 → $(\overline{D0})+1=32{,}767$

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 → 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

(D0)=-32,768 → $(\overline{D0})+1=-32{,}768$

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 → 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The maximum absolute value is 32,767.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 30 | D | ROR | P | **D, n** | Rotating bits rightwards |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **n** | | | | | ○ | ○ | | | | | | | | | |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

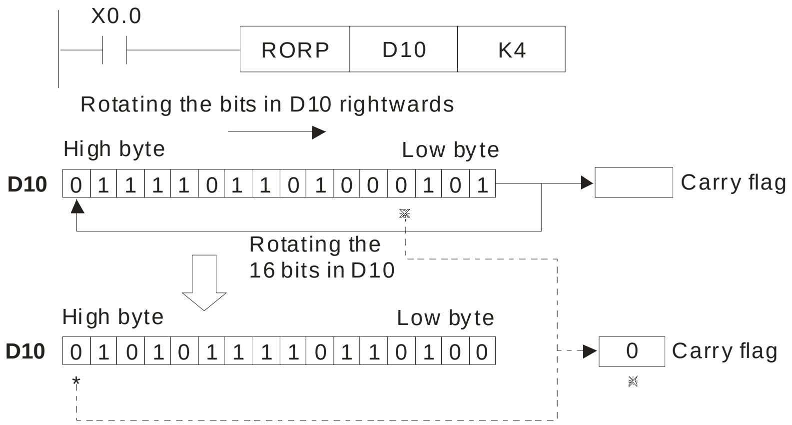
## Explanation

- **D**: Device which is rotated; **n**: Number of bits forming a group
- The bits in **D** are divided into groups (**n** bits as a group), and these groups are rotated rightwards.
- The **n**th bit from the right is transmitted to a carry flag.
- Generally, the pulse instructions RORP and DRORP are used.
- If the operand **D** is KnM/KnS, Kn in KnM/KnS must be K4 (16 bits) or K8 (32 bits).
- 16-bit instruction: 1≤**n**≤16; 32-bit instruction: 1≤**n**≤32
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

## Example

- When X0.0 is turned from OFF to ON, the bits in D10 are divided into groups (four bits as a group), and these groups are rotated rightwards.

(The bit marked with ※ is transmitted to a carry flag.)



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 31 | D | ROL | P | D, n | Rotating bits leftwards |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| n | | | | | ○ | ○ | | | | | | | | | |

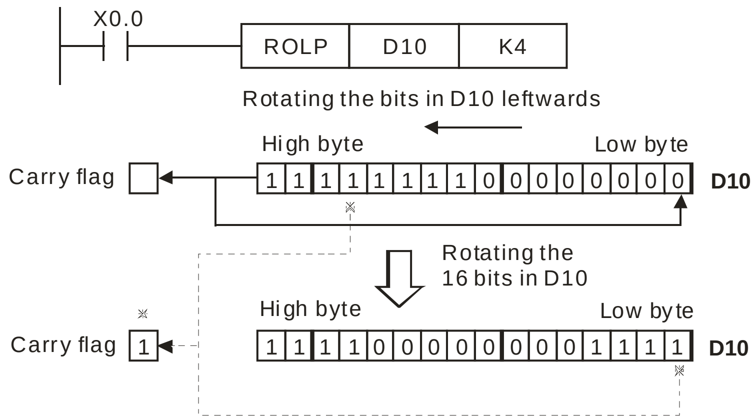
| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **D**: Device which is rotated; **n**: Number of bits forming a group
- The bits in **D** are divided into groups (**n** bits as a group), and these groups are rotated leftwards.
- The **n**th bit from the left is transmitted to a carry flag.
- Generally, the pulse instructions ROLP and DROLP are used.
- If the operand **D** is KnM/KnS, Kn in KnM/KnS must be K4 (16 bits) or K8 (32 bits).
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K4M0 and K4S16 (decimal numeral system).
- 16-bit instruction: 1≤**n**≤16; 32-bit instruction: 1≤**n**≤32
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

## Example

- When X0.0 is turned from OFF to ON, the bits in D10 are divided into groups (four bits as a group), and these groups are rotated leftwards. (The bit marked with ※ is transmitted to a carry flag.)



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 32 | D | RCR | P | **D, n** | Rotating bits rightwards with a carry flag |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **n** | | | | | ○ | ○ | | | | | | | | | |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

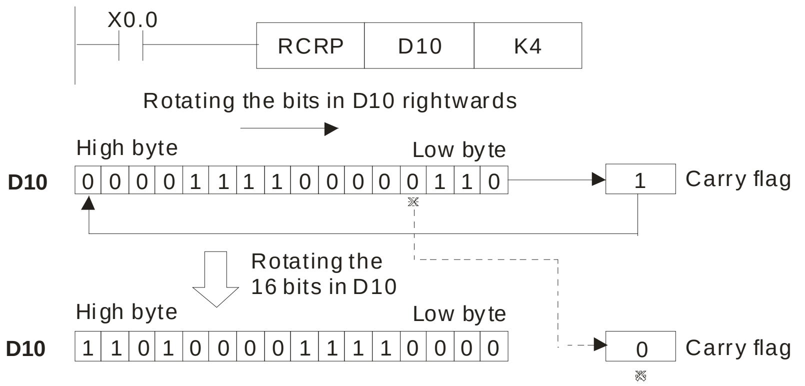
## Explanation

- **D**: Device which is rotated; **n**: Number of bits forming a group
- The bits in **D** are divided into groups (**n** bits as a group), and these groups are rotated rightwards with a carry flag.
- The **n**th bit from the right is transmitted to a carry flag.
- Generally, the pulse instructions RCRP and DRCRP are used.
- If the operand **D** is KnM/KnS, Kn in KnM/KnS must be K4 (16 bits) or K8 (32 bits).
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K4M0 and K4S16 (decimal numeral system).
- 16-bit instruction: 1≤**n**≤16; 32-bit instruction: 1≤**n**≤32
- SM1066 is the carry flag in an Ox motion subroutine, and SM22 is a carry flag.

5

## Example

- When X0.0 is turned from OFF to ON, the bits in D10 are divided into groups (four bits as a group), and these groups are rotated rightwards with a carry flag. (The bit marked with ※ is transmitted to the carry flag.)



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 33 | D | RCL | P | **D, n** | Rotating bits leftwards with a carry flag |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **n** | | | | | ○ | ○ | | | | | | | | | |

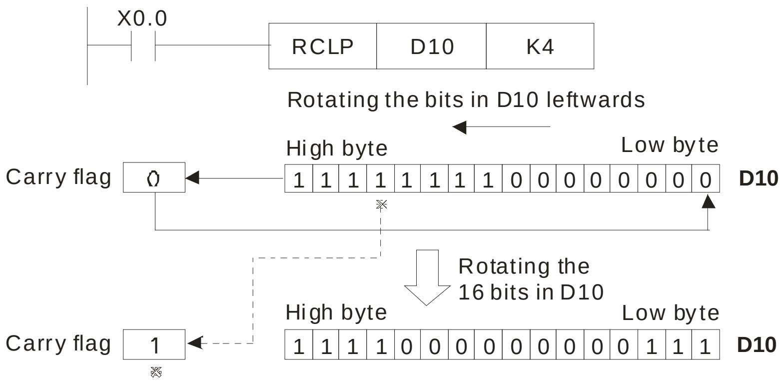
| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **D**: Device which is rotated; **n**: Number of bits forming a group
- The bits in **D** are divided into groups (**n** bits as a group), and these groups are rotated leftwards with a carry flag.
- The **n**th bit from the left is transmitted to a carry flag.
- Generally, the pulse instructions RCLP and DRCLP are used.
- If the operand **D** is KnM/KnS, Kn in KnM/KnS must be K4 (16 bits) or K8 (32 bits).
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K4M0 and K4S16 (decimal numeral system).
- 16-bit instruction: 1≤**n**≤16; 32-bit instruction: 1≤**n**≤32
- SM1066 is the carry flag in an Ox motion subroutine, and SM22 is a carry flag.

## Example

- When X0.0 is turned from OFF to ON, the bits in D10 are divided into groups (four bits as a group), and these groups are rotated leftwards with a carry flag. (The bit marked with ※ is transmitted to the carry flag.)



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 34 | | SFTR | P | **S, D, $n_1$, $n_2$** | Moving the states of bit devices rightwards |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | | | | |
| **D** | | ● | ● | ● | | | | | | | | | | | |
| **$n_1$** | | | | | ○ | ○ | | | | | | | | | |
| **$n_2$** | | | | | ○ | ○ | | | | | | | | | |

| Pulse instruction | 16-bit instruction (9 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

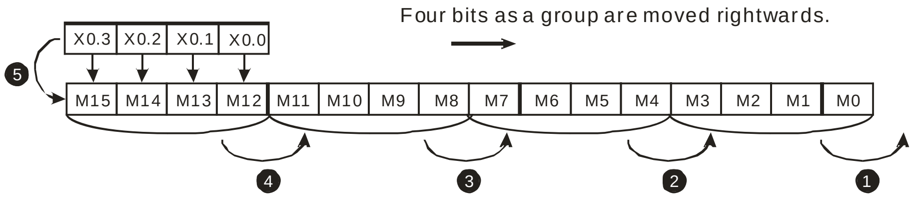
**Explanation**

- **S**: Initial device which is moved; **D**: Initial device which is moved; **$n_1$**: Number of bits which are moved; **$n_2$**: Number of bits forming a group
- 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 moved rightwards. The states of the **$n_2$** bit devices starting from **S** are moved to the vacant devices in the devices starting from **D**.
- Generally, the pulse instruction SFTRP is used.
- $1 \leq n_2 \leq n_1 \leq 1024$

**Example**

- When X0.0 is turned from OFF to ON, the states of the sixteen bit devices starting from M0 are divided into groups (four bits as a group), and these groups are moved rightwards.
- The states of the bit devices are moved rightwards in the order ❶~❺ during a scan cycle.

❶ M3~M0 → The states of M3~M0 are carried.
❷ M7~M4 → M3~M0
❸ M11~M8 → M7~M4
❹ M15~M12 → M11~M8
❺ X0.3~X0.0 → M15~M12

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 35 | | SFTL | P | **S, D, $n_1$, $n_2$** | Moving the states of bit devices leftwards |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | | | | | | |
| **D** | | ● | ● | ● | | | | | | | | | | | |
| **$n_1$** | | | | | ○ | ○ | | | | | | | | | |
| **$n_2$** | | | | | ○ | ○ | | | | | | | | | |

| Pulse instruction | 16-bit instruction (9 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

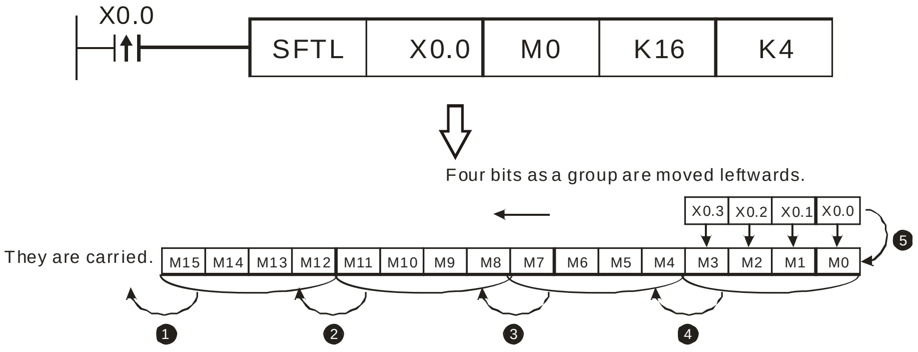
## Explanation

- **S**: Initial bit device which is moved; **D**: Initial bit device which is moved; **$n_1$**: Number of bits which are moved; **$n_2$**: Number of bits forming a group
- 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 moved leftwards. The states of the **$n_2$** bit devices starting from **S** are moved to the vacant devices in the devices starting from **D**.
- Generally, the pulse instruction SFTRP is used.
- $1 \leq n_2 \leq n_1 \leq 1024$

## Example

- When X0.0 is turned from OFF to ON, the states of the sixteen bit devices starting from M0 are divided into groups (four bits as a group), and these groups are moved leftwards.
- The states of the bit devices are moved leftwards in the order ❶~❺ during a scan cycle.

❶ M15~M12 → The states of M3~M0 are carried.
❷ M11~M8 → M15~M12
❸ M7~M4 → M11~M8
❹ M3~M0 → M7~M4
❺ X0.3~X0.0 → M3~M0




5

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 36 | | WSFR | P | **S, D, $n_1$, $n_2$** | Moving the values in word devices rightwards |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | ● | ● | ● | ● | ● | ● | | |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | | |
| **$n_1$** | | | | | ○ | ○ | | | | | | | | | |
| **$n_2$** | | | | | ○ | ○ | | | | | | | | | |

| Pulse instruction | 16-bit instruction (9 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

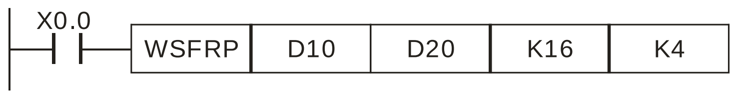
## Explanation

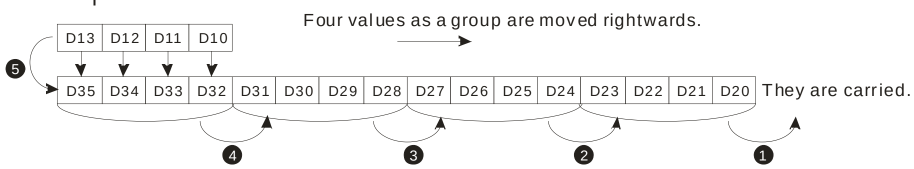
- **S**: Initial word device which is moved; **D**: Initial word device which is moved; **$n_1$**: Number of values which are moved; **$n_2$**: Number of values forming a group
- The values in the **$n_1$** word devices starting from **D** are divided into groups (**$n_2$** values as a group), and these groups are moved rightwards. The values in the **$n_2$** word devices starting from **S** are moved to the vacant word devices in the word devices starting from **D**.
- Generally, the pulse instruction WSFRP is used.
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K1M0 and K4S16 (decimal numeral system).
- 1≤**n2**≤**n1**≤512

## Example

- When X0.0 is turned from OFF to ON, the values in the sixteen word devices starting from D20 are divided into groups (four values as a group), and these groups are moved rightwards.
- The values in the word devices are moved rightwards in the order ❶~❺ during a scan cycle.

❶ D23~D20 → The values in D23~D20 are carried.
❷ D27~D24 → D23~D20
❸ D31~D28 → D27~D24
❹ D35~D32 → D31~D28
❺ D13~D10 → D35~D32



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 37 | | WSFL | P | **S, D, $n_1$, $n_2$** | Moving the values in word devices leftwards |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | ● | ● | ● | ● | ● | ● | | |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | | |
| **$n_1$** | | | | | ○ | ○ | | | | | | | | | |
| **$n_2$** | | | | | ○ | ○ | | | | | | | | | |

| Pulse instruction | 16-bit instruction (9 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

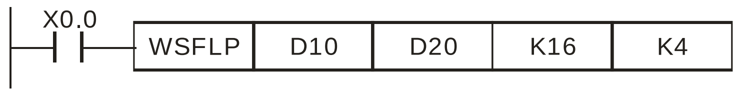
Explanation

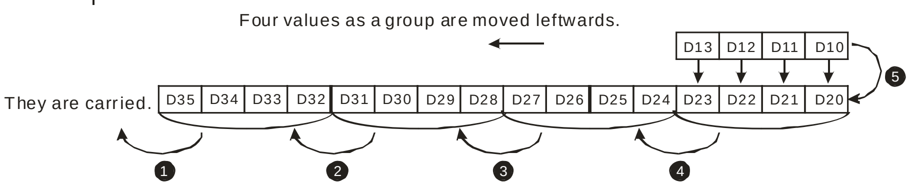
- **S**: Initial word device which is moved; **D**: Initial word device which is moved; **$n_1$**: Number of values which are moved; **$n_2$**: Number of values forming a group
- The values in the **$n_1$** word devices starting from **D** are divided into groups (**$n_2$** values as a group), and these groups are moved leftwards. The values in the **$n_2$** word devices starting from **S** are moved to the vacant word devices in the word devices starting from **D**.
- Generally, the pulse instruction WSFLP is used.
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K1M0 and K4S16 (decimal numeral system).
- 1≤**n2**≤**n1**≤512

Example

- When X0.0 is turned from OFF to ON, the values in the sixteen word devices starting from D20 are divided into groups (four values as a group), and these groups are moved leftwards.
- The values in the word devices are moved leftwards in the order ❶~❺ during a scan cycle.

❶ D35~D32 → The values in D35~D32 are carried.
❷ D31~D28 → D35~D32
❸ D27~D24 → D31~D28
❹ D23~D20 → D27~D24
❺ D13~D10 → D23~D20



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 38 | | SFWR | P | **S, D, n** | Moving a value and writing it into a word device |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | | |
| **n** | | | | | ○ | ○ | | | | | | | | | |

| Pulse instruction | 16-bit instruction (9 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

## Explanation

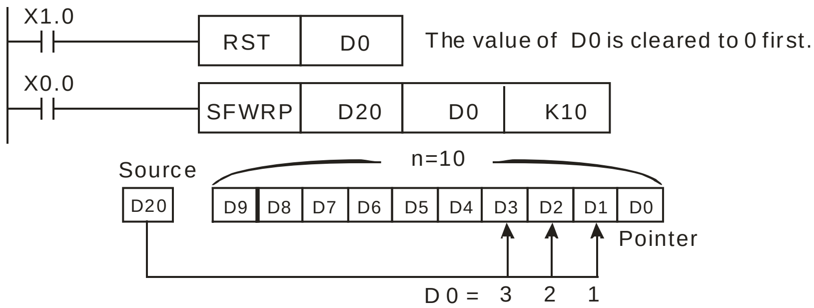
- **S**: Device which is moved; **D**: Initial device; **n**: Number of devices
- The values in the **n** word devices starting from **D** are defined as first in, first out values, and **D** is taken as a pointer. When the instruction is executed, the value of the pointer **D** increases by one, and the value in **S** is written into the device to which the pointer **D** points. When the value of the pointer is greater than or equal to **n**-1, the instruction does not process the writing of the value, and a carry flag is ON.
- When the value of the pointer **D** is greater than **n**-1, the instruction does not process the writing of a value, and the carry flag SM22 is ON. SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K1M0 and K4S16 (decimal numeral system).
- Generally, the pulse instruction SFWRP is used.
- 2≤**n**≤512
- SM20 is a zero flag, and SM968 is the zero flag in O100.

## Example

- The value of the pointer D0 is cleared to 0 first. When X0.0 is turned from OFF to ON, the value in D20 is written into D1, and the value of D0 becomes 1. When X0.0 is turned from OFF to ON again, the value in D20 is written to D2, and the value in D0 becomes 2.
- The value in D20 is moved and written into D1 in the way described below.

  ❶ The value in D20 is written into D1.

  ❷ The value of D0 becomes 1.




## Additional remark

- The instruction SFWR can be used with the instruction SFRD to write a value and read values.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 39 | | SFRD | P | **S, D, n** | Moving a value and reading it from a word device |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | | |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **n** | | | | | ○ | ○ | | | | | | | | | |

| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

Explanation

- **S**: Initial device; **D**: Device into which a value is written; **n**: Number of devices
- The values in the **n** word devices starting from **S** are defined as first in, first out values, and **S** is taken as a pointer. When the instruction is executed, the value in **S** decreases by one, the value in **S+1** is written into **D**, the values in **S**+**n**-1~**S**+2 are moved rightwards, and the value in **S**+**n**-1 is unchanged. When the value in **S** is equal to 0, the instruction does not process the reading of the values, and the zero flag SM20 is ON.
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K1M0 and K4S16 (decimal numeral system).
- Generally, the pulse instruction SFRDP is used.
- 2≤**n**≤512

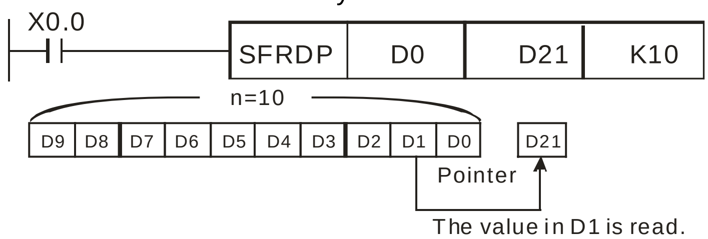
Example

- When X0.0 is turned from OFF to ON, the value in D1 is written into D21, the values in D9~D2 are moved rightwards, the value in D9 is unchanged, and the value in D0 decreases by one.
- The value in D1 is moved and written into D21 in the way described below.

  ❶ The value in D1 is written into D21.

  ❷ The values in D9~D2 are moved rightwards.

  ❸ The value in D0 decreases by one.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 40 | | ZRST | P | $D_1$, $D_2$ | Resetting a zone |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | | ○ | ○ | ○ | | | | | | ○ | ○ | ○ | ○ | | |
| $D_2$ | | ○ | ○ | ○ | | | | | | ○ | ○ | ○ | ○ | | |

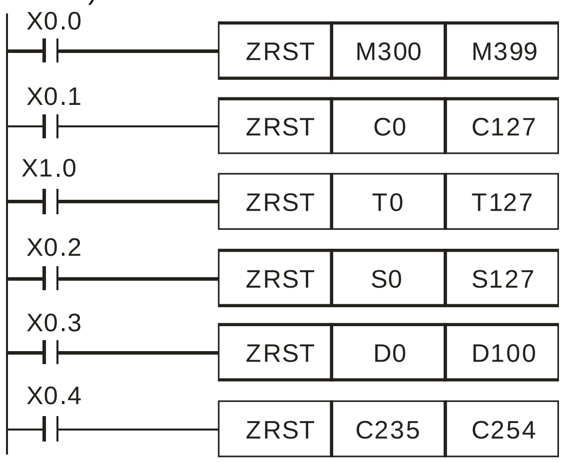
| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

## Explanation

- **$D_1$**: Initial device which is reset; **$D_2$**: Final device which is reset
- The instruction ZRST can be used to reset 16-bit counter and 32-bit counters.
- If the device number of **$D_1$** is greater than the device number of **$D_2$**, only **$D_2$** will be reset.

## Example

- When X0.0 is ON, the auxiliary relays M300~M399 are reset to OFF.
- When X1.0 is ON, the 16-bit counters C0~C127 are reset. (The values of C0~C127 are cleared to 0, and the contacts and the coils are reset to OFF.)
- When X1.0 is ON, the timers T0~T127 are reset. (The values of T0~T127 are cleared to 0. and the contacts and the coils are reset to OFF.)
- When X0.2 is ON, the stepping relays S0~S127 are reset to OFF.
- When X0.3 is ON, the data registers D0~D100 are reset to 0.
- When X0.4 is ON, the 32-bit counters C235~C254 are reset. (The values of C235~C254 are cleared to 0, and the contacts and the coils are reset to OFF.)




## Additional remark

- The instruction RST can be used to reset a single device, e.g. a Y device, an M device, an S device, a T device, a C device, or a D device.

| API | Instruction code | | Operand | Function |
|---|---|---|---|---|
| 41 | DECO | P | S, D, n | Decoder |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | ● | ● | ● | ● | ○ | ○ |
| **D** | | ● | ● | ● | | | | | | ● | ● | ● | ● | ○ | ○ |
| **n** | | | | | ○ | ○ | | | | | | | | | |

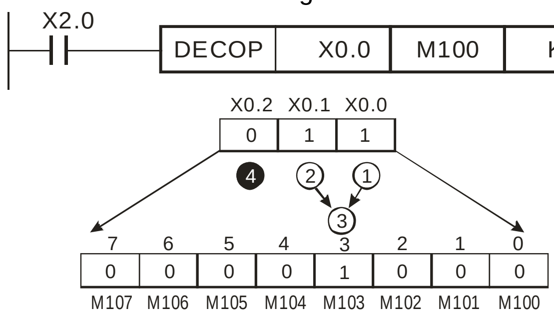
| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

## Explanation

- **S**: Source device; **D**: Device in which a decoding result is stored; **n**: Number of bits which are decoded
- The low **n** bits in **S** are decoded as the low $2^n$ bits in **D**.
- Generally, the pulse instruction DECOP is used.
- **D** is in the range of 1 to 8.

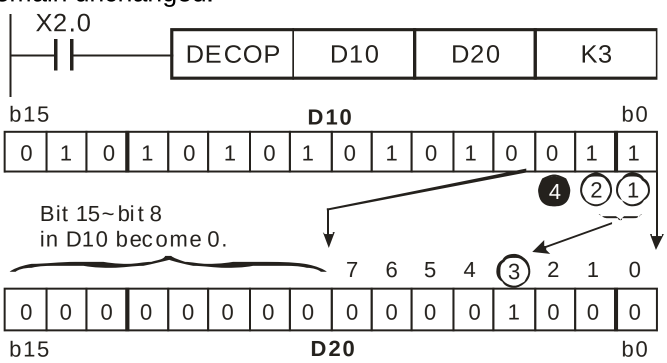
## Example 1

- When **D** is a bit device, **n** is in the range of 1 to 8. If **n** is 0, or greater than 8, an error will occur.
- If **n** is 8, the maximum number of bits which can be decoded is $2^8$=256.
- When X2.0 is turned from OFF to ON, the instruction DECOP decodes X0.0~X0.2 as M100~M107.
- If the value in **S** is 3, M103 will be ON.
- After the instruciton is executed, X2.0 will be OFF, and the states of M100~M107 will remain unchanged.




## Example 2

- When **D** is a word device, **n** is in the range of 1 to 8. If **n** is 0, or greater than 8, an error will occur.
- If **n** is 8, the maximum number of bits which can be decoded is $2^8$=256.
- When X2.0 is turned from OFF to ON, the instruction DECOP decodes b2~b0 in D10 as b7~b0 in D20, and b15~b8 in D20 become 0.
- The low 3 bits in D10 are decoded as the low 8 bits in D20. The high 8 bits in D20 are 0.
- After the instruciton is executed, X2.0 will be OFF, and the value in D20 will remain unchanged.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 42 | | ENCO | P | **S, D, n** | Encoder |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | ● | ● | ● | ● | | | | | | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | | | ● | ● | ● | ● | ○ | ○ |
| **n** | | | | | ○ | ○ | | | | | | | | | |

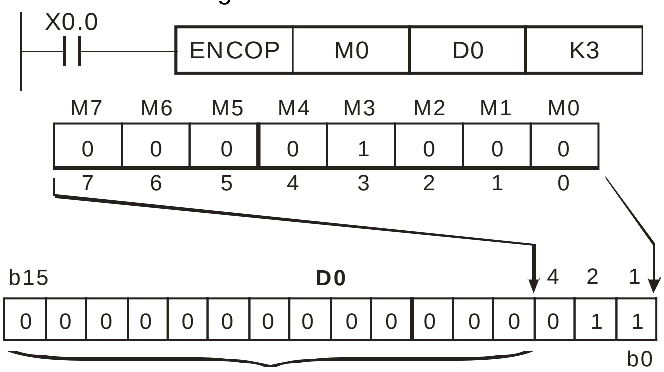
| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

Explanation

- **S**: Source device; **D**: Device in which an encoding result is stored; **n**: Number of bits which are encoded
- The low $2^n$ bits in **S** are encoded as the low **n** bits in **D**.
- If there are many bits which are 1 in **S**, the first bit which is 1 from the left will be processed.
- Generally, the pulse instruction ENCOP is executed.
- The instruction supports V devices and Z devices. (If the 16-bit instructions used, Z devices can not be used. If the 32-bit instruction is used, V devices can not be used.)
- If **S** is a bit device, **n** is in the range of 1 to 8. If **S** is a word device, **n** is in the range of 1 to 4.

Example 1

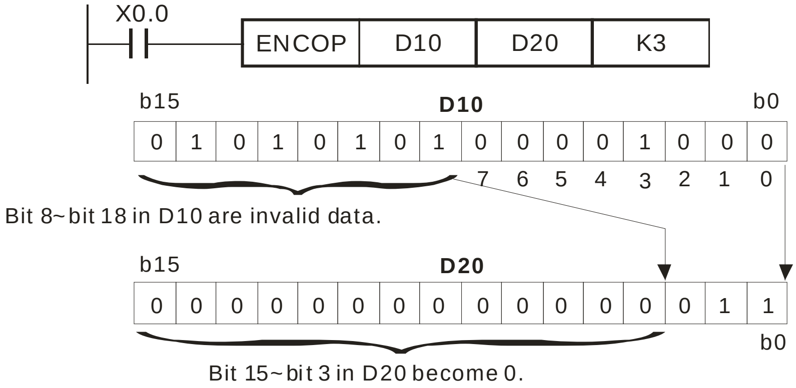
- When **S** is a bit device, **n** is in the range of 1 to 8. If **n** is 0, or greater than 8, an error will occur.
- If **n** is 8, the maximum number of bits which can be decoded is $2^8$=256.
- When X0.0 is turned from OFF to ON, the instruction ENCOP encodes the 8 bits in M0~M7 as the low 3 bits in D0, and b15~b3 in D0 become 0.
- After the instruction ENCOP is executed, X0.0 will be OFF, and the data in **D** will remain unchanged.




Bit 15~bit 3 in D0 become 0.

**Example 2**

- When **S** is a word device, **n** is in the range of 1 to 4. If **n** is 0, or larger than 4, an error will occur.
- If **n** is 4, the maximum number of bits which can be decoded is $2^4$=16.
- When X0.0 is turned from OFF to ON, the instruction ENCOP encodes the 8 bits in D10 as the low 3 bits in D20, and b15~b3 in D20 become 0.
- After the instruction ENCOP is executed, X0.0 will be OFF, and the data in **D** will remain unchanged.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 43 | D | SUM | P | **S**、**D** | Number of bits which are ON |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | | | ● | ● | ● | ● | ○ | ○ |

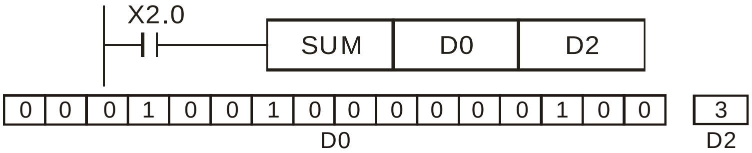
| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (5 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **S**: Source device; **D**: Destination device
- The number of bits which are 1 in **S** is stored in **D**.
- If the bits in **S** are 0, a zero flag will be ON.
- If the 32-bit instruction is used, **D** will occupy two registers.
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K1M0 and K4M16 (decimal numeral system).
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.

## Example

- When X2.0 is ON, the number of bits which are 1 in D0 is stored in D2.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 44 | D | BON | P | **S, D, n** | Checking the state of a bit |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | ● | ● | ● | | | | | | | | | | | |
| **n** | | | | | | | | | | ● | ● | ● | ● | ○ | ○ |

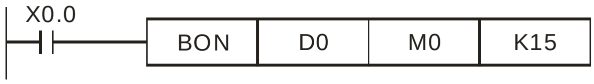
| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (8 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **S**: Source device; **D**: Device in which a check result is stored; **n**: Bit whose state is judged
- The state of the **n**th bit in **S** is checked, and the result is stored in **D**.
- 16-bit instruction: **n**=0~15; 32-bit instruction: **n**=0~31
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K1M0 and K4M16 (decimal numeral system).

## Example

- If the 15th bit in D0 is 1 when X0.0 is ON, M0 will be ON. If the 15th bit in D0 is 0 when X0.0 is ON, M0 will be OFF.
- When X0.0 is turned OFF, the state of M0 remains unchanged.




b15 … b0

| b15 | | | | | | | | | | | | | | | b0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | M0=Off |

D0

| b15 | | | | | | | | | | | | | | | b0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | M0=On |

D0

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 45 | D | MEAN | P | **S, D, n** | Mean |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | ● | ● | ● | | | | | | | | | | | |
| **n** | | | | | | | | | | ● | ● | ● | ● | ○ | ○ |

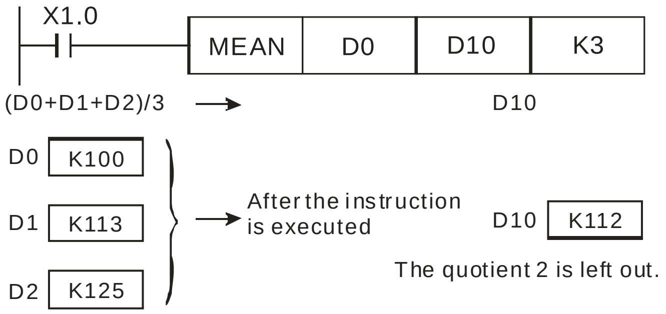
| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (8 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **S**: Initial device; **D**: Device in which a mean is stored; **n**: Number of devices
- After the values in the **n** devices starting from **S** are added up, the mean of the sum is stored in **D**.
- If a remainder appears in a calculation, it will be left out.
- If **S** is not in a valid range, only the devices in the valid range will be processed.
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K1M0 and K4S16 (decimal numeral system).
- If **n** is not in the range of 1 to 64, an operation error will occur.
- **n**=1~64

## Example

- When X1.0 is ON, the values in the three registers starting from D0 are added up. After the values are added up, the sum will be divided by 3. The quotient is stored in D10, and the remainder is left out.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 46 | | ANS | P | **S, m, D** | Driving an annunciator |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **m** | | ● | ● | ● | | | | | | | | | | | |
| **D** | | | | | | | | | | ● | ● | ● | ● | ○ | ○ |

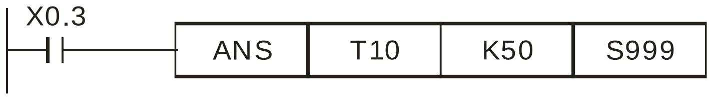
| Pulse instruction | 16-bit instruction (7steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

Explanation

- **S**: Timer; **m**: Time; **D**: Annunciator
- The instruction ANS is used to drive an annunciator.
- **S**: T0~T183
  **m**: K1~K32,767 (Unit: 100 ms)
  **D**: S912~S1023
  See the explanation of ANR for more information.
- The instruction supports V devices and Z devices. (If the 16-bit instructions used, Z devices can not be used. If the 32-bit instruction is used, V devices can not be used.)

Example

- If X0.3 is ON for more than 5 seconds, the annunciator S999 will be ON. Even if X0.3 is turned OFF, S999 will still be ON. (However, T10 will be reset to OFF, and the value of T10 will be 0.)



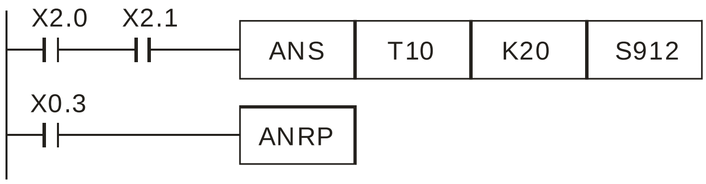
| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 47 | | ANR | P | – | Resetting an annunciator |

| Pulse instruction | 16-bit instruction (1 step) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

**Explanation**

**Example**

- The instruction ANR is used to reset an annunciator.
- If more than one annunciator is ON simultaneously, the annunciator whose number is smallest will be reset.
- Generally, the pulse instruction ANRP is used.
- If X2.0 and X2.1 are ON for more than 2 seconds, the annunciator S912 will be ON. If X2.0 and D2.1 are turned OFF, S912 will still be ON, T10 will be reset to OFF, and the value of T10 will be 0.
- If X2.0 and X2.1 are not ON for 2 seconds, the value of T10 will become 0.
- When X0.3 is turned from OFF to ON, the annunciator whose number is smallest in the annunciators which are driven is reset.
- When X0.3 is turned from OFF to ON again, the next annunciator whose number is smallest in the annunciators which are driven is reset.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 48 | D | SQR | P | **S, D** | Square root of a binary value |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **S**: Source device; **D**: Device in which a result is stored
- The square root of the value in **S** is calculated, and the result is stored in **D**.
- The value in **S** can only be a positive value. If the value in **S** is a negative value, an error will occur, and the instruction will not be executed.
- The value stored in **D** is an integer. The fractional part of a square root calculated is dropped. If the fractional part of a square root calculated is dropped, SM601 will be ON.
- If the value in **D** is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

## Example

- When X2.0 is ON, the square root of the value in D0 is calculated, and the result is stored in D12.


$$\sqrt{D0} \rightarrow D12$$

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 49 | D | FLT | P | **S, D** | Converting a binary integer into a binary floating-point value |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

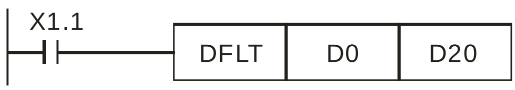
5

Explanation

- **S**: Source device; **D**: Conversion result
- The instruction is used to convert a binary integer into a binary floating-point value.
  1. If the absolute value of the converesion result is greater than the maximum floating-point value avaliable, a carry flag will be ON.
  2. If absolute value of the conversion result is less than the mimum floating-point vlaue available, a borrow flag will be ON.
  3. If the conversion result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

Example

- When X1.1 is ON, the binary integer in (D1, D0) is converted into a binary floating-point value, and the conversion result is stored in (D21, D20).
- Suppose the value in the 32-bit register (D1, D0) is K100,000. When X1.1 is ON, K100,000 is converted into the 32-bit floating-point number 16#4735000, and 16#4735000 is stored in the 32-bit register (D21, D20).



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 50 | | REF | P | **D, n** | Refreshing the states of I/O devices |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **D** | ○ | ○ | | | | | | | | | | | | | |
| **n** | | | | | ○ | ○ | | | | | | | | | |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

## Explanation

- **D**: Initial I/O device whose state is refreshed; **n**: Number of I/O devices whose states are refreshed
- The states of I/O devices are not refreshed until the instruction END is executed. When the scan of a program starts, the states of external inputs are read, and stored in the input memory. After the instruction END is executed, the contents of the output memory will be sent to output terminals. Therefore, users can use this instruction when they need the latest I/O data in an operation process.
- **D** must be an I/O device whose number ends with 0, e.g. X0.0, X1.0, Y0.0 or Y1.0. The instruction can not be used to refresh the I/O devices in a digital extension module.
- **D** must be an I/O device in a PLC.
  - If **D** is X0.0 and **n** is less than or equal to 8, the states of X0.0~X0.7 will be refreshed. If **n** is greater than 8, the states of the input devices and the states of the output devices in the motion control module used will be refreshed.
  - If **D** is Y0.0, and **n** is less than or equal to 8, the states of Y0.0~Y0.7 will be refreshed. If **n** is greater than 8, the states of the input devices and the states of the output devices in the motion control module used will be refreshed.
- **n** is in the range of 4 to the number of I/O devices in the motion control module used, and is a multiple of 4.

## Example 1

- When X0.0 is ON, the AH500 motion control module reads the states of X0.0~X0.7 immediately. The input signals are refreshed without any delay.


## Example 2

- When X0.0 is ON, the states of Y0.0~Y0.7 are sent to output terminals. The output signals are refreshed immediately without the need to wait for the execution of the instruction END.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 61 | D | SER | P | **$S_1$, $S_2$, D, n** | Searching data |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | | ● | ● | ● | ● | ● | ● | | |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | | |
| **n** | | | | | ○ | ○ | | | | ○ | ○ | | | | |

| Pulse instruction | 16-bit instruction (9 steps) | 32-bit instruction (11 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

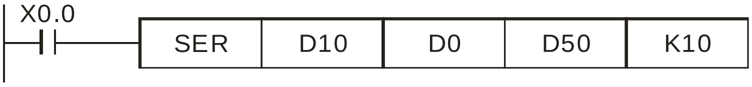


Explanation

- **$S_1$**: Initial device involved in a comparison; **$S_2$**: Value which is compared; **D**: Initial device in which a comparison result is stored (5 consecutive devices are occupied.); **n**: Number of values
- **$S_1$** is the initial register involved in a comparison, and **n** is the number of values which are compared. The values in the **n** registers starting from **$S_1$** are compared with the value in **$S_2$**, and the comparison results are stored in the five registers starting from **D**.
- If the 32-bit instruction is used, **$S_1$**, **$S_2$**, **D**, and **n** will be 32-bit registers.
- 16-bit instruction: **n**=1~256; **n**=1~128 (32-bit instruction)
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K1M0 and K4S16 (decimal numeral system).

Example

- 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. If none of the values in D10~D19 are equal to the value in D0, the values in D50~D52 will be 0.
- A comparison is based on algebra (-10 < 2).
- The number of the minimum value is stored in D53, and the number of the maximum value is stored in D54. If there is more than one minimum value/maximum value, the number which is the biggest will be stored.



| $S_1$ | Value | Value which is compared | 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 |

n

| D | Value | Description |
|---|---|---|
| D50 | 4 | Number of values which are equal to the value in D0 |
| D51 | 1 | Number of the first value which is equal to the value in D0 |
| D52 | 8 | Number of the last value which is equal to the value in D0 |
| D53 | 7 | Number of the minimum value |
| D54 | 9 | Number of the maximum value |

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 66 | | ALT | P | **D** | Alternating between ON and OFF |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **D** | ● | ● | ● | | | | | | | | | | | | |

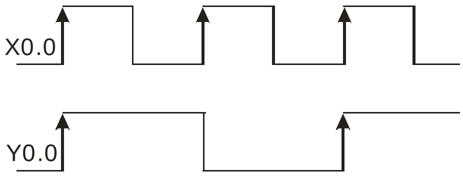
| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

Explanation

- **D**: Destination device
- When the instruction ALT is executed, the state of **D** alternates between ON and OFF.
- Generally, the pulse instruction ALTP is used.

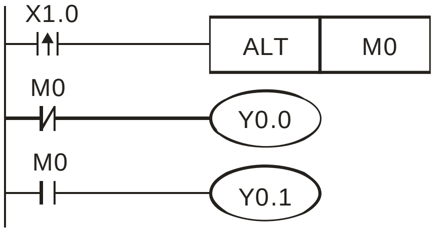
Example 1

- When X0.0 is turned from OFF to ON for the first time, Y0.0 is ON. When X0.0 is turned from OFF to ON for the second time, Y0.0 is OFF.

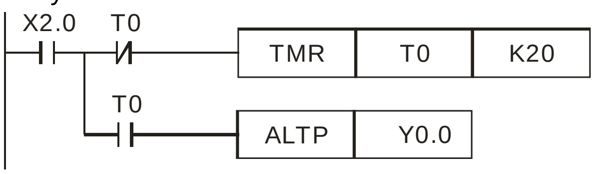

Example 2

- In the beginning, M0 is OFF, and therefore Y0.0 is ON, and Y0.1 is OFF. When X1.0 is turned from OFF to ON for the first time, M0 is ON. Therefore, Y0.1 is ON, and Y0.0 is OFF. When X1.0 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 X2.0 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 |
|---|---|---|---|---|---|
| 78 | D | FROM | P | **$m_1$, $m_2$, D, n** | Reading data from a control register in a special module |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$m_1$** | | | | | ○ | ○ | | | | ● | ● | ● | ● | ○ | ○ |
| **$m_2$** | | | | | ○ | ○ | | | | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | | | ● | ● | ● | ● | | |
| **n** | | | | | ○ | ○ | | | | ● | ● | ● | ● | ○ | ○ |

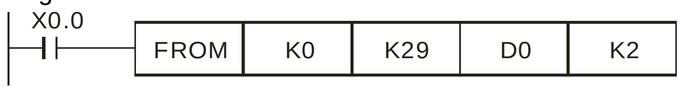
| Pulse instruction | 16-bit instruction (9 steps) | 32-bit instruction (12 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **$m_1$**: Special module number (**$m_1$** is in the range of 0 to 255.); **$m_2$**: Control register number (**$m_2$** is in the range of 0 to 499.); **D**: Device in which the data read will be stored; **n**: Quantity of data which will be read (16-bit instruction: 1~(500-**$m_2$**); 32-bit instruction: 1~(500-**$m_2$**)/2
- A motion control module can read the data in a control register in a special module by means of the instruction.

## Example

- The value in CR#29 in special module 0 is read, and then stored in D0 in the motion control module. The value in CR#30 in special module 0 is read, and then stored in D1 in the motion control module. The two values are read at the same time.
- When X0.0 is ON, the instruciton is executed. When X0.0 is turned OFF, the instruction is not executed, and the values which are read remain unchanged.




5

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 79 | D | TO | P | **$m_1$, $m_2$, S, n** | Writing data into a control register in a special module |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$m_1$** | | | | | ○ | ○ | | | | ● | ● | ● | ● | ○ | ○ |
| **$m_2$** | | | | | ○ | ○ | | | | ● | ● | ● | ● | ○ | ○ |
| **S** | | | | | ○ | ○ | | | | ● | ● | ● | ● | ○ | ○ |
| **n** | | | | | ○ | ○ | | | | ● | ● | ● | ● | ○ | ○ |

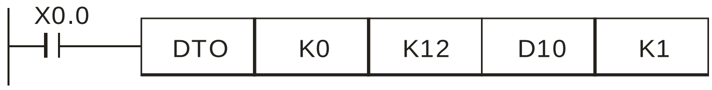
| Pulse instruction | 16-bit instruction (9 steps) | 32-bit instruction (13 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

Explanation

- **$m_1$**: Special module number (**$m_1$** is in the range of 0 to 255.); **$m_2$**: Control register number (**$m_2$** is in the range of 0 to 499.); **D**: Data which will be written into a control register; **n**: Quantity of data which will be written (16-bit instruction: 1~(500-**$m_2$**); 32-bit instruction: 1~(500-**$m_2$**)/2
- A motion control module can write data into a control register in a special module by means of the instruction.

Example

- The 32-bit instruction DTO is used. The value in (D11, D10) is written into (CR#13, CR#12) in special module 0. One value is written at a time.
- When X0.0 is ON, the instruction is executed. When X0.0 is turned OFF, the instruction is not executed, and the value which is written remains unchanged.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 87 | D | ABS | P | **D** | Absolute value |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

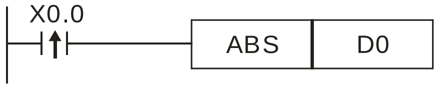
| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction (3 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **D**: Device whose absolute value will be gotten
- When the instruction ABS is executed, the absolute value of the value in **D** is gotten
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K4M0 and K4S16 (decimal numeral system).
- Generally, the pulse instructions ABSP and DABSP are used.

## Example

- When X0.0 is turned from OFF to ON, the absolute value of the value in D0 is gotten.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 110 | D | ECMP | P | **$S_1$, $S_2$, D** | Comparing binary floating-point numbers |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | ● | ● | ● | | | | | | | | | | | |

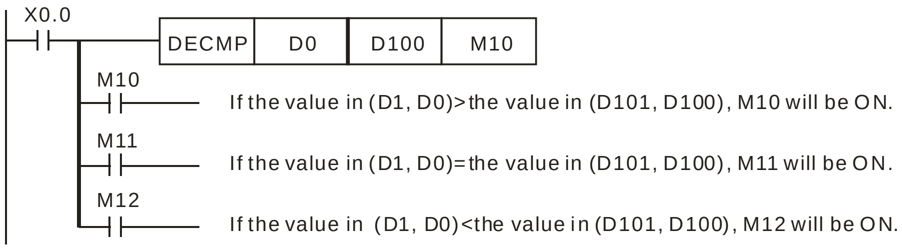
| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **$S_1$**: Binary floating-point number 1; **$S_2$**: Binary floating-point number 2; **D**: Comparison result (D occupies three consecutive devices.)
- The instruction is used to compare the binary floating-point value in **$S_1$** with that in **$S_2$**. The comparison result is stored in **D**.
- If **$S_1$** is a floating-point number, the instruction will be used to compare the floating-point number with the binary floating-point value in **$S_2$**. If **$S_2$** is a floating-point number, the instruction will be used to compare the binary floating-point value in **$S_1$** with the floating-point number.
- F represents a floating-point number. There is a decimal point in a floating-point number.

## Example

- If the operand **D** is M10, M10, M11, and M12 will be occupied automatically.
- When X0.0 is ON, the instruction DECMP is executed, and M10, M11, or M12 is ON. When X0.0 is OFF, the execution of the instruction DECMP stops, and the states of M10, M11, and M12 remain unchanged.
- If users want to get the result that the value in (D1, D0)≧the value in (D101, D100), they have to connect M10 and M11 in series. If users want to get the result that the value in (D1, D0)≦the value in (D101, D100), they have to connect M11 and M12 in series. If users want to get the result that the value in (D1, D0)≠the value in (D101, D100), they have to connect M10, M11, and M12 in series.
- If users want to reset M10, M11, or M12, they can use the instruction RST or ZRST.




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 111 | D | EZCP | P | **$S_1$, $S_2$, S, D** | Binary floating-point zonal comparison |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | ● | ● | ● | | | | | | | | | | | |

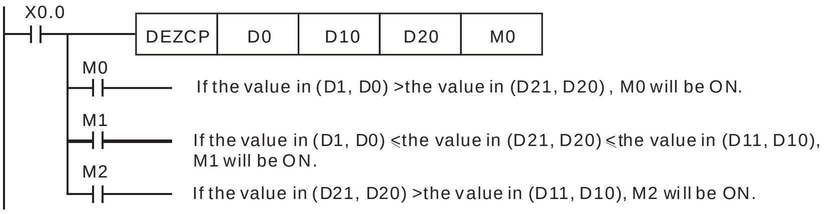
| Pulse instruction | 16-bit instruction | 32-bit instruction (12 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **$S_1$**: Minimum binary floating-point value; **$S_2$**: Maximum binary floating-point value; **S**: Binary floating-point value; **D**: Comparison result (D occupies three consecutive devices.)
- The instruction is used to compare the binary floating-point value in **S** with that in **$S_1$**, and compare the binary floating-point value in **S** with that in **$S_2$**. The comparison result is stored in **D**.
- If **$S_1$** is a floating-point number, the instruction will be used to compare the floating-point number with the binary floating-point value in **$S_2$**. If **$S_2$** is a floating-point number, the instruction will be used to compare the binary floating-point value in **$S_1$** with the floating-point number.
- If the binary floating-point value in **$S_1$** is greater than that in **$S_2$**, the binary floating-point value in **$S_1$** will be taken as the maximum/minimum value during the execution of the instruction EZCP.
- If the operand **D** is M0, M0, M1, and M2 will be occupied automatically.

## Example

- When X0.0 is ON, the instruction DEZCP is executed, and M0, M1, or M2 is ON. When X0.0 is OFF, the execution of the instruction DEZCP stops, and the states of M0, M1, and M2 remain unchanged.
- If users want to reset M0, M1, or M2, they can use the instruction RST or ZRST.




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 112 | D | MOVR | P | **S, D** | Transferring a floating-point value |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | | | | | | |
| **D** | | | | | | | | | | ● | ● | ● | ● | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

Explanation

- **S**: Source; **D**: Destination
- The operand **S** can be a floating-point number.
- When the instruction is executed, the value in **S** is transferred to **D**. When the instruction is not executed, the value in **D** is unchanged.
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K1M0 and K4S16 (decimal numeral system)

Example

- When X0.0 is OFF, the value in (D11, D10) is unchanged. When X0.0 is ON, the value F1.2 is transferred to the data register (D11, D10).

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 116 | D | RAD | P | **S, D** | Converting a degree to a radian |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

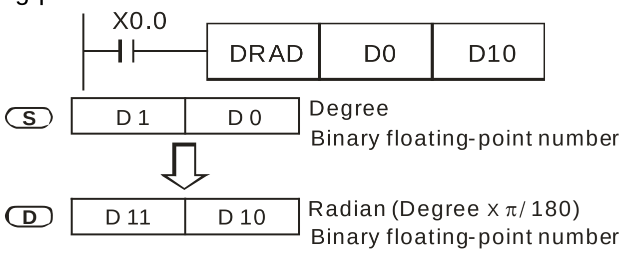
## Explanation

- **S**: Source (degree); **D**: Conversion result (radian)
- The equation below is used to convert a degree into a radian.

  Radian = Degree×(π/180)
- If the absolute value of a conversion result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of a conversion reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If a converseion result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

## Example

- When X0.0 is ON, the degree in (D1, D0) is converted into a radian, and the conversion result is stored in (D11, D10). The radian in (D11, D10) is a floating-point number.




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 117 | D | DEG | P | **S, D** | Converting a radian to a degree |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

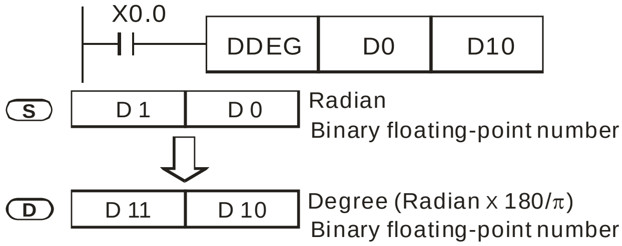
## Explanation

- **S**: Source (radian); **D**: Conversion result (degree)
- The equation below is used to convert a radian into a degree.

  Degree = Radian×(180/π)
- If the absolute value of a conversion result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of a conversion reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If a converseion result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

## Example

- When X0.0 is ON, the radian in (D1, D0) is converted into a degree, and the conversion result is stored in (D11, D10). The degree in (D11, D10) is a floating-point number.




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 120 | D | EADD | P | **$S_1$, $S_2$, D** | Binary floating-point addition |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

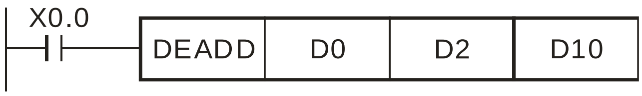
| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

Explanation

- **$S_1$**: Augend; **$S_2$**: Addend; **D**: Sum
- The binary floating-point value in **$S_2$** is added to the binary floating-point value in **$S_1$**, and the sum is stored in **D**.
- If **$S_1$** is a floating-point value, the instruction will be used to add the binary floating-point value in **$S_2$** to the floating-point value. If **$S_2$** is a floating-point value, the instruction will be used to add the floating-point value to the binary floating-point value in **$S_1$**.
- **$S_1$** and **$S_2$** can be the same register. If the instruction DEADD is used under the circumstances, the value in the register is added to itself whenever the conditional contact is ON in a scan cycle. Generally, the pulse instruction DEADDP is used.
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

Example 1

- When X0.0 is ON, the binary floating-point value in (D3, D2) is added to the binary floating-point value in (D1, D0), and the sum is stored in (D11, D10).




Example 2

- When X0.2 is ON, F1234.0 is added to the binary floating-point value in (D11, D10), and the sum is stored in (D21, D20).


Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 121 | D | ESUB | P | $S_1$, $S_2$, D | Binary floating-point subtraction |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | | | | | ○ | | | ● | ● | | | | |
| $S_2$ | | | | | | | ○ | | | ● | ● | | | | |
| D | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

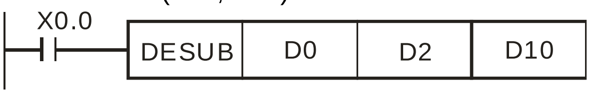


Explanation

- $S_1$: Minuend; $S_2$: Subtrahend; **D**: Difference
- The binary floating-point value in $S_2$ is subtracted from the binary floating-point value in $S_1$, and the difference is stored in **D**.
- If $S_1$ is a floating-point value, the instruction will be used to subtract the binary floating-point value in $S_2$ from the floating-point value. If $S_2$ is a floating-point value, the instruction will be used to subtract the floating-point value from the binary floating-point value in $S_1$.
- $S_1$ and $S_2$ can be the same register. If the instruction DESUB is used under the circumstances, the value in the register is subtracted from itself whenever the conditional contact is ON in a scan cycle. Generally, the pulse instruction DESUBP is used.
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

Example 1

- When X0.0 is ON, the binary floating-point value in (D3, D2) is subtracted from the binary floating-point value in (D1, D0), and the difference is stored in (D11, D10).




Example 2

- When X0.2 is ON, the binary floating-point value in (D1, D0) is subtracted from F1234.0, and the difference is stored in (D11, D10).


Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 122 | D | EMUL | P | **$S_1$, $S_2$, D** | Binary floating-point multiplication |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

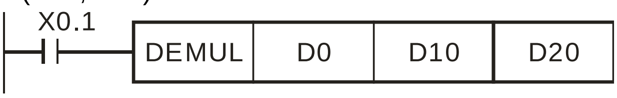
| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **$S_1$**: Multiplicand; **$S_2$**: Multiplier; **D**: Product
- The binary floating-point value in **$S_1$** is multiplied by the binary floating-point value in **$S_2$**, and the product is stored in **D**.
- If **$S_1$** is a floating-point value, the instruction will be used to multiply the floating-point value by the binary floating-point value in **$S_2$**. If **$S_2$** is a floating-point value, the instruction will be used to multiply the binary floating-point value in **$S_1$** by the floating-point value.
- **$S_1$** and **$S_2$** can be the same register. If the instruction DEMUL is used under the circumstances, the value in the register is multiplied by itself whenever the conditional contact is ON in a scan cycle. Generally, the pulse instruction DEMULP is used.
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

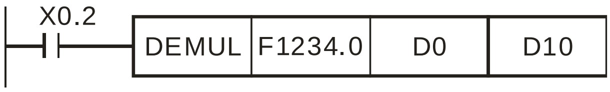
## Example 1

- When X0.1 is ON, the binary floating-point value in (D1, D0) is multiplied by the binary floating-point value in (D11, D10), and the product is stored in (D21, D20).




## Example 2

- When X0.2 is ON, F1234.0 is multiplied by the binary floating-point value in (D1, D0), and the product is stored in (D11, D10).




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 123 | D | EDIV | P | **$S_1$, $S_2$, D** | Binary floating-point division |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

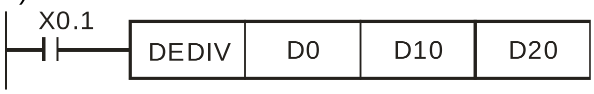
| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **$S_1$**: Dividend; **$S_2$**: Divisor; **D**: Quotient and remainder
- The binary floating-point value in **$S_1$** is divided by the binary floating-point value in **$S_2$**, and the quotient is stored in **D**.
- If **$S_1$** is a floating-point value, the instruction will be used to divide the floating-point value by the binary floating-point value in **$S_2$**. If **$S_2$** is a floating-point value, the instruction will be used to divide the binary floating-point value in **$S_1$** by the floating-point value.
- If the value in **$S_2$** is 0, an operation error will occur, the instruciton will not be executed, an operation error flag will be ON, and the error code 16#0E19 will appear.
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.
- SM1049 is the operation error flag in an Ox motion subroutine, and SM953 is the operation error flag in O100.

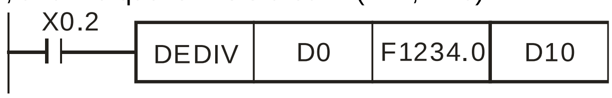
## Example 1

- When X0.1 is ON, the binary floating-point value in (D1, D0) is divided by the binary floating-point value in (D11, D10), and the quotient is stored in (D21, D20).




## Example 2

- When X0.2 is ON, the binary floating-point value in (D1, D0) is divided by F1234.0, and the quotient is stored in (D11, D10).




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 124 | D | EXP | P | **S, D** | Exponent of a binary floating-point number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

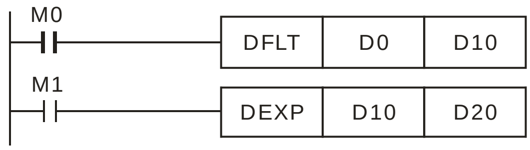
| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **S**: Source device; **D**: Device in which an operation result is stored
- EXP$^{[D+1,\ D]}$=[**S**+1，**S**]
- e is a base (e=2.71828), and **S** is an exponent.
- The value in **S** can be a positive value or a negative value. **D** must be a 32-bit register, and the value in **S** must be a floating-point value.
- The value in **D** is e$^{S}$. (e is 2.71828, and **S** represents a source value.)
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

## Example

- When M0 is ON, the value in (D1, D0) is converted into a floating-point value, and the conversion result is stored in (D11, D10).
- When M1 is ON, the exponentiation with the value in (D11, D10) as an exponent is performed. The result is a floating-point number, and is stored in (D21, D20).




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 125 | D | LN | P | **S, D** | Natural logarithm of a binary floating-point number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

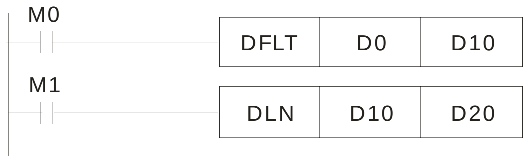
| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **S**: Source device; **D**: Device in which an operation result is stored
- The natural logarithm of the value in **S** is calculated
  Ln[**S**+1, **S**]=[**D**+1, **D**]
- The value in **S** can only be a positive value. **D** must be a 32-bit register, and the value in **S** must be a floating-point value.
- If the value in **S** is not a positive value, an operation error will occur, the instruciton will not be executed, an operation error flag will be ON, and the error code 16#0E19 will appear.
- $e^D$=**S**.→The value in **D**=ln**S** (**S**: Source device)
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.
- SM1049 is the operation error flag in an Ox motion subroutine, and SM953 is the operation error flag in O100.

## Example

- When M0 is ON, the value in (D1, D0) is converted into a binary floating-point value, and the conversion result is stored in (D11, D10).
- When M1 is ON, the natural logarithm of the floating-point value in (D11, D10) is calculated, and the operation result is stored in (D21, D20).




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 126 | D | LOG | P | **$S_1$, $S_2$, D** | Logarithm of a binary floating-point number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

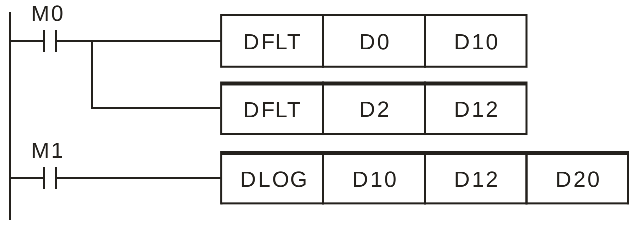
| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **$S_1$**: Device in which the base is stored; **$S_2$**: Source device; **D**: Device in which an operation result is stored
- The logarithm of the value in **$S_2$** with respect to the value in **$S_1$** is calculated, and the operation result is stored in **D**.
- The values in **$S_1$** and **$S_2$** can only be positive values. **D** must be a 32-bit register, and the values in **$S_1$** and **$S_2$** must be floating-point values.
- $S_1^{D}=S_2 \rightarrow D=Log_{S1}S_2$
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

## Example

- When M0 is ON, the values in (D1, D0) and (D3, D2) are converted into binary floating-point values, and the conversion results are stored in (D11, D10) and (D13, D12) respectively.
- When M1 is ON, the logarithm of the binary floating-point value in (D13, D12) with respect to the binary floating-point value in (D11, D10) is calculated, and the operation result is stored in (D21, D20).




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 127 | D | ESQR | P | **S, D** | Square root of a binary floating-point number |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **S**: Source device; **D**: Device in which a result is stored
- The square root of the binary floating-point value in **S** is calculated, and the result is stored in **D**.
- If **S** is a floating-point value, the instruction will be used to calculate the floating-point value.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1049 is the operation error flag in an Ox motion subroutine, and SM953 is the operation error flag in O100.
- If the value in **S** is not a positive value, an operation error will occur, the instruciton will not be executed, an operation error flag will be ON, and the error code 16#0E19 will appear.

## Example 1

- When X0.0 is ON, the square root of the binary floating-point value in (D1, D0) is calculated, and the result is stored in (D11, D10).


$$\sqrt{(D1, D0)} \longrightarrow (D11, D10)$$

Binary floating-point number — Binary floating-point number

## Example 2

- When X0.2 is ON, the square root of F1234.0 is calculated, and the result is stored in (D11, D10).


## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 128 | D | POW | P | **$S_1$, $S_2$, D** | Power of a floating-point number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

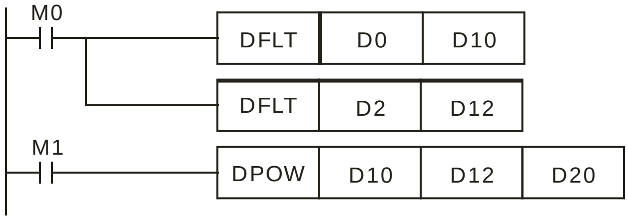
| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **$S_1$**: Device in which a base is stored; **$S_2$**: Device in which a power is stored; **D**: Device in which the operation result is stored
- The binary floating-point value in **$S_1$** is raised to the power of the value in **$S_2$**, and the operation result is stored in **D**.
  **D**=POW[**$S_1$**+1, **$S_1$**]$^{[S2+1,\ S2]}$
- The value in **$S_1$** can only be a positive value, whereas the value in **$S_2$** can be a positive value or a negative value. **D** must be a 32-bit register, and the values in **$S_1$** and **$S_2$** must be floating-point values.
- If the values in **$S_1$** and **$S_2$** are invalid, an operation error will occur, the instruciton will not be executed, an operation error flag will be ON, and the error code 16#0E19 will appear.
- If the absolute value of an operation result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an operation reusIt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.
- SM1049 is the operation error flag in an Ox motion subroutine, and SM953 is the operation error flag in O100.

## Example

- When M0 is ON, the values in (D1, D0) and (D3, D2) are converted into binary floating-point values, and the conversion results are stored in (D11, D10) and (D13, D12) respectively.
- When M1 is ON, the binary floating-point value in (D11, D10) is raised to the power of the binary floating-point value in (D13, D12), and the operation result is stored in (D21, D20).




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 129 | D | INT | P | **S, D** | Converting a binary floating-point number into a binary integer |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

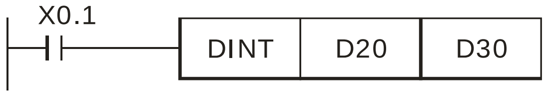
| Pulse instruction | 16-bit instruction | 32-bit instruction (5 steps) |
|---|---|---|
| ✓ | – | ✓ |

Explanation

- **S**: Source device; **D**: Conversion result
- The binary floating-point value in **S** is converted into a binary value. The integer part of the binary value is stored in **D**, and the fractional part of the binary value is dropped.
- The instruction is the opposite of API 49 DFLT.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.
- SM1049 is the operation error flag in an Ox motion subroutine, and SM953 is the operation error flag in O100.
- If a conversion result is 0, a zero flag will be ON.
  If the fractional part of a conversion result is dropped, a borrow flag will be ON.
  If a converesion result is not in the range of -2,147,483,648 to 2,147,483,647, a carry flag will be ON.

Example

- When X0.1 is ON, the binary floating-point value in (D21, D20) is converted into a binary value. The integer part of the binary value is stored in (D31, D30), and the fractional part of the binary value is dropped.



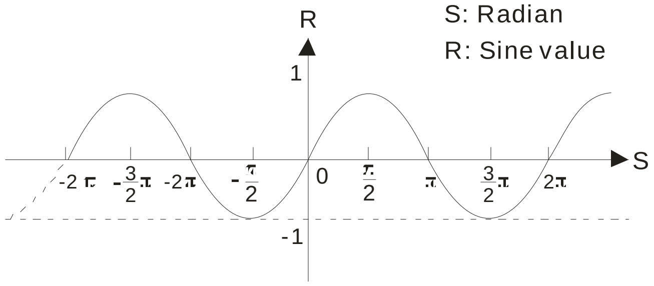
| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 130 | D | SIN | P | **S, D** | Sine of a binary floating-point number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

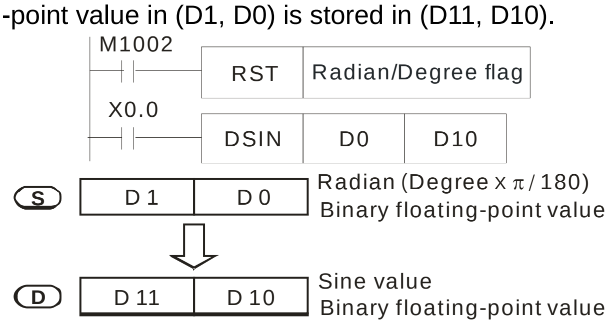
- **S**: Source value; **D**: Sine value
- Whether the source value in **S** is a radian or a degree depends on the state of a radian/degree flag.
- If a radian/degree flag is OFF, the source value in **S** is a radian.
  Radian=Degree×π/180.
- If a radian/degree flag is ON, the source value in **S** is a degree.
  ($0^{o}$≦Degree≦$360^{0}$)
- If an operation result is 0, a zero flag will be ON.
- The sine of the source value in **S** is stored in **D**.
  The relation between radians and sine values is shown below.




- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1049 is the operation error flag in an Ox motion subroutine, and SM953 is the operation error flag in O100.

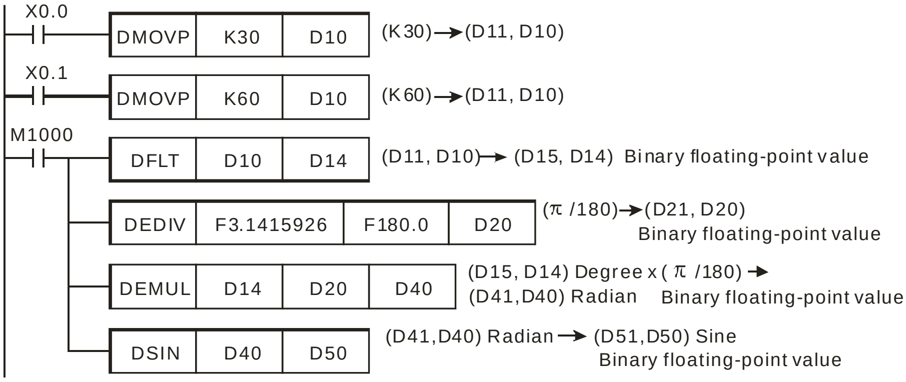
## Example 1

- A radian/degree flag is reset to OFF. The binary floating-point value in (D1, D0) is a radian. When X0.0 is ON, the sine of the binary floating-point value in (D1, D0) is stored in (D11, D10).



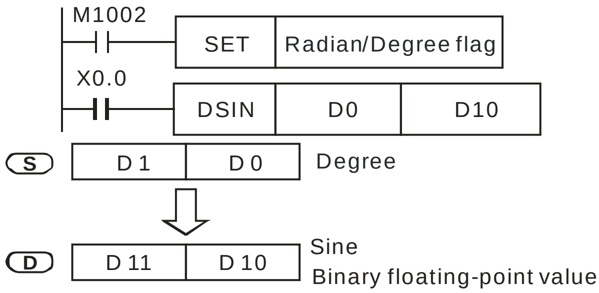
**Example 2**

- A radian/degree flag is OFF. A degree is set by means of X0.0 or X0.1. After the degree is converted into a radian, the sine of the radian will be calculated.




**Example 3**

- A radian/degree flag is set to ON. The value in (D1, D0) is a degree in the range of 0° to 360°. When X0.0 is ON, the sine of the value in (D1, D0) is stored in (D11, D10). The value in (D11, D10) is a binary floating-point value.




**Additional remark**

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

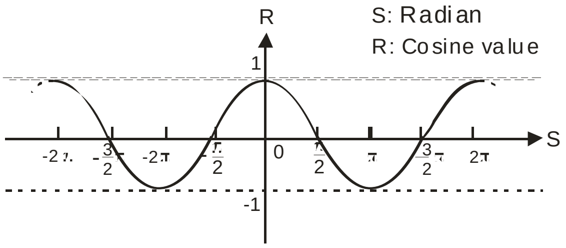
| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 131 | D | COS | P | **S, D** | Cosine of a binary floating-point number |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

Explanation

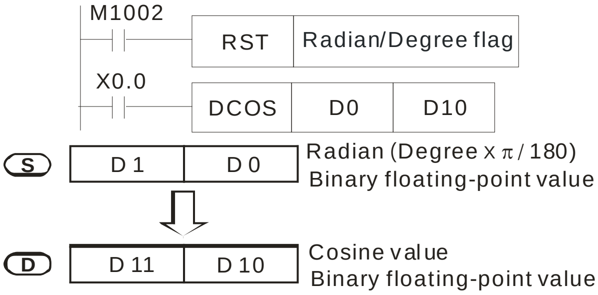
- **S**: Source value; **D**: Cosine value
- Whether the source value in **S** is a radian or a degree depends on the state of a radian/degree flag.
- If a radian/degree flag is OFF, the source value in **S** is a radian. Radian=Degree×π/180.
- If a radian/degree flag is ON, the source value in **S** is a degree. ($0^{o}$≦Degree≦$360^{0}$)
- If an operation result is 0, a zero flag will be ON.
- The cosine of the source value in **S** is stored in **D**. The relation between radians and cosine values is shown below.




- Radian/Degree flag: If a radian/degree flag is OFF, the source value in **S** is a radian. If a radian/degree flag is ON, the source value in **S** is a degree in the range of 0° to 360°.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1049 is the operation error flag in an Ox motion subroutine, and SM953 is the operation error flag in O100.

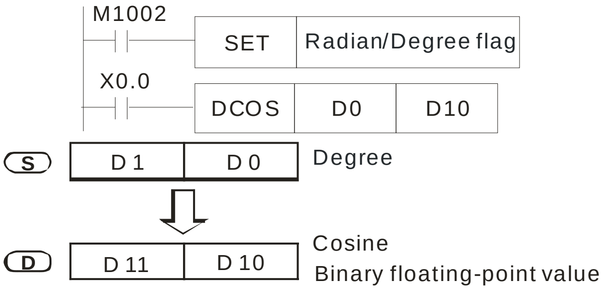
**Example 1**

- A radian/degree flag is reset to OFF. The binary floating-point value in (D1, D0) is a radian. When X0.0 is ON, the cosine of the binary floating-point value in (D1, D0) is stored in (D11, D10).




**Example 2**

- A radian/degree flag is set to ON. The value in (D1, D0) is a degree in the range of 0° to 360°. When X0.0 is ON, the cosine of the value in (D1, D0) is stored in (D11, D10). The value in (D11, D10) is a binary floating-point value.




5

**Additional remark**

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 132 | D | TAN | P | **S, D** | Tangent of a binary floating-point number |

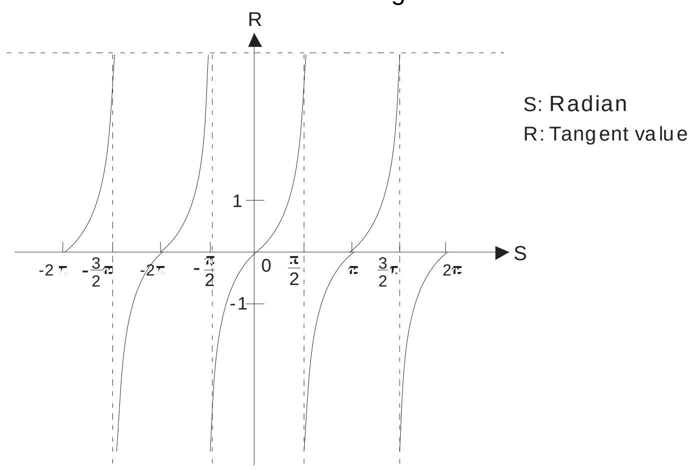
| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

5

## Explanation

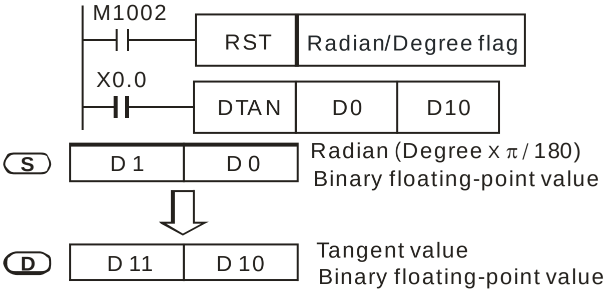
- **S**: Source value; **D**: Tangent value
- Whether the source value in **S** is a radian or a degree depends on the state of a radian/degree flag.
- If a radian/degree flag is OFF, the source value in **S** is a radian.
  Radian=Degree×π/180.
- If a radian/degree flag is ON, the source value in **S** is a degree.
  ($0^{o}$≦Degree≦$360^{0}$)
- If an operation result is 0, a zero flag will be ON.
- The tangent of the source value in **S** is stored in **D**.
  The relation between radians and tangent values is shown below.




- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1049 is the operation error flag in an Ox motion subroutine, and SM953 is the operation error flag in O100.

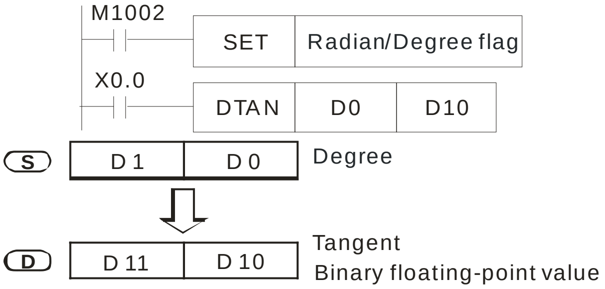
## Example 1

- A radian/degree flag is reset to OFF. The binary floating-point value in (D1, D0) is a radian. When X0.0 is ON, the tangent of the binary floating-point value in (D1, D0) is stored in (D11, D10).



**Example 2**

- A radian/degree flag is set to ON. The value in (D1, D0) is a degree in the range of 0° to 360°. When X0.0 is ON, the tangent of the value in (D1, D0) is stored in (D11, D10). The value in (D11, D10) is a binary floating-point value.




**Additional remark**

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

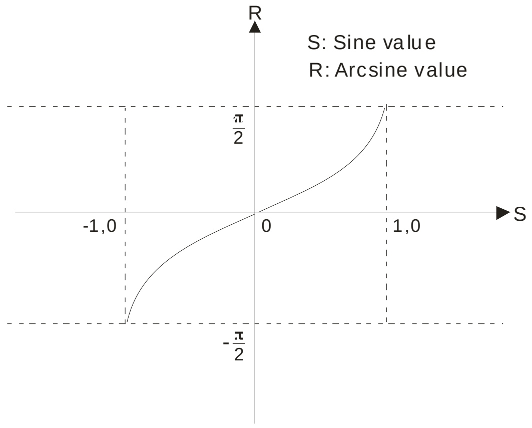
| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 133 | D | ASIN | P | **S, D** | Arcsine of a binary floating-point number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

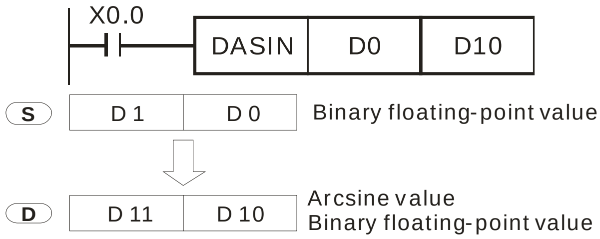
- **S**: Source value (binary floating-point value); **D**: Arcsine value
- Arcsine value=$\sin^{-1}$
  The relation between sine values and arcsine values is shown below.




- The decimal floating-point value into which the sine value in **S** is converted can only be in the range of -1.0 to +1.0. If it is not in the range, the instruction will not be executed, an operation error flag will be ON, and the error code 16#0E19 will appear.
- If a conversion result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1049 is the operation error flag in an Ox motion subroutine, and SM953 is the operation error flag in O100.

## Example

- When X0.0 is ON, the arcsine of the binary floating-point value in (D1, D0) is stored in (D11, D10).




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

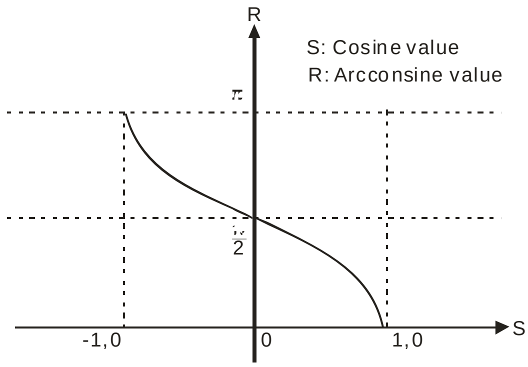
| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 134 | D | ACOS | P | **S, D** | Arccosine of a binary floating-point number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

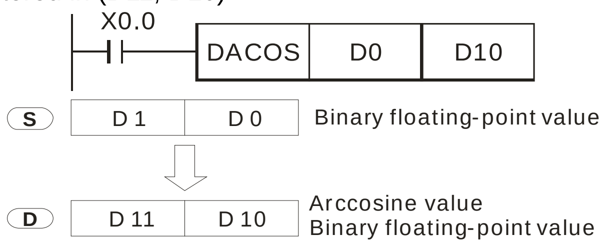
- **S**: Source value (binary floating-point value); **D**: Arccosine value
- Arccosine value=cos$^{-1}$
  The relation between cosine values and arccosine values is shown below.




- The decimal floating-point value into which the cosine value in **S** is converted can only be in the range of -1.0 to +1.0. If it is not in the range, the instruction will not be executed, an operation error flag will be ON, and the error code 16#0E19 will appear.
- If a conversion result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1049 is the operation error flag in an Ox motion subroutine, and SM953 is the operation error flag in O100.

## Example

- When X0.0 is ON, the arccosine of the binary floating-point value in (D1, D0) is stored in (D11, D10).




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

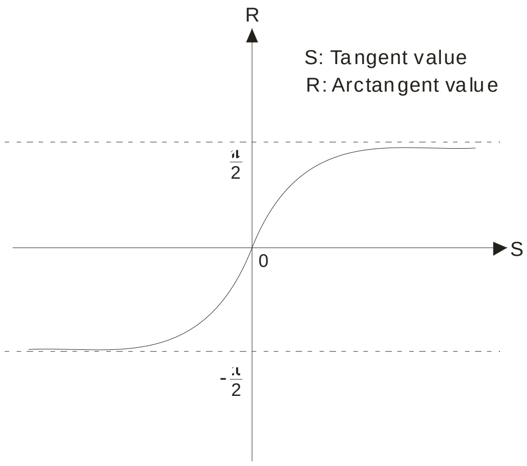
| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 135 | D | ATAN | P | **S, D** | Arctangent of a binary floating-point number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

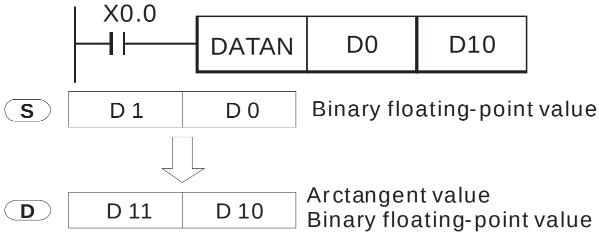
- **S**: Source value (binary floating-point value); **D**: Arctangent value
- Arctangent value=$\tan^{-1}$
  The relation between tangent values and arctangent values is shown below.




- If a conversion result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.

## Example

- When X0.0 is ON, the arctangent of the binary floating-point value in (D1, D0) is stored in (D11, D10).




## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 136 | D | SINH | P | **S, D** | Hyperbolic sine of a binary floating-point number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

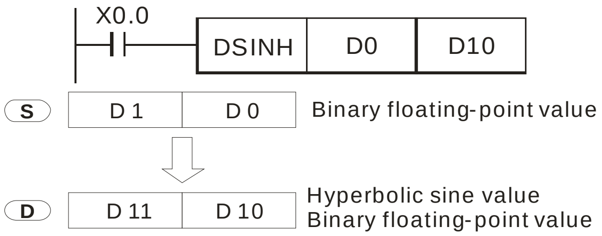
| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **S**: Source value (binary floating-point value); **D**: Hyperbolic sine value
- Hyperbolic sine value=$(e^{s}-e^{-s})/2$

## Example

- When X0.0 is ON, the hyperbolic sine of the binary floating-point number in (D1, D0) is stored in (D11, D10).




- If the absolute value of a conversion result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of a conversion reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If a converseion result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 137 | D | COSH | P | **S, D** | Hyperbolic cosine of a binary floating-point number |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

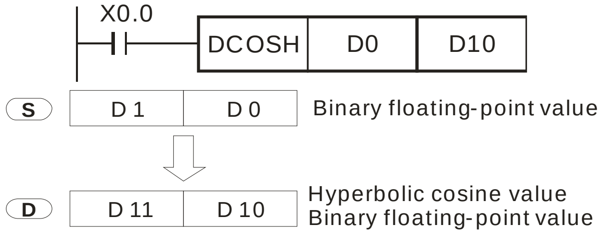
| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **S**: Source value (binary floating-point value); **D**: Hyperbolic cosine value
- Hyperbolic cosine value=$(e^{S}+e^{-S})/2$

## Example

- When X0.0 is ON, the hyperbolic cosine of the binary floating-point number in (D1, D0) is stored in (D11, D10).




- If the absolute value of a conversion result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of a conversion result is less than the minimum floating-point value available, a borrow flag will be ON.
- If a conversion result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

## Additional remark

- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 138 | D | TANH | P | **S, D** | Hyperbolic tangent of a binary floating-point number |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction | 32-bit instruction (6 steps) |
|---|---|---|
| ✓ | – | ✓ |

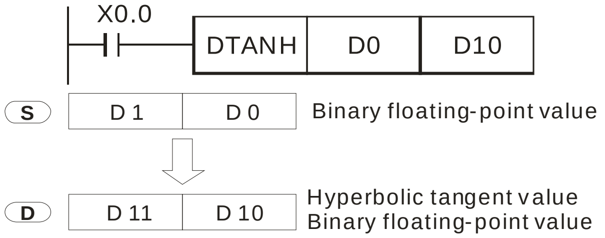


## Explanation

- **S**: Source value (binary floating-point value); **D**: Hyperbolic tangent value
- Hyperbolic tangent value=$(e^{S}-e^{-S})/(e^{S}+e^{-S})$

## Example

- When X0.0 is ON, the hyperbolic tangent of the binary floating-point number in (D1, D0) is stored in (D11, D10).




## Additional remark

- If the absolute value of a conversion result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of a conversion reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If a converseion result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.
- Please refer to section 5.3 for more information about performing operations on floating-point numbers.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 172 | D | ADDR | P | **$S_1$, $S_2$, D** | Floating-point addition |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

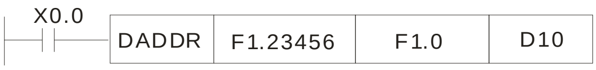
| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

Explanation

- **$S_1$**: Augend; **$S_2$**: Addend; **D**: Sum
- **$S_1$** and **$S_2$** can be floating-point values.
- **$S_1$** and **$S_2$** can be floating-point values, or data registers in which floating-point values are stored.
- If **$S_1$** and **$S_2$** are data registers in which floating-point values are stored, the function of API 172 DAADR is the same as the function of API 120 DEADD.
- The floating-point value in **$S_2$** is added to the floating-point value in **$S_1$**, and the sum is stored in **D**.
- **$S_1$** and **$S_2$** can be the same register. If the instruction DAADR is used under the circumstances, the value in the register is added to itself whenever the conditional contact is ON in a scan cycle. Generally, the pulse instruction DADDRP is used.
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

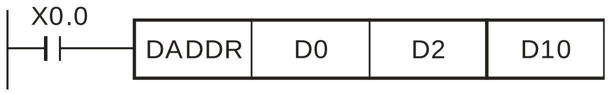
Example 1

- When X0.0 is ON, the floating-point value F1.0 is added to the floating-point value F1.23456, and the sum F2.23456 is stored in (D11, D10).




Example 2

- When X0.0 is ON, the floating-point value in (D3, D2) is added to the floating-point value in (D1, D0), and the sum is stored in (D11, D10).



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 173 | D | SUBR | P | **$S_1$, $S_2$, D** | Floating-point subtraction |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

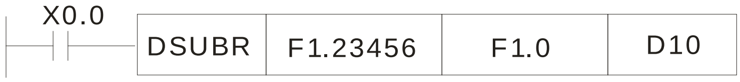
| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

Explanation

- **$S_1$**: Minuend; **$S_2$**: Subtrahend; **D**: Subtrahend
- **$S_1$** and **$S_2$** can be floating-point values.
- **$S_1$** and **$S_2$** can be floating-point values, or data registers in which floating-point values are stored.
- If **$S_1$** and **$S_2$** are data registers in which floating-point values are stored, the function of API 172 DSUBR is the same as the function of API 121 DESUB.
- The floating-point value in **$S_2$** is subtracted from the floating-point value in **$S_1$**, and the difference is stored in **D**.
- **$S_1$** and **$S_2$** can be the same register. If the instruction DSUBR is used under the circumstances, the value in the register is subtracted from itself whenever the conditional contact is ON in a scan cycle. Generally, the pulse instruction DSUBRP is used.
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

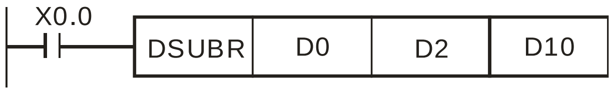
Example 1

- When X0.0 is ON, the floating-point value F1.0 is subtracted from the floating-point value F1.23456, and the difference F0.23456 is stored in (D11, D10).




Example 2

- When X0.0 is ON, the floating-point value in (D3, D2) is subtracted from the floating-point value in (D1, D0), and the difference is stored in (D11, D10).



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 174 | D | MULR | P | **$S_1$, $S_2$, D** | Floating-point multiplication |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

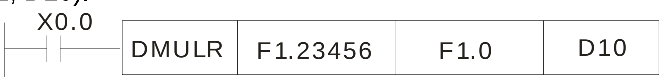
| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **$S_1$**: Multiplicand; **$S_2$**: Multiplier; **D**: Product
- **$S_1$** and **$S_2$** can be floating-point values.
- **$S_1$** and **$S_2$** can be floating-point values, or data registers in which floating-point values are stored.
- If **$S_1$** and **$S_2$** are data registers in which floating-point values are stored, the function of API 172 DMULR is the same as the function of API 122 DEMUL.
- The floating-point value in **$S_1$** is multiplied by the floating-point value in **$S_2$**, and the product is stored in **D**.
- **$S_1$** and **$S_2$** can be the same register. If the instruction DSUBR is used under the circumstances, the value in the register is multiplied by itself whenever the conditional contact is ON in a scan cycle. Generally, the pulse instruction DMULRP is used.
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

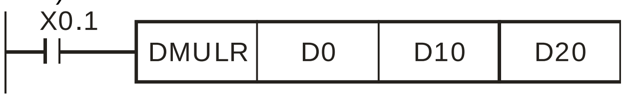
## Example 1

- When X0.0 is ON, the floating-point value F1.23456 is multiplied by the floating-point value F1.0, and the product F1.23456 is stored in (D11, D10).




## Example 2

- When X0.0 is ON, the floating-point value in (D1, D0) is multiplied by the floating-point value in (D11, D10), and the product is stored in (D21, D20).



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 175 | D | DIVR | P | **$S_1$, $S_2$, D** | Floating-point division |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | | | ○ | | | ● | ● | | | | |
| **$S_2$** | | | | | | | ○ | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

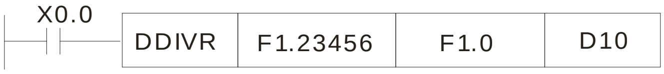
| Pulse instruction | 16-bit instruction | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | – | ✓ |

## Explanation

- **$S_1$**: Dividend; **$S_2$**: Divisor; **D**: Quotient
- **$S_1$** and **$S_2$** can be floating-point values.
- **$S_1$** and **$S_2$** can be floating-point values, or data registers in which floating-point values are stored.
- If **$S_1$** and **$S_2$** are data registers in which floating-point values are stored, the function of API 172 DDIVR is the same as the function of API 123 DEDIV.
- The floating-point value in **$S_1$** is divided by the floating-point value in **$S_2$**, and the product is stored in **D**.
- **$S_1$** and **$S_2$** can be the same register. If the instruction DSUBR is used under the circumstances, the value in the register is divided by itself whenever the conditional contact is ON in a scan cycle. Generally, the pulse instruction DDIVRP is used.
- If the absolute value of an oepration result is greater than the maximum floating-point value available, a carry flag will be ON.
- If the absolute value of an oepration reuslt is less than the minimum floating-point value available, a borrow flag will be ON.
- If an operation result is 0, a zero flag will be ON.
- SM1064 is the zero flag in an Ox motion subroutine, and SM968 is the zero flag in O100.
- SM1065 is the borrow flag in an Ox motion subroutine, and SM969 is the borrow flag in O100.
- SM1066 is the carry flag in an Ox motion subroutine, and SM970 is the carry flag in O100.

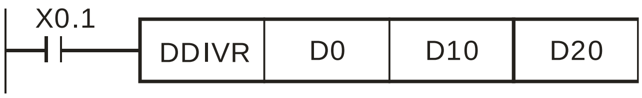
## Example 1

- When X0.0 is ON, the floating-point value F1.23456 is divided by the floating-point value F1.0, and the quotient F1.23456 is stored in (D11, D10).




## Example 2

- When X0.0 is ON, the floating-point value in (D1, D0) is divided by the floating-point value in (D11, D10), and the quotient is stored in (D21, D20).



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 215~217 | D | LD# | | **$S_1$, $S_2$** | Logical operation |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (7 steps) |
|---|---|---|
| – | ✓ | ✓ |

## Explanation

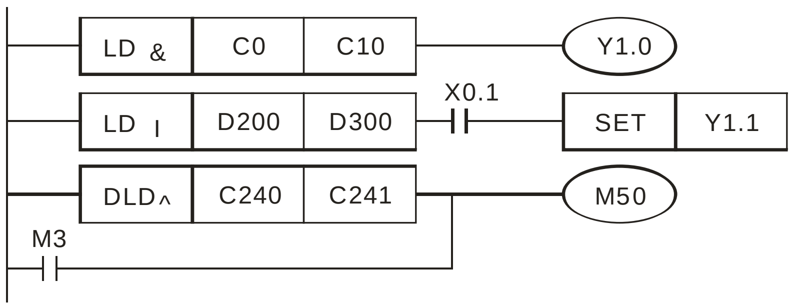
- **$S_1$**: Source device 1; **$S_2$**: Source device 2
- The instruction is used to compare the value in **$S_1$** with that in **$S_2$**. If the comparison result is not 0, the condition of the instruction is met. If the comparison result is 0, the condition of the instruction is not met.
- The instruction LD# can be connected to a busbar directly.

| API No. | 16-bit instruction | 32-bit instruction | ON | OFF |
|---|---|---|---|---|
| 215 | LD& | DLD& | $S_1$ & $S_2$ ≠ 0 | $S_1$ & $S_2$ = 0 |
| 216 | LD\| | DLD\| | $S_1 \mid S_2 \neq 0$ | $S_1 \mid S_2 = 0$ |
| 217 | LD^ | DLD^ | $S_1$ ^ $S_2$ ≠ 0 | $S_1$ ^ $S_2$ = 0 |

- &: Logical AND operation
- |: Logical OR operation
- ^: Logical exclusive OR operation
- If a 32-bit counter is used, the 32-bit insturciton DLD# must be used. If a 32-bit counter and the 16-bit instruction LD# are used, ,a program error will occur, and the ERROR LED indicator on the motion control module will blink. (C200~C255 are 32-bit counters.)

## Example

- A logical AND operator takes the values in C0 and C10, and performs the logical AND operation on each pair of corresponding bits. If the operation result is not 0, Y1.0 will be ON.
- A logical OR operator takes the values in D200 and D300, and performs the logical OR operation on each pair of corresponding bits. If the operation result is not 0 and X0.1 is ON, Y1.1 will be set to ON.
- A logical operator XOR takes the values in C240 and C241, and performs the logical exclusive OR operation on each pair of corresponding bits. If the operation result is not 0, or if M3 is ON, M50 will be ON.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 218~220 | D | AND# | | **$S_1$, $S_2$** | Logical operation |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (7 steps) |
|---|---|---|
| – | ✓ | ✓ |

## Explanation

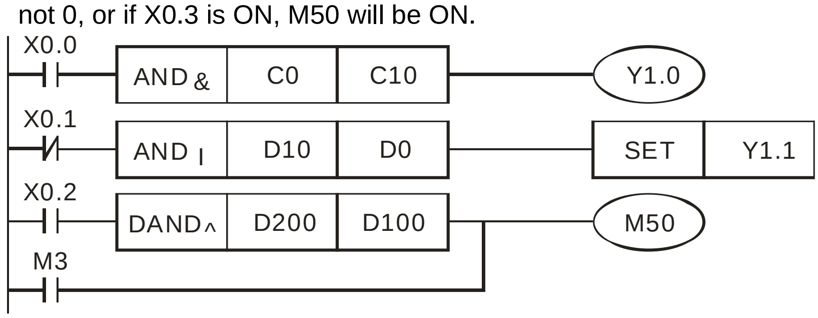
- **$S_1$**: Source device 1; **$S_2$**: Source device 2
- The instruction is used to compare the value in **$S_1$** with that in **$S_2$**. If the comparison result is not 0, the condition of the instruction is met. If the comparison result is 0, the condition of the instruction is not met.
- The instruction AND# is connected to a contact in series.

<table>
<tr><th>API No.</th><th>16-bit instruction</th><th>32-bit instruction</th><th>ON</th><th>OFF</th></tr>
<tr><td>218</td><td>AND&</td><td>DAND&</td><td>$S_1$ & $S_2$ $\neq 0$</td><td>$S_1$ & $S_2$ $=0$</td></tr>
<tr><td>219</td><td>AND|</td><td>DAND|</td><td>$S_1$ | $S_2$ $\neq 0$</td><td>$S_1$ | $S_2$ $=0$</td></tr>
<tr><td>220</td><td>AND^</td><td>DAND^</td><td>$S_1$ ^ $S_2$ $\neq 0$</td><td>$S_1$ ^ $S_2$ $=0$</td></tr>
</table>

- &: Logical AND operation
- |: Logical OR operation
- ^: Logical exclusive OR operation
- If a 32-bit counter is used, the 32-bit instruction DAND# must be used. If a 32-bit counter and the 16-bit instruction AND# are used, a program error will occur, and the ERROR LED indicator on the motion control module will blink. (C200~C255 are 32-bit counters.)

## Example

- When X0.0 is ON, a logical AND operator takes the values in C0 and C10, and performs the logical AND operation on each pair of corresponding bits. If the operation result is not 0, Y1.0 will be set to ON.
- When X0.1 is OFF, a logical OR operator takes the values in D10 and D0, and performs the logical OR operation on each pair of corresponding bits. If the operation result is not 0, Y1.1 will be ON.
- When X0.2 is ON, a logical XOR operator takes the values in (D201, D200) and (D101, D100), and performs the logical exclusive OR operation on each pair of corresponding bits. If the operation result is not 0, or if X0.3 is ON, M50 will be ON.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 221~223 | D | OR# | | $S_1$, $S_2$ | Logical operation |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| $S_2$ | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (7 steps) |
|---|---|---|
| – | ✓ | ✓ |

## Explanation

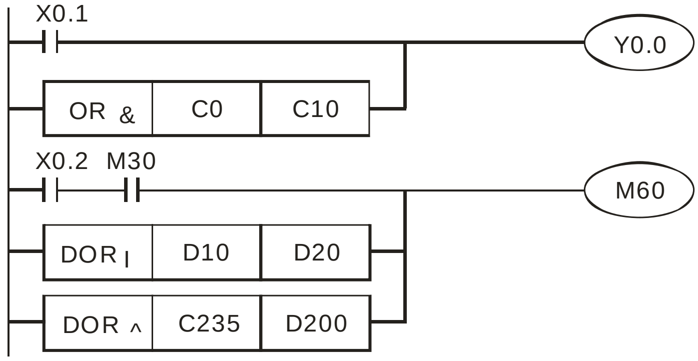
- $S_1$: Source device 1; $S_2$: Source device 2
- The instruction is used to compare the value in $S_1$ with that in $S_2$. If the comparison result is not 0, the condition of the instruction is met. If the comparison result is 0, the condition of the instruction is not met.
- The instruction OR# is connected to a contact in parallel.

| API No. | 16-bit instruction | 32-bit instruction | ON | OFF |
|---|---|---|---|---|
| 221 | OR& | DOR& | $S_1$ & $S_2$ $\neq 0$ | $S_1$ & $S_2$ $=0$ |
| 222 | OR\| | DOR\| | $S_1$ \| $S_2$ $\neq 0$ | $S_1$ \| $S_2$ $=0$ |
| 223 | OR^ | DOR^ | $S_1$ ^ $S_2$ $\neq 0$ | $S_1$ ^ $S_2$ $=0$ |

- &: Logical AND operation
- |: Logical OR operation
- ^: Logical exclusive OR operation
- If a 32-bit counter is used, the 32-bit instruction DOR# must be used. If a 32-bit counter and the 16-bit instruction OR# are used, a program error will occur, and the ERROR LED indicator on the motion control module will blink. (C200~C255 are 32-bit counters.)

## Example

- When X0.1 is ON, Y0.0 is ON. Besides, when a logical AND operator 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.
- When X0.2 and M30 are ON, M60 is ON. When a logical OR operator performs the logical OR operation on each pair of corresponding bits in the 32-bit register (D11, D10) and the 32-bit register (D21, D20), and the operation result is not 0, M60 is ON. Besides, when the logical XOR operator performs the logical exclusive OR operation on each pair of corresponding bits in the 32-bit counter C2350 and the 32-bit register (D201, D200), and the operation result is not 0, M60 is ON.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 224~230 | D | LD※ | | $S_1$, $S_2$ | Comparing values |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| $S_2$ | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (7 steps) |
|---|---|---|
| – | ✓ | ✓ |

## Explanation

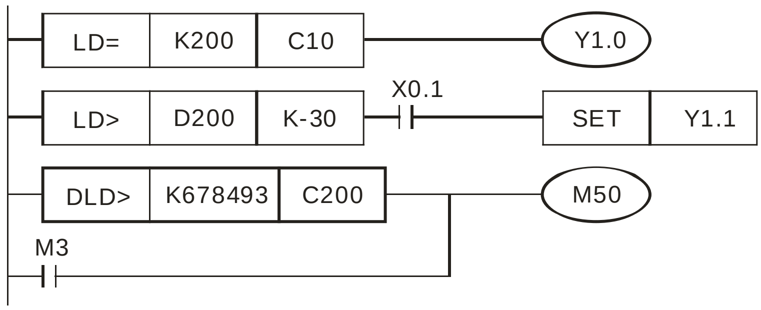
- $S_1$: Source device 1; $S_2$: Source device 2
- The instruction is used to compare the value in $S_1$ with that in $S_2$. Take the instruction LD= for instance. If the comparison result is that the value in $S_1$ is equal to that in $S_2$, the condition of the instruction is met. If the comparison result is that the value in $S_1$ is not equal to that in $S_2$, the condition of the instruction is not met.
- The instruction LD※ can be connected to a busbar directly.

| API No. | 16-bit instruction | 32-bit instruction | ON | OFF |
|---|---|---|---|---|
| 224 | LD = | DLD = | $S_1 = S_2$ | $S_1 \neq S_2$ |
| 225 | LD > | DLD > | $S_1 > S_2$ | $S_1 \leq S_2$ |
| 226 | LD < | DLD < | $S_1 < S_2$ | $S_1 \geq S_2$ |
| 228 | LD < > | DLD < > | $S_1 \neq S_2$ | $S_1 = S_2$ |
| 229 | LD < = | DLD < = | $S_1 \leq S_2$ | $S_1 > S_2$ |
| 230 | LD > = | DLD > = | $S_1 \geq S_2$ | $S_1 < S_2$ |

- If a 32-bit counter is used, the 32-bit insturciton DLD※ must be used. If a 32-bit counter and the 16-bit instruction LD※ are used, ,a program error will occur, and the ERROR LED indicator on the motion control module will blink. (C200~C255 are 32-bit counters.)

## Example

- When the value in C10 is equal to K200, Y1.0 is ON.
- When the value in D200 is greater than K-30, and X0.1 is ON, Y1.1 is set to ON.
- When the value in C200 is less than K678,493, or when M3 is ON, M50 is ON.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 232~238 | D | AND※ | | $S_1$, $S_2$ | Comparing values |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| $S_2$ | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (7 steps) |
|---|---|---|
| – | ✓ | ✓ |

## Explanation

- $S_1$: Source device 1; $S_2$: Source device 2
- The instructions are used to compare the value in $S_1$ with that in $S_2$. Take the instruction AND= for instance. If the comparison result is that the value in $S_1$ is equal to that in $S_2$, the condition of the instruction is met. If the comparison result is that the value in $S_1$ is not equal to that in $S_2$, the condition of the instruction is not met.
- The instruction AND※ is connected to a contact in series.

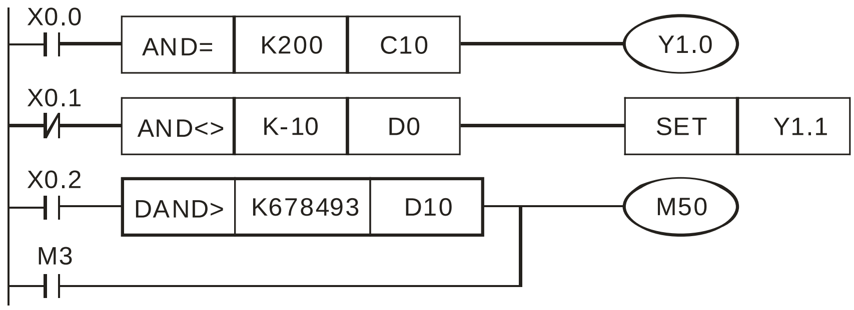
| API No. | 16-bit instruction | 32-bit instruction | ON | OFF |
|---|---|---|---|---|
| 232 | AND = | DAND = | $S_1 = S_2$ | $S_1 \neq S_2$ |
| 233 | AND > | DAND > | $S_1 > S_2$ | $S_1 \leqq S_2$ |
| 234 | AND < | DAND < | $S_1 < S_2$ | $S_1 \geqq S_2$ |
| 236 | AND < > | DAND < > | $S_1 \neq S_2$ | $S_1 = S_2$ |
| 237 | AND < = | DAND < = | $S_1 \leqq S_2$ | $S_1 > S_2$ |
| 238 | AND > = | DAND > = | $S_1 \geqq S_2$ | $S_1 < S_2$ |

- If a 32-bit counter is used, the 32-bit insturciton DAND※ must be used.

  If a 32-bit counter and the 16-bit instruction AND※ are used, ,a program error will occur, and the ERROR LED indicator on the motion control module will blink. (C200~C255 are 32-bit counters.)

## Example

- When X0.0 is ON and the present value in C10 is equal to K200, Y1.0 is ON.
- When X0.1 is OFF and the value in D0 is not equal to K-10, Y1.1 is set to ON.
- When X0.2 is ON and the value in (D11, D10) is less than 678,493, or when M3 is ON, M50 is ON.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 240~246 | D | OR※ | | **$S_1$, $S_2$** | Comparing values |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| $S_2$ | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (5 steps) | 32-bit instruction (7 steps) |
|---|---|---|
| – | ✓ | ✓ |

## Explanation

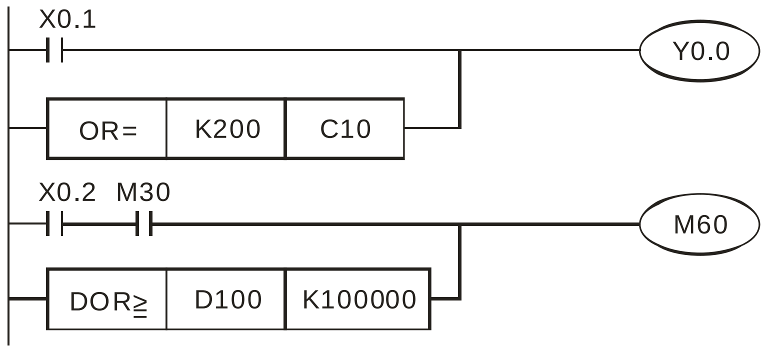
- **$S_1$**: Source device 1; **$S_2$**: Source device 2
- The instructions are used to compare the value in **$S_1$** with that in **$S_2$**. Take the instruction OR= for instance. If the comparison result is that the value in **$S_1$** is equal to that in **$S_2$**, the condition of the instruction is met. If the comparison result is that the value in **$S_1$** is not equal to that in **$S_2$**, the condition of the instruction is not met.
- The instruction OR※ is connected to a contact in parallel.

| API No. | 16-bit instruction | 32-bit instruction | ON | OFF |
|---|---|---|---|---|
| 240 | OR = | **D**OR = | $S_1 = S_2$ | $S_1 \neq S_2$ |
| 241 | OR > | **D**OR > | $S_1 > S_2$ | $S_1 \leq S_2$ |
| 242 | OR < | **D**OR < | $S_1 < S_2$ | $S_1 \geq S_2$ |
| 244 | OR < > | **D**OR < > | $S_1 \neq S_2$ | $S_1 = S_2$ |
| 245 | OR < = | **D**OR < = | $S_1 \leq S_2$ | $S_1 > S_2$ |
| 246 | OR > = | **D**OR > = | $S_1 \geq S_2$ | $S_1 < S_2$ |

- If a 32-bit counter is used, the 32-bit insturciton DOR※ must be used. If a 32-bit counter and the 16-bit instruction OR※ are used, ,a program error will occur, and the ERROR LED indicator on the motion control module will blink. (C200~C255 are 32-bit counters.)

## Example

- When X0.1 is ON, or when the present value in C10 is equal to K200, Y0.0 is ON.
- When X0.2 and M30 are ON, or when the value in (D101, D100) is greater than or equal to K100,000, M60 is ON.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 152 | D | SWAP | P | **S** | Interchanging the high byte in a device with the low byte in the device |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

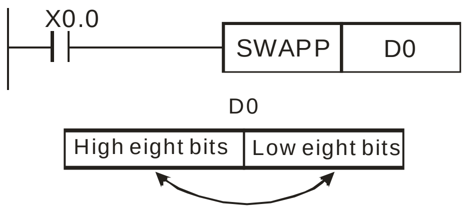
| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction (3 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **S**: Source device
- When the 16-bit instruction is executed, the high eight bits in **S** are interchanged with the low eight bits in **S**.
- When the 32-bit instruction is executed, the high eight bits in **S** are interchanged with the low eight bits in **S**, and the high eight bits in **S**+1 are interchanged with the low eight bits in **S**+1.
- Generally, the pulse instructions SWAPP and DSWAPP are used.

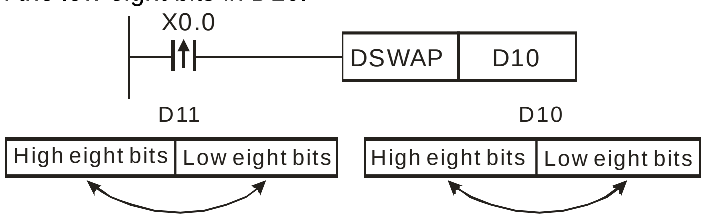
## Example 1

- When X0.0 is ON, the high byte in D0 is interchanged with the low byte in D0.




## Example 2

- When X0.0 is ON, the high eight bits in D11 are interchanged with the low eight bits in D11, and the high eight bits in D10 are interchanged with the low eight bits in D10.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 154 | D | RAND | P | **$S_1$, $S_2$, D** | Random value |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **$S_2$** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

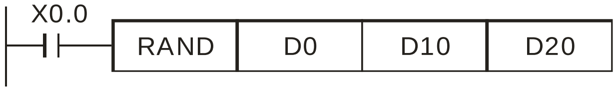
Explanation

- **$S_1$**: Minimum random value; **$S_2$**: Maximum random value; **D**: Result
- 16-bit instruction: The value in **$S_1$** and the value in **$S_2$** are in the range of K0 to K32,767.
  32-bit instruction: The value in **$S_1$** and the value in **$S_2$** are in the range of K0 to K2,147,483,647.
- The value in **$S_1$** must be less than the value in **$S_2$**. If the value in **$S_1$** is greater than the value in **$S_2$**, an operation error will occur.
- If KnM/KnS is used, it is suggested that M device numbers/S device numbers should start from a number which is a multiple of 16 in the decimal numeral system, e.g. K4M0 and K4S16 (decimal numeral system).

5

Example

- When X0.0 is ON, the instruction RAND is used to generate a random value in the range of the value in D0 to the value in D10, and the random value is stored in D20.



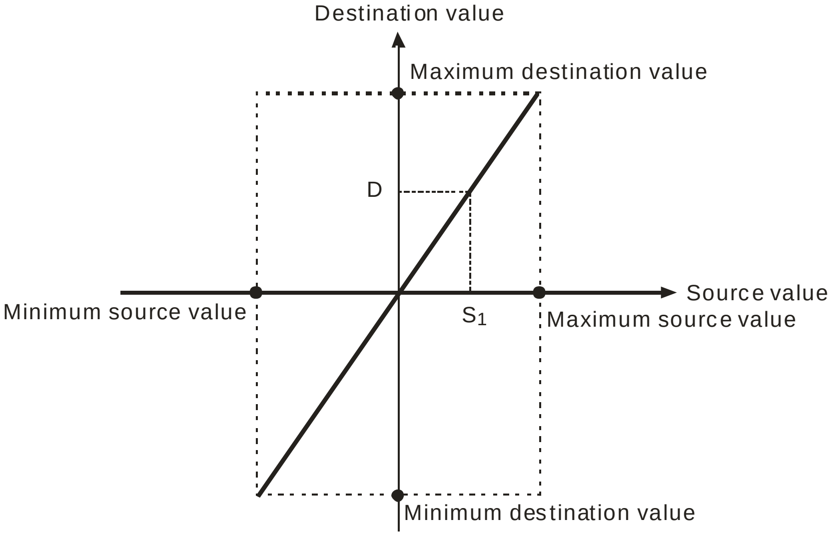
| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 202 | | SCAL | P | $S_1$, $S_2$, $S_3$, D | Scale |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | | | ○ | ○ | | | | ● | ● | | | | |
| $S_2$ | | | | | ○ | ○ | | | | ● | ● | | | | |
| $S_3$ | | | | | ○ | ○ | | | | ● | ● | | | | |
| D | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

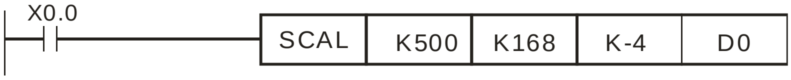
## Explanation

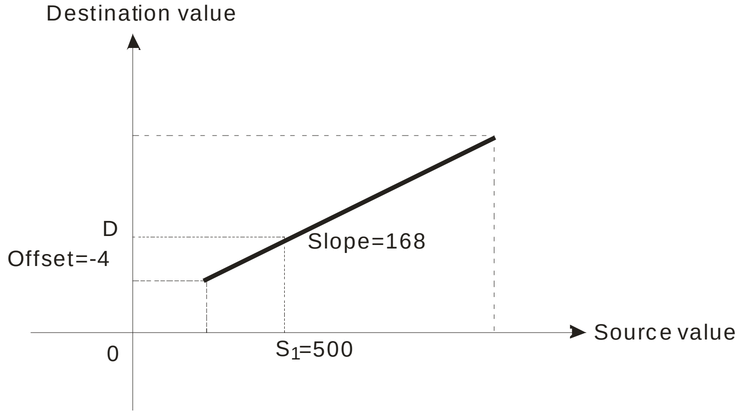
- $S_1$: Source device; $S_2$: Slope (Unit: 0.001); $S_3$: Offset; **D**: Destination device
- The values in $S_1$, $S_2$, and $S_3$ must be in the range of -32767 to 32767.
- Equation: **D**=($S_1$×$S_2$)÷1000+$S_3$
- To obtain the value in $S_2$, users have to use the slope equation below, round the result to the nearest integer, and get a 16-bit integer. To obtain the value in $S_3$, the users have to use the offset equation below, round the result to the nearest integer, and get a 16-bit integer.
- Slope equation: $S_2$=[(Maximum destination value–Minimum destination value)÷(Maximum source value–Minimum source value)]×1,000
- Offset equation: $S_3$= Minimum destination value–Minimum source value×$S_2$÷1,000
- Output curve



**Example 1**

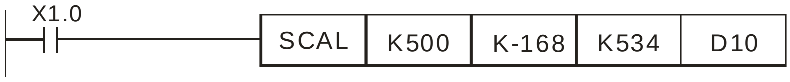
- Suppose the values in $\mathbf{S_1}$, $\mathbf{S_2}$, and $\mathbf{S_3}$ are 500, 168, and -4 respectively. When X0.0 is ON, the instruction SCAL is executed, and a scale is stored in D0.
- Equation: D0=(500×168)÷1000+(-4)=80

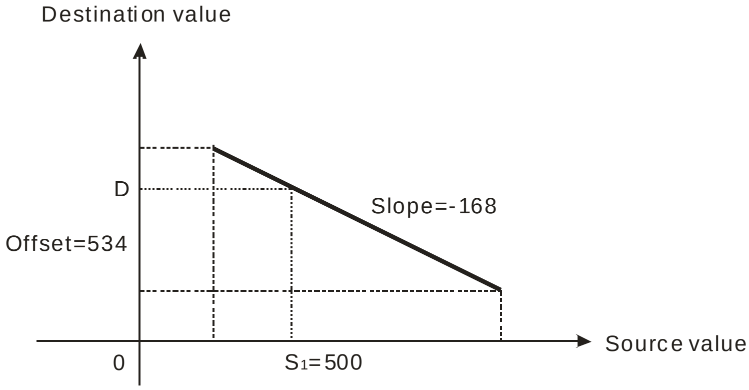



**Example 2**

- Suppose the values in $\mathbf{S_1}$, $\mathbf{S_2}$, and $\mathbf{S_3}$ are 500, -168, and 534 respectively. When X0.0 is ON, the instruction SCAL is executed, and a scale value is stored in D10.
- Equation: D10=(500×-168)÷1000+534=450




**Additional remark**

- Only when a slope and an offset are known can the instruction SCAL be used. If a slope and an offset are unknown, it is suggested that users should use the instruction SCLP.
- The value in $\mathbf{S_2}$ must be in the range of –32,768 to 32,767 (The actual value in $\mathbf{S_2}$ must be in the range of –32,768 to 32,767.) If the value in $\mathbf{S_2}$ is not in the range, please use the instruction SCLP instead.
- If users use the slop equation above, the maximum source value must be greater than the minimum source value, and the maximum destination value do not have to be greater than the minimum destination value.
- If the value in **D** is greater 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.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 203 | D | SCLP | P | **$S_1$, $S_2$, D** | Parameter scale |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **$S_1$** | | | | | ○ | ○ | | | | ● | ● | | | | |
| **$S_2$** | | | | | ○ | ○ | | | | ● | ● | | | | |
| **D** | | | | | | | | | | ● | ● | | | | |

| Pulse instruction | 16-bit instruction (7 steps) | 32-bit instruction (9 steps) |
|---|---|---|
| ✓ | ✓ | ✓ |

## Explanation

- **$S_1$**: Source device; **$S_2$**: Parameter (Unit: 0.001); **D**: Destination device
- 16-bit instruction: The setting of **$S_2$** is described below.

| Device number | Parameter | Setting range |
|---|---|---|
| **$S_2$** | Maximum source value | -32768~32767 |
| **$S_2$**+1 | Minimum source value | -32768~32767 |
| **$S_2$**+2 | Maximum destination value | -32768~32767 |
| **$S_2$**+3 | Minimum destination value | -32768~32767 |

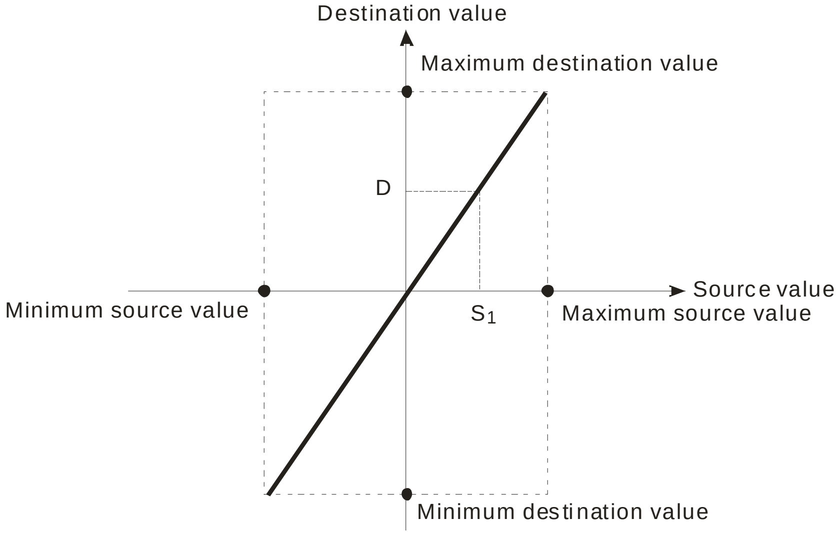
- If the 16-bit instruction is used, **$S_2$** will occupy four consecutive devices.
- 32-bit instruction: The setting of **$S_2$** is decribed below.

| Device number | Parameter | Setting range | |
|---|---|---|---|
| | | Integer | Floating-point value |
| **$S_2$**, **$S_2$**+1 | Maximum source value | -2,147,483,648~ 2,147,483,647 | 32-bit floating-point values available |
| **$S_2$**+2, **$S_2$**+3 | Minimum source value | | |
| **$S_2$**+4, **$S_2$**+5 | Maximum destination value | | |
| **$S_2$**+6, **$S_2$**+7 | Minimum destination value | | |

- If the 32-bit instruction is used, **$S_2$** will occupy eight consecutive devices.
- Flag: M1162 is a decimal integer/binary floating-point value flag. (ON: Binary floating-point value)
- Equation: **D**=[(**$S_1$**–Minimum source value)×(Maximum destination value–Minimum destination value)]÷(Maximum source value–Minimum source value)+Minimum destination value
- Relation between the source value in **$S_1$** and the destination value in **D**:
  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
- After the parameters above are substituted for y, k, x, and b in the equation y=kx+b, the equation below will be obtained.

  y=kx+b=**D**=k**$S_1$**+b=Slope×**$S_1$** + Offset = Slope×**$S_1$** + Minimum destination value–Minimum source value×Slope=Slope×(**$S_1$**–Minimum source value) + Minimum destination value =(**$S_1$**–Minimum source value)×(Maximum destination value–Minimum destination value)÷(Maximum source value–Minimum source value) + Minimum destination value

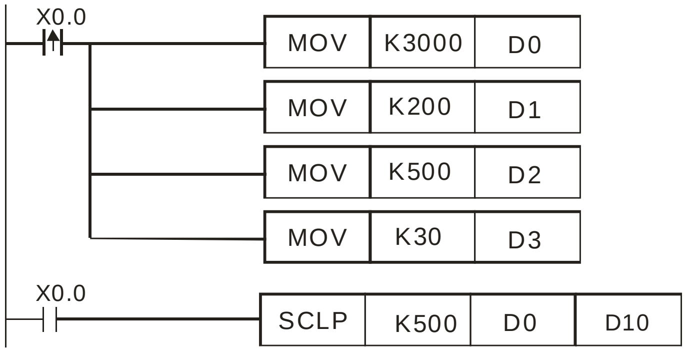
- If the value in **$S_1$** is greater than the maximum source value, the value in **$S_1$** will be equal to the maximum source value. If the value in **$S_1$** is less than the minimum source value, the value in **$S_1$** will be equal to the minimum source value. After input values and parameters are set, an output curve will be gotten.

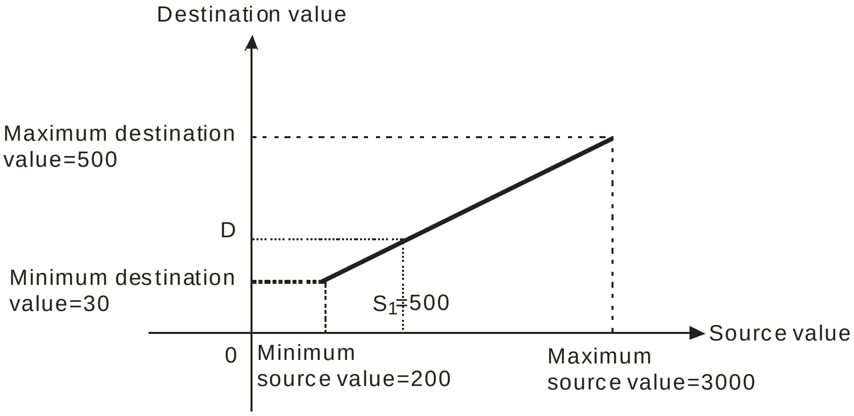



5

Example 1

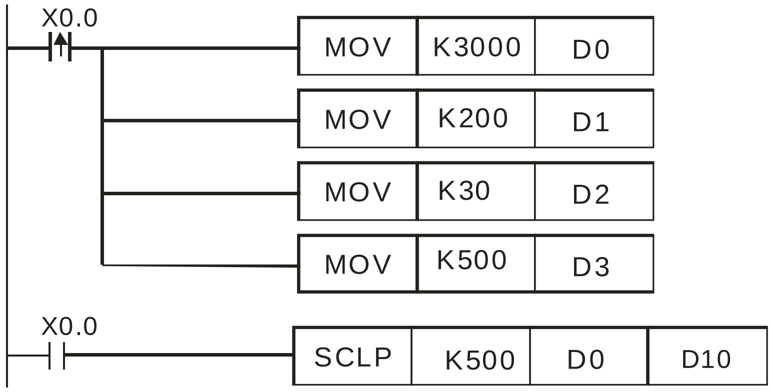
- Suppose the value in **$S_1$** is 500, the maximum source value in D0 is 3,000, the minimum source value in D1 is 200, the maximum destination value in D2 is 500, and the minimum destination value in D3 is 30. When X0.0 is ON, the instruction SCLP is executed, and a scale is stored in D10.
- Equation: D10=[(500–200)×(500–30)]÷(3,000–200)+30=80.35
80.35 is rounded to the nearest integer, and becomes 80. 80 is stored in D10.

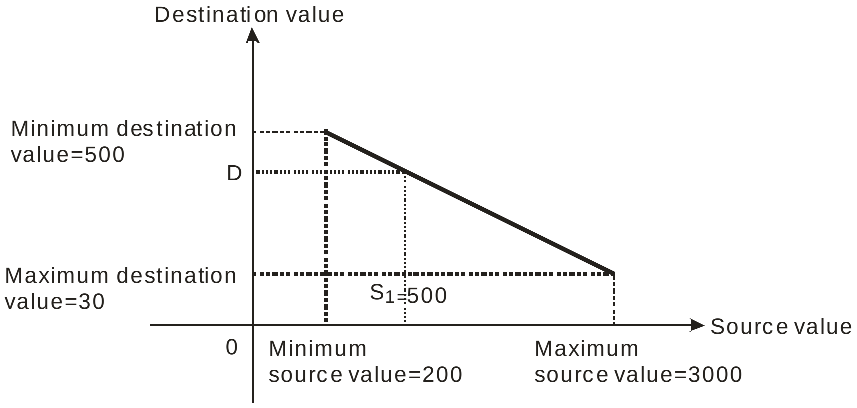


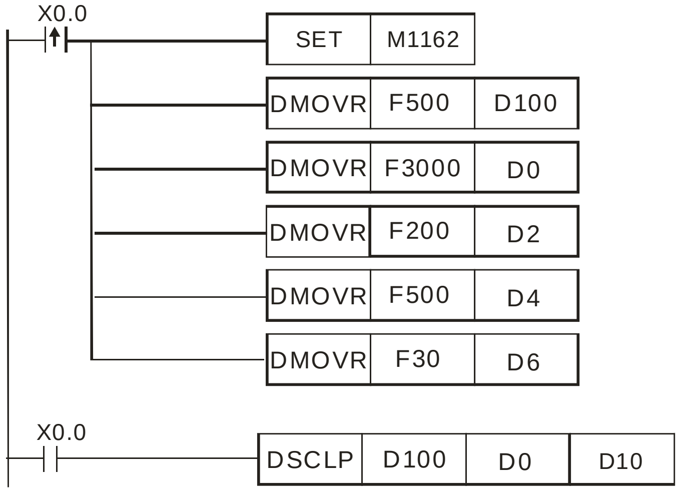
Example 2

- 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 a scale is stored in D10.
- Eequation: D10=[(500–200)×(30–500)]÷(3,000–200)+500=449.64
  449.64 is rounded to the nearest integer, and becomes 450. 450 is stored in D10.




- Suppose $\mathbf{S_1}$ is D100, the value in D100 is F500, the maximum source value in D0 is F3000, the minimum source value in D2 is F200, the maximum destination value in D4 is F500, and the minimum destination value in D6 is F30. When X0.0 is ON, M1162 is set to ON, the instruction DSCLP is executed, and a scale is stored in D10.
- Equation: D10=[(F500–F200)×(F500–F30)]÷(F3000–F200)+F30=F80.35
  F80.35 is rounded to the nearest integer, and becomes F80. F80 is stored in D10.



Additional remark

- 16-bit instruction: The value in $\mathbf{S_1}$ is in the range of the minimum source value and the maximum source value, i.e. the value in $\mathbf{S_1}$ is in the range of -32,768 to 32,767. If the value in $\mathbf{S_1}$ exceeds the minimum source value/the maximum source value, the minimum source value/the maximum source value will be used.
- 32-bit instruction: The integer in $\mathbf{S_1}$ is in the range of the minimum source value and the maximum source value, i.e. the integer in $\mathbf{S_1}$ is in the range of -2,147,483,648 to 2,147,483,647. If the integer in $\mathbf{S_1}$ exceeds the minimum source value/the maximum source value, the minimum source value/the maximum source value will be used.
- 32-bit instruction: The floating-point value in $\mathbf{S_1}$ is in the range of the minimum source value and the maximum source value, i.e. the floating-point value in $\mathbf{S_1}$ is a 32-bit floating-point value available. If the floating-point value in $\mathbf{S_1}$ exceeds the minimum source value/the maximum source value, the minimum source value/the maximum source value will be used.
- If users use the instruction, the maximum source value must be greater than the minimum source value, and the maximum destination value does not have to be greater than the minimum destination value.

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 256 | | CJN | P | **S** | Negated conditional jump |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | | | | | | | | |

| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

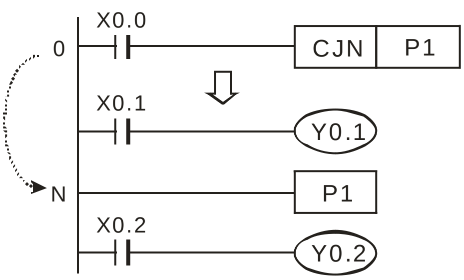
## Explanation

- **S**: Pointer (S is in the rage of P0~P255. A pointer does not support V devices and Z devices)
- If the conditional contact connected to CJN is ON, the next address will be executed. If the conditional contact connected to CJN is not ON, the address to which **S** points will be executed.
- If some part of the main program O100 does not need to be executed, users can use CJN or CJNP to shorten the scan time. Besides, if a dual output is used, users can use CJ or CJP.
- If the program specified by a pointer is prior to the instruction CJN, a watchdog timer error will occur, and the main program O100 will not be executed. Please use the instruction carefully.
- The instruction CJN can specify the same pointer repeatedly. The pointer specified by CJN can not be the same as the pointer specified by CALL, otherwise an error will occur.
- When the instruction CJN/CJNP in a program is executed, the actions of the devices in the program are as follows.
  - The states of the Y devices, the states of the M devices, and the states of the S devices in the program remain the same as those before the execution of the jump.
  - The 10 millisecond timers in the program stop counting.
  - The general counters in the program stop counting, and the general applied instructions in the program are not executed.
  - If the instructions which are used to reset the timers in the program are driven before the jump is executed, the timers will still be reset during the execution of the jump.

## Example

- When X0.0 is OFF, the execution of the program jumps from address 0 to address N (P1), and the addresses between address 0 and address N are skipped.
- When X0.0 is ON, the execution of the program starts from address 0, and the instruction CJN is not executed.

(Negated conditional jump)



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 257 | | JMP | | **S** | Unconditional jump |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | | | | | | | | | | | |

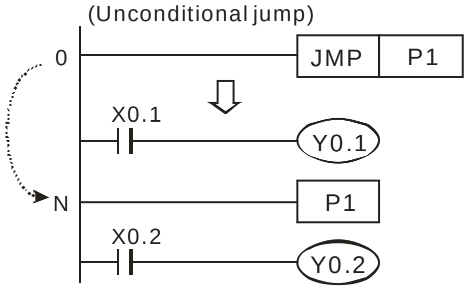
| Pulse instruction | 16-bit instruction (3 steps) | 32-bit instruction |
|---|---|---|
| – | ✓ | – |

Explanation

- The function of JMP is similar to the function of CJ. CJ must be driven by a contact whereas JMP does not have to be driven by a contact.
- The pulse instruction JMPP is not supported.

Example

- After address 0 is scanned, address N will be executed whether there is a conditional contact before the instruction JMP (and whether the conditional contact is ON or OFF), and the addresses between address 0 and address N (P1) will be skipped.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 258 | | BRET | | – | Returning to a busbar |

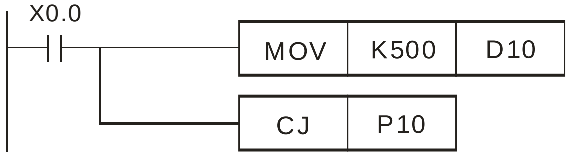
| Pulse instruction | 16-bit instruction (1 step) | 32-bit instruction |
|---|---|---|
| – | ✓ | – |

Explanation

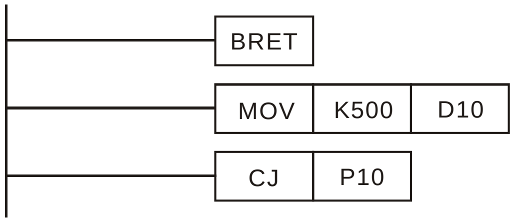
- The instruction BRET does not have to be driven by a contact.
- After the instruction BRET is executed, the instructions which should be driven by a conditional contact will seem to be connected to a busbar, and will be executed.

Example

- In the general program shown below, the instructions are executed only when X0.0 is ON.




- After the instruction BRET is added, the instructions which should be driven by a contact will seem to be connected to a busbar, and will be executed.



| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 259 | | MMOV | P | **S, D** | Converting a 16-bit value into a 32-bit value |

| Device | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

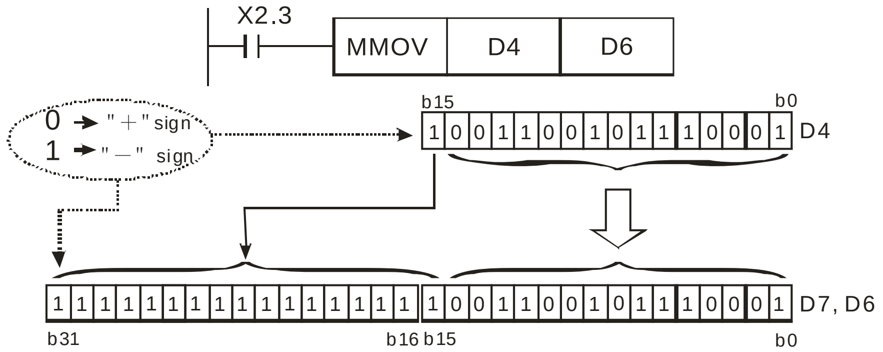
| Pulse instruction | 16-bit instruction (1 step) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

## Explanation

- **S**: Source device (16-bit device); **D**: Destination device (32-bit device)
- The value in the 16-bit device **S** is transferred to the 32-bit device **D**. The sign bit in **S** is duplicated, and stored in **D**.

## Example

- When X2.3 is ON, the value in D4 is transferred to D6 and D7.




Bit 15 is D4 is transferred to bit 15~ bit 31 in (D7, D6). The value in (D7, D6) becomes a negative value. (The value in D4 is also a negative value.)

| API | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 260 | | RMOV | P | **S, D** | Converting a 32-bit value into a 16-bit value |

| **Device** | Xn.n | Yn.n | M | S | K | 16# | F | KnM | KnS | D | W | T | C | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S** | | | | | ○ | ○ | | ● | ● | ● | ● | ● | ● | ○ | ○ |
| **D** | | | | | | | | ● | ● | ● | ● | ● | ● | ○ | ○ |

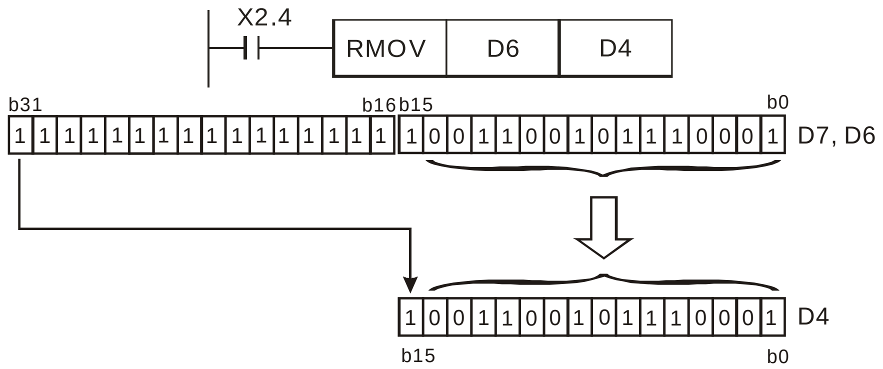
| Pulse instruction | 16-bit instruction (6 steps) | 32-bit instruction |
|---|---|---|
| ✓ | ✓ | – |

Explanation

- **S**: Source device (32-bit device); **D**: Destination device (16-bit device)
- Data in the 32-bit device **S** is transferred to the 16-bit device **D**. The sign bit in **S** is retained.

Example

- When X2.4 is ON, data in D6 and D7 is transferred to D4.




When X2.4 is ON, bit 31 in D7 is transferred to bit 15 in D4, bit 0~bit 14 in D6 are transferred, and bit 15~bit 30 in D6 and D7 are not transferred.

## 5.7 Motion Control Function Block Table

| Type | Name | Description | Model | | | Page number |
|---|---|---|---|---|---|---|
| | | | 20MC | 10PM/ 15PM | 05PM | |
| Uniaxial motion control function blocks | Absolute single-speed motion | Starting absolute single-speed motion | ✓ | ✓ | ✓ | 5-138 |
| | Relative single-speed motion | Starting relative single-speed motion | ✓ | ✓ | ✓ | 5-141 |
| | Absolute two-speed motion | Starting absolute two-speed motion | ✓ | ✓ | ✓ | 5-146 |
| | Relative two-speed motion | Starting relative two-speed motion | ✓ | ✓ | ✓ | 5-150 |
| | Inserting single-speed motion | Inserting single-speed motion | ✓ | ✓ | ✓ | 5-154 |
| | Inserting two-speed motion | Inserting two-speed motion | ✓ | ✓ | ✓ | 5-158 |
| | JOG motion | Starting JOG motion | ✓ | ✓ | ✓ | 5-162 |
| | Manual pulse generator mode | Enabling a manual pulse generator mode | ✓ | ✓ | ✓ | 5-165 |
| | Retuning home | Starting motion of returning home | - | ✓ | ✓ | 5-168 |
| | Stopping uniaxial motion | Stopping the motion of the axis specified | ✓ | ✓ | ✓ | 5-171 |
| | Parameter setting 1 | Setting motion parameters | ✓ | ✓ | ✓ | 5-174 |
| | Parameter setting 2 | Setting motion parameters | ✓ | ✓ | ✓ | 5-176 |
| | Reading the present position/speed of an axis | Reading the present position/speed of an axis | ✓ | ✓ | ✓ | 5-179 |
| | State of an axis | Reading and clearing the present erroneous state of an axis | ✓ | ✓ | ✓ | 5-181 |
| | Setting the present position of an axis | Setting the present position of an axis | ✓ | ✓ | ✓ | 5-183 |
| | Setting the polarities of input terminals | Setting the polarities of input terminals | ✓ | ✓ | ✓ | 5-185 |
| | Electronic gear motion | Starting electronic gear motion | ✓ | ✓ | ✓ | 5-188 |
| | Electronic cam motion | Starting electronic cam motion | ✓ | ✓ | ✓ | 5-190 |
| | Reading a cam point | Reading a particular point in a cam chart | ✓ | ✓ | ✓ | 5-194 |
| | Writing a cam point | Modifying a particular point in a cam chart | ✓ | ✓ | ✓ | 5-196 |
| | Calculating a synchronization ratio | Calculating a synchronization ratio | ✓ | ✓ | ✓ | 5-198 |
| | Creating a cam curve | Creating a cam curve | ✓ | ✓ | ✓ | 5-200 |
| | Updating a cam curve | Updating a cam curve | ✓ | ✓ | ✓ | 5-203 |

| Type | Name | Description | Model | | | Page number |
|---|---|---|---|---|---|---|
| | | | 20MC | 10PM/ 15PM | 05PM | |
| Multiaxial motion control function blocks | Setting the parameters of G-code motion | Setting the parameters of G-code motion | ✓ | ✓ | ✓ | 5-205 |
| | Executing G-code motion | Setting and executing an Ox motion subroutine | ✓ | ✓ | ✓ | 5-207 |
| | Stopping G-code motion | Stopping the execution of an Ox motion subroutine | ✓ | ✓ | ✓ | 5-210 |
| | Reading an M-code | Reading an M-code | ✓ | ✓ | ✓ | 5-212 |
| | Multiaxial absolute linear interpolation | Starting multiaxial absolute linear interpolation | ✓ | ✓ | ✓ | 5-215 |
| | Multiaxial relative linear interpolation | Starting multiaxial relative linear interpolation | ✓ | ✓ | ✓ | 5-217 |
| | Stopping multiaxial linear interpolation | Stopping multiaxial linear interpolation | ✓ | ✓ | ✓ | 5-219 |
| Network function blocks | Starting/Stopping a servo drive | Starting or stopping the servo drive specified on a DMCNET. | ✓ | - | - | 5-221 |
| | Resetting a servo drive | Resetting the servo drive specified on a DMCNET | ✓ | - | - | 5-222 |
| | Writing the value of a parameter into a servo drive | Writing the value of a parameter into the servo drive specified on a DMCNET | ✓ | - | - | 5-224 |
| | Reading the value of a parameter from a servo drive | Reading the value of a parameter from the servo drive specified on a DMCNET | ✓ | - | - | 5-226 |
| | Instructing a servo drive to return home | Instructing the servo drive specified on a DMCNET to return home | ✓ | - | - | 5-229 |
| | Initializing a servo drive | Initializing the servo drive specified on a DMCNET | ✓ | - | - | 5-232 |
| | Instructing a servo drive to capture values | Instructing the servo drive specified on a DMCNET to capture values | ✓ | - | - | 5-235 |
| | Setting an Ethernet IP address | Setting the Ethernet IP address of the module used | ✓ | ✓ | ✓ | 5-237 |
| Other motion control function blocks | Backing a main program up onto an SD card | Backing a main program up onto an SD card | ✓ | ✓ | ✓ | 5-239 |
| | Backing the values in devices up onto an SD card | Backing the values in the devices in a module up onto an SD card | ✓ | ✓ | ✓ | 5-240 |
| | Restoring the values in devices in an SD card | Reading the values in the devices specified from the file specified in an SD card | ✓ | ✓ | ✓ | 5-242 |
| | High-speed counter | Starting a high-speed counter | ✓ | ✓ | ✓ | 5-244 |
| | High-speed timer | Starting a high-speed timer | ✓ | ✓ | ✓ | 5-246 |
| | Setting high-speed comparison | Starting high-speed comparison | ✓ | ✓ | ✓ | 5-248 |
| | Resetting high-speed comparison | Resetting high-speed comparison | ✓ | ✓ | ✓ | 5-251 |
| | Setting high-speed capture | Starting high-speed capture | ✓ | ✓ | ✓ | 5-252 |
| | High-speed masking | Starting high-speed masking | ✓ | ✓ | ✓ | 5-255 |
| | Setting an interrupt | Setting the trigger for an interrupt subroutine | ✓ | ✓ | ✓ | 5-257 |
| | Absolute encoder | Starting the reading of the position of an absolute encoder | - | ✓ | - | 5-258 |

## 5.8 Introduction of the Pins in a Motion Control Function Block

### 5.8.1 Definitions of Input Pins/Output Pins

Common input pins and output pins in motion control function blocks are listed below. The pins listed below do not appear in a single motion control function block. For example, a motion control function block only has one input pin, that is, it has either the Execute input pin or the Enable input pin.

| Input pin | | | |
|---|---|---|---|
| **Name** | **Description** | **Format** | **Setting value** |
| Execute | Starting the motion control function block | BOOL | True/False |
| Enable | Starting the motion control function block | BOOL | True/False |
| **Output pin** | | | |
| **Name** | **Description** | **Format** | **Setting value** |
| Done | The execution of the function block is complete. | BOOL | There is a transition in the Done output pin's signal from low to high when the execution of motion control function block is complete. |
| Valid | An output value is valid. | BOOL | There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. |
| Busy | The motion control function block is being executed. | BOOL | There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | There is a transition in the Aborted output pin's signal from low to high when the execution of the motion control function block is interrupted by a command. |
| Error | An error occurs in a function block. | BOOL | There is a transition in the Error output pin's signal from low to high when an error occurs in the motion control function block. |

A motion control function block has either the Execute input pin or the Enable input pin. The Execute input pin/The Enable input pin in a motion control function block is used to start the motion control function block. A motion control function block generally has the Busy output pin and the Done output pin. The Busy output pin and the Done output pin in a function block indicate the state of the motion control function block. If the execution of motion control function block is to be interrupted by another motion control function block, the Aborted output pin will be added to the motion control function block. Besides, the Error output pin in a motion control function block is used to indicate that an error occurs in the motion control function block when the motion control function block is executed.

A motion control function block has not only the Execute input pin/the Enable input pin, but also value/state input pins. The characteristics of the value/state input pins are described below.

- Use of input values:
  - If the input pin that a motion control function block has is the Execute input pin, values are used when there is a transition in the Execute input pin's signal from low to high. If a new value is created, it becomes valid when the Execute input pin is triggered again.
  - If the input pin that a motion control function block has is the Enable input pin, values are used when there is a transition in the Enable input pin's signal from low to high. Compared with the Execute input pin, the Enable input pin is used more often when a value used is updated repeatedly.
- An input value exceeds a range.

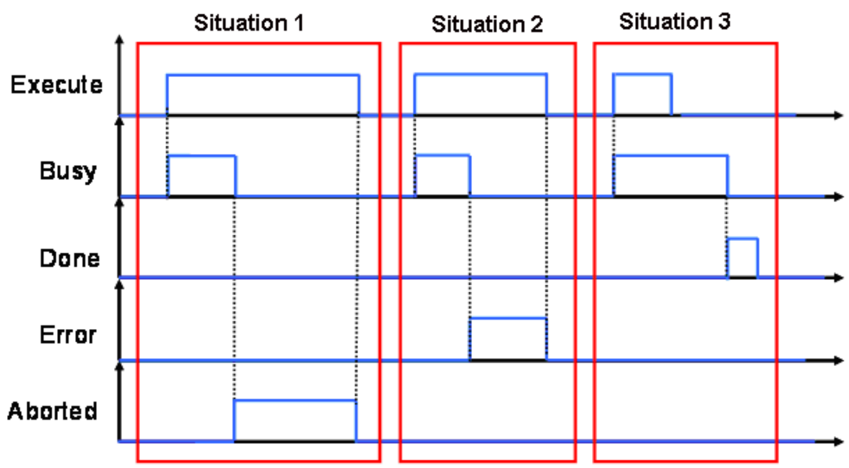
  After a motion control function block is started, the input values which are not in ranges allowed will be limited, or result in an error occurring in the motion control function block. If an error occurring in a motion control function block results in an error occurring in an axis, the motion control function block is applied incorrectly. Users should prevent incorrect values from being generated in an applied program.
- Output pins are mutually exclusive.
  - If the input pin that a motion control function block has is the Execute input pin, only the Busy output pin, the Done output pin, the Aborted output pin, or the Error output pin can be set to True. If the Execute input pin is set to True, the Busy output pin, the Done output pin, the Aborted output pin, or the Error output pin must be set to True.
  - If the input pin that a motion control function block has is the Enable input pin, the Valid output pin and the Error output pin are mutually exclusive, and only the Valid output pin or the Error output pin can be set to True.
- Time when output data/states are valid
  - If the input pin that a motion control function block has is the Execute input pin, the Done output pin, the Error output pin, the Aborted output pin, and data output are reset when there is a transition in the Execute input pin's signal from high to low, but the execution of the function block does not stop when there is a transition in the Execute input pin's signal from high to low. Even if the Execute input pin in a motion control function block is reset before the execution of the motion control function block is complete, output states will still be generated and retained for one cycle. If a motion control function block is started again before the execution of the motion control function block is complete, the motion control function block will not give feedback to the Done output pin and the Aborted output pin, and an error will occur.
  - If the input pin that a motion control function block has is the Enable input pin, the Valid output pin, the Busy output pin, and the Error output pin are reset when there is a transition in the Enable input pin's signal from high to low.
- Characteristic of the Done output pin

  The Done output pin in a motion control function block will be set to True after the motion control function block is executed successfully.
- Characteristic of the Busy output pin
  - If the input pin that a motion control function block has is the Execute input pin, the motion control function block uses the Busy output pin to indicate that the execution of the motion control function block is not complete, and new output states (values) are expected to be generated. The Busy output pin is set to True when there is a transition in the Execute input pin's signal from low to high. When the Done output pin, the Aborted output pin, and the Error output pin are set to True, the Busy output pin are reset.
  - If the input pin that a motion control function block has is the Enable input pin, the motion control function block uses the Busy output pin to indicate that the execution of the motion control function block is not complete, and new output states (values) are expected to be

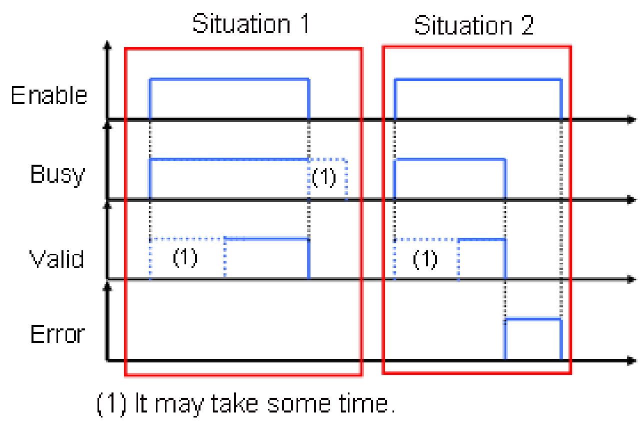
generated. The Busy output pin in a motion control function block is set to True when there is a transition in the Enable input pin's signal from low to high, and is set to True when the motion control function block is executed. When the Busy output pin is set to True, output states (values) still change.

- Characteristic of the Aborted output pin
  The Aborted output pin in a motion control function block is set to True when the execution of the motion control function block is interrupted by a command.
- Relation between the Enable input pin and the Valid output pin
  If the input pin that a motion control function block has is the Enable input pin, the motion control function block uses the Busy output pin to indicate whether output data/states are valid. The Valid output pin is set to True only when the Enable input pin is set to true or output data/state are valid. If an error occurs in a motion control function block, output data/states will not be valid, and the Valid output pin will be set to False. The Valid output pin in a motion control function block will not be reset until the error occurring in the motion control function block is eliminated, and output data/states become valid.

### 5.8.2 Timing Diagram for Input/Output Pins




Situation 1: The execution of the motion control function block is interrupted.
Situation 2: An error occurs in the motion control function block.
Situation 3: The execution of the motion control function block is complete normally.

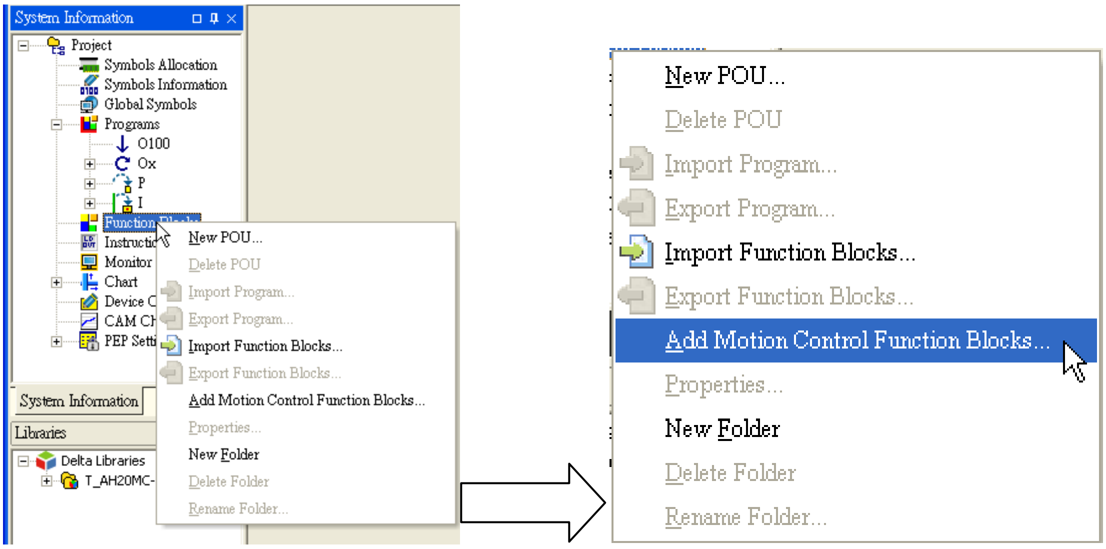



Situation 1: The motion control function block is executed normally.
Situation 2: An error occurs in the motion control function block.

## 5.8.3 Introducing the Use of PMSoft

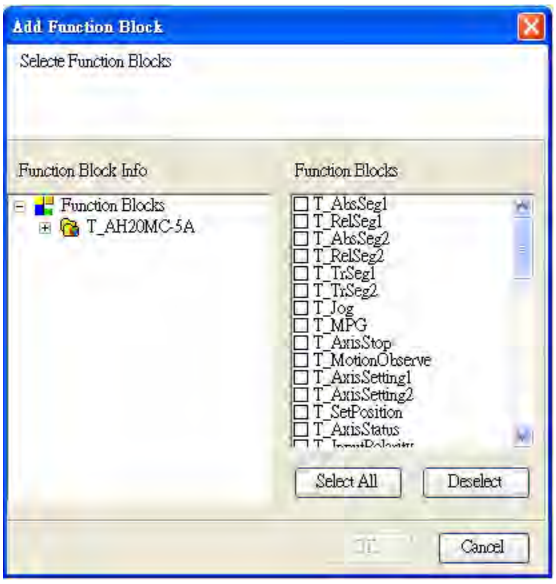
The use of the motion control function blocks in PMSoft is introduced below.

(1) Right-click **Function Blocks** in the system information area in PMSoft.



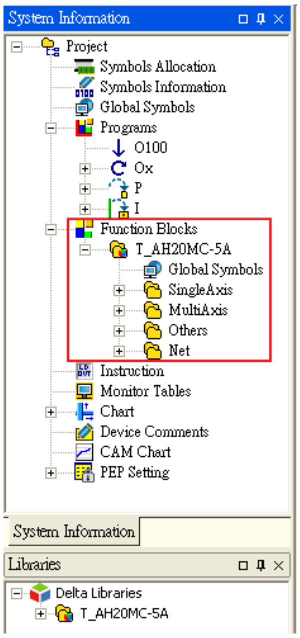

Click **Add Motion Control Function Blocks...** on the context menu.

(2) The **Add Function Block** window appears.

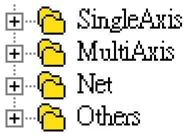



Users can select motion control function blocks in the **Add Function Block** window. If the users click **Select All**, all the motion control function blocks in the **Add Function Block** window will be selected. After users select motion control function blocks, they have to click **OK**.

(3) After the users click **OK**, the motion control function blocks selected in the **Add Function Block** window will be automatically added to **Function Blocks** in the system information area.

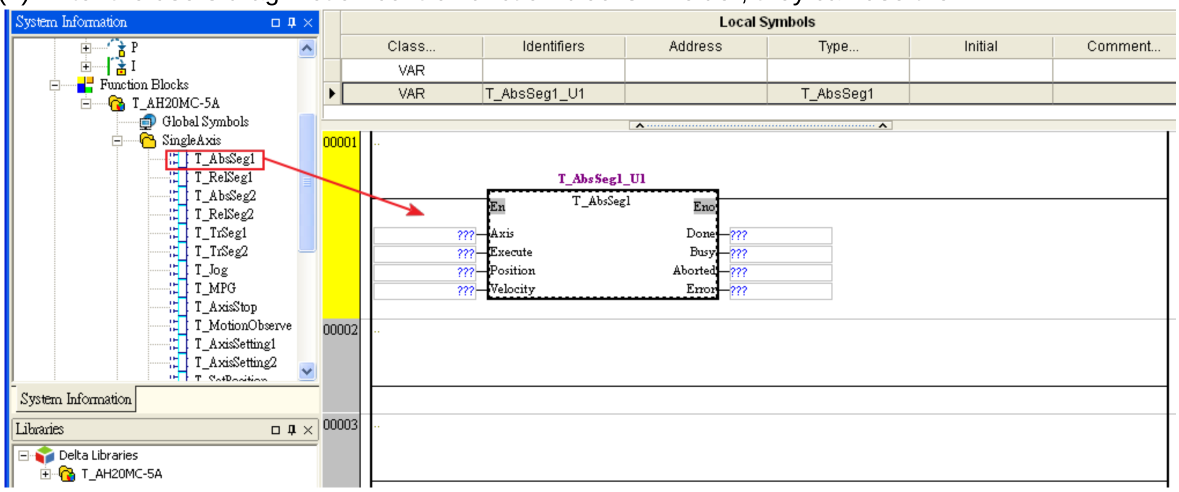



- The folders added to **Function Blocks** are shown below.




- Definitions of the folders
  - SingleAxis: Uniaxial motion (Uniaxial point-to-point motion, electronic gear synchronization, and electronic cam synchronization)
  - MultiAxis: Multi-axis motion (G-code execution, multi-axis linear interpolation)
  - Net: Communication (DMCNET and Ethernet)
  - Others: Other functions (using a memory card, counting pulses, measuring time, high-speed comparison, high-speed capture, and setting interrupts)

(4) After the users drag motion control function blocks in folder, they can use them.



## 5.9 Delta-defined Parameter Table

Delta-defined parameters are for input pins in Delta motion control function blocks. Users can directly use Delta-defined parameters to operate motion control function blocks without having to know the descriptions of the input pins in the motion control function blocks. Delta-defined parameters are described below.

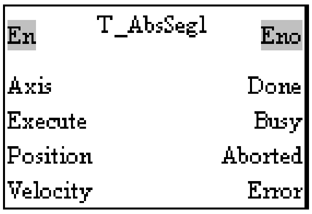
| Name | Type | Value | Motion control function block | Description |
|---|---|---|---|---|
| TRUE | BOOL | True | All motion control function blocks | Input pin |
| FALSE | BOOL | False | | Input pin |
| mcRising | BOOL | True | T_TrSeg2, T_TrSeg1, T_HomeReturn | Transition in DOG's signal from low to high |
| mcFalling | BOOL | False | | Transition in DOG's signal from high to low |
| mcPositive | BOOL | True | T_HomeReturn | Returning home in the positive direction |
| mcNegative | BOOL | False | | Returning home in the negative direction |
| mcSCurve | BOOL | True | T_AxisSetting2 | Speed curve: S curve |
| mcTrapezoid | BOOL | False | | Speed curve: Trapezoid curve |
| mcNC | BOOL | True | T_InputPolatiry | Normally-closed contact |
| mcNO | BOOL | False | | Normally-open contact |
| mc32bits | BOOL | True | T_DMCServoWrite | 32-bit value |
| mc16bits | BOOL | False | | 16-bit value |
| mcUp_Up | BOOL | True | T_HTmr | A high-speed timer becomes active when its signal goes from low to high. |
| mcUp_Down | BOOL | False | | A high-speed timer becomes active when its signal goes from high to low. |
| mcCmpSet | BOOL | True | T_Compare | An output is set when the condition of a comparison is met. |
| mcCmpRst | BOOL | False | | An output is reset when the condition of a comparison is met. |
| mcMotor | WORD | 0 | T_AxisSetting2 | Motor unit |
| mcMachine | WORD | 1 | | Mechanical unit |
| mcComp | WORD | 2 | | Compound unit |
| mcUD | WORD | 0 | T_AxisSetting2, T_HCnt | Counting up/down |
| mcPD | WORD | 1 | | Pulses+Directions |
| mcAB | WORD | 2 | | A/B-phase pulses |
| mc4AB | WORD | 3 | | Four times the frequency of A/B-phase pulses |
| mcSD_M | WORD | 0 | T_SDDevRead | Using M devices |
| mcSD_D | WORD | 5 | | Using D devices |
| mcSD_W | WORD | 6 | | Using W devices |

| Name | Type | Value | Motion control function block | Description |
|---|---|---|---|---|
| IntTimer | WORD | 0 | T_Interrupt | An interrupt signal is triggered by a time interval. |
| IntX8 | WORD | 1 | | The source of an interrupt signal is X0.8. |
| IntX9 | WORD | 2 | | The source of an interrupt signal is X0.9. |
| IntX10 | WORD | 3 | | The source of an interrupt signal is X0.10. |
| IntX11 | WORD | 4 | | The source of an interrupt signal is X0.11. |
| IntX12 | WORD | 5 | | The source of an interrupt signal is X0.12. |
| IntX13 | WORD | 6 | | The source of an interrupt signal is X0.13. |
| IntX14 | WORD | 7 | | The source of an interrupt signal is X0.14. |
| IntX15 | WORD | 8 | | The source of an interrupt signal is X0.15. |
| mcCmpAxis1 | WORD | 0 | T_Compare | The source of a comparison is the present position of the first axis. |
| mcCmpAxis2 | WORD | 1 | | The source of a comparison is the present position of the second axis. |
| mcCmpAxis3 | WORD | 2 | | The source of a comparison is the present position of the third axis. |
| mcCmpAxis4 | WORD | 3 | | The source of a comparison is the present position of the fourth axis. |
| mcCmpC200 | WORD | 4 | | The source of a comparison is the value of C200. |
| mcCmpC204 | WORD | 5 | | The source of a comparison is the value of C204. |
| mcCmpC208 | WORD | 6 | | The source of a comparison is the value of C208. |
| mcCmpC212 | WORD | 7 | | The source of a comparison is the value of C212. |
| mcCmpY8 | WORD | 0 | T_Compare | The device used for a comparison is Y0.8. |
| mcCmpY9 | WORD | 1 | | The device used for a comparison is Y0.9. |
| mcCmpY10 | WORD | 2 | | The device used for a comparison is Y0.10. |
| mcCmpY11 | WORD | 3 | | The device used for a comparison is Y0.11. |
| mcCmpRstC200 | WORD | 4 | | The device used for a comparison is C200. |
| mcCmpRstC204 | WORD | 5 | | The device used for a comparison is C204. |
| mcCmpRstC208 | WORD | 6 | | The device used for a comparison is C208. |
| mcCmpRstC212 | WORD | 7 | | The device used for a comparison is C212. |

| Name | Type | Value | Motion control function block | Description |
|---|---|---|---|---|
| mcCapAxis1 | WORD | 1 | T_Capture | The source of capture is the present position of the first axis. |
| mcCapAxis2 | WORD | 2 | | The source of capture is the present position of the second axis. |
| mcCapAxis3 | WORD | 3 | | The source of capture is the present position of the third axis. |
| mcCapAxis4 | WORD | 4 | | The source of capture is the present position of the fourth axis. |
| mcCapC200 | WORD | 7 | | The source of capture is the value of C200. |
| mcCapC204 | WORD | 8 | | The source of capture is the value of C204. |
| mcCapC208 | WORD | 9 | | The source of capture is the value of C208. |
| mcCapC212 | WORD | 10 | | The source of capture is the value of C212. |
| mcCapX0 | WORD | 0 | T_Capture | The source of a capture signal is X0.0. |
| mcCapX1 | WORD | 1 | | The source of a capture signal is X0.1. |
| mcCapX2 | WORD | 2 | | The source of a capture signal is X0.2. |
| mcCapX3 | WORD | 3 | | The source of a capture signal is X0.3. |
| mcCapX8 | WORD | 8 | | The source of a capture signal is X0.8. |
| mcCapX9 | WORD | 9 | | The source of a capture signal is X0.9. |
| mcCapX10 | WORD | 10 | | The source of a capture signal is X0.10. |
| mcCapX11 | WORD | 11 | | The source of a capture signal is X0.11. |
| mcCapX12 | WORD | 12 | | The source of a capture signal is X0.12. |
| mcCapX13 | WORD | 13 | | The source of a capture signal is X0.13. |
| mcCapX14 | WORD | 14 | | The source of a capture signal is X0.14. |
| mcCapX15 | WORD | 15 | | The source of a capture signal is X0.15. |

# 5.10 Uniaxial Motion Control Function Blocks

## 5.10.1 Absolute Single-speed Motion




1. Motion control function block
   The motion control function block T_AbsSeg1 is used to start absolute single-speed motion. The value of the Axis input pin indicates an axis number, and the value of the Velocity input pin indicates the speed of single-speed motion. The value of the Position input pin indicates the target position of single-speed motion, and the target position is an absolute position.
2. Input pins/Output pins

| **Input pin** | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | Motion is started when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Position | Absolute position | DWORD | K-2,147,483,648~ K2,147,483,647 | The value of the Position input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity | Target speed | DWORD | K1~K2,147,483,647 | When the motion control function block is executed, the value of the Velocity input pin is updated repeatedly. |
| **Output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal when motion is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |

| Output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The axis specified is in motion before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

The number of pulses is a unit for the Position input pin, and the number of pulses per second is a unit for the Velocity input pin. Users can change the unit used by means of the motion control function block T_AxisSetting2.

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

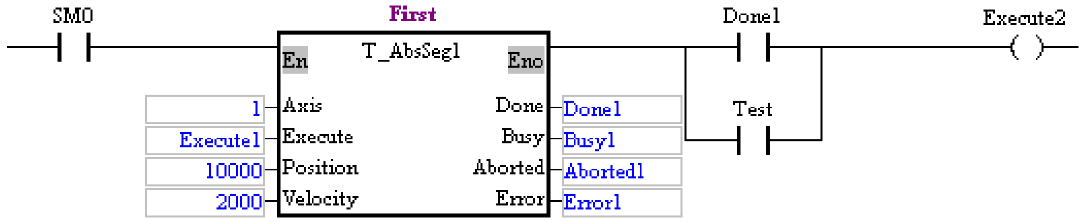
| Error | Troubleshooting |
|---|---|
| The motion control function block conflicts with other motion control function blocks. | Make sure that other uniaxial motion control function blocks are not started or the execution of other uniaxial motion control function blocks is complete before the motion control function block is started. |

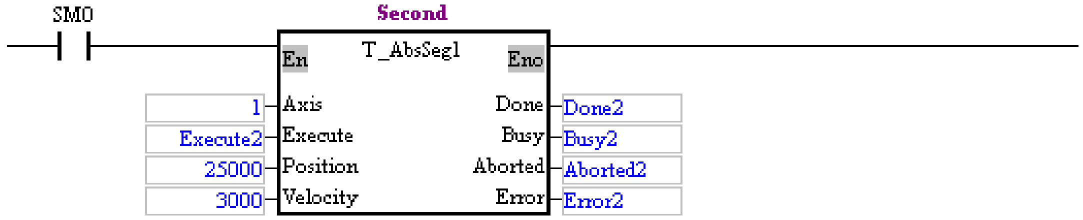
4. Example

Purposes:

- After the first single-speed motion is complete, the second single-speed motion will be executed.
- The second single-speed motion is executed before the execution of the first single-speed motion is complete.

The motion control function block named FIRST is set so that the first axis moves at a speed of 2,000 pulses per second, and moves for 10,000 pulses. The motion control function block named SECOND is set so that the first axis moves at a speed of 3,000 pulses per second, and moves for 15,000 pulses.




- After the first single-speed motion is complete, the second single-speed motion will be executed.

  Steps:

  (a) Set Execute1 to True.

  (b) Wait for a transition in Done2's signal from low to high or a transition in Error2's signal from low to high.

- The second single-speed motion is executed before the execution of the first single-speed motion is complete.

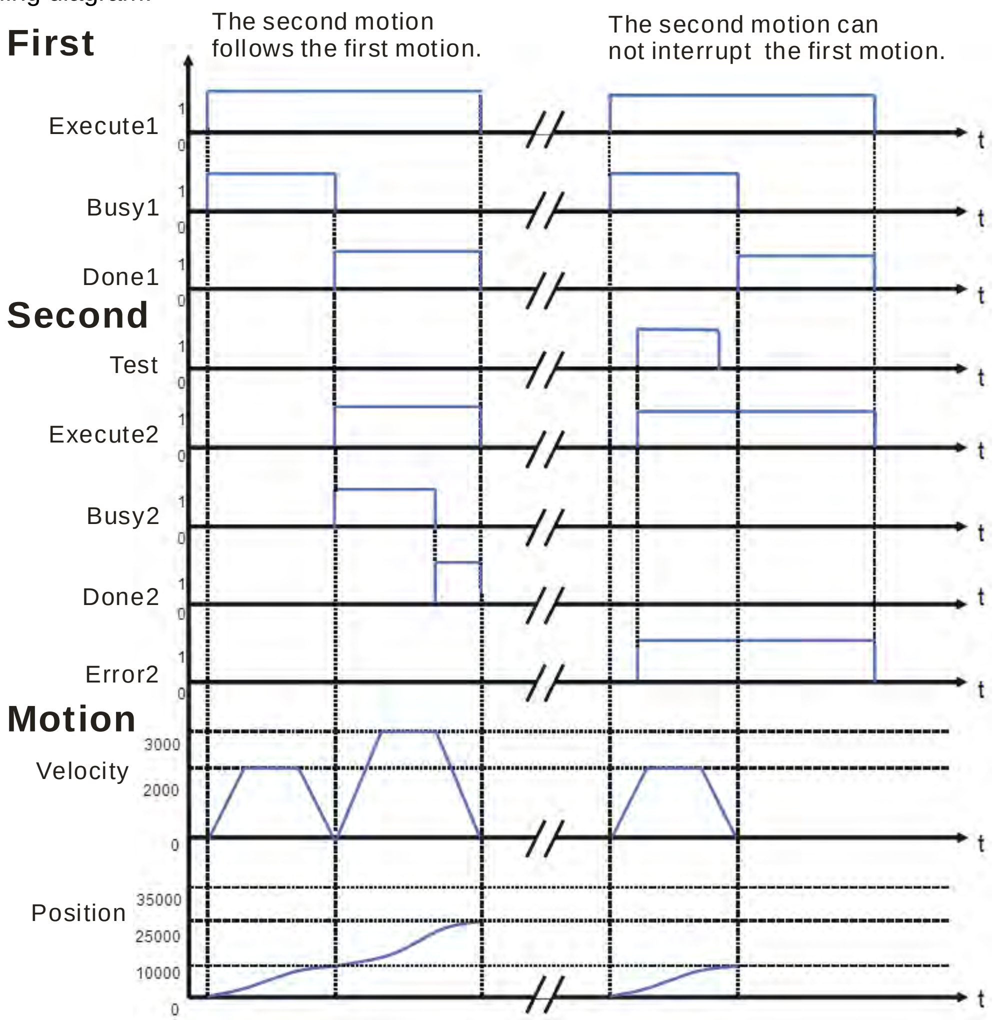
  Steps:

  (a) Set Execute1 to True.

  (b) Set Test to ON when Busy1 is set to True.

  (c) Wait for a transition in Done2's signal from low to high or a transition in Error2's signal from low to high.

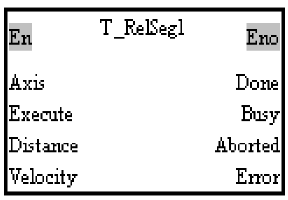
Timing diagram:




- After the first single-speed motion is complete, the second single-speed motion will be executed.
  After the execution of the motion control function block named FIRST is complete, the motion control function block named SECOND will be executed. The first axis moves for 25,000 pulses.
- The second single-speed motion is executed before the execution of the first single-speed motion is complete.
  When Error2 is set to True, the first axis moves for 10,000 pulses. The motion control function block named SECOND is invalid.

5. Modules which are supported
   The motion control function block T_AbsSeg1 supports AH05PM-5A, AH15PM-5A, AH10PM-5A, and AH20MC-5A.

## 5.10.2 Relative Single-speed Motion




1. Motion control function block
   The motion control function block T_RelSeg1 is used to start relative single-speed motion. The

value of the Axis input pin indicates an axis number, and the value of the Velocity input pin indicates the speed of single-speed motion. The value of the Distance input pin indicates the distance for which single-speed motion moves, and the distance is a relative distance.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | Motion is started when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Distance | Relative distance | DWORD | K-2,147,483,646~ K2,147,483,646 | The value of the Distance input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity | Target speed | DWORD | K1~K2,147,483,647 | When the motion control function block is executed, the value of the Velocity input pin is updated repeatedly. |
| **Output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | ● There is a transition in the Done output pin's signal when motion is complete. | ● There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |

| Output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The axis specified is in motion before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

The number of pulses is a unit for the Distance input pin, and the number of pulses per second is a unit for the Velocity input pin. Users can change the unit used by means of the motion control function block T_AxisSetting2.

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

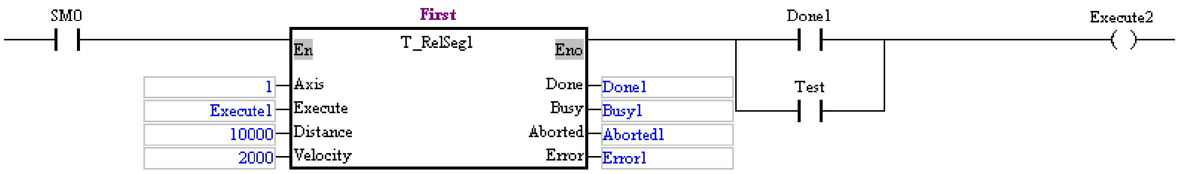
| Error | Troubleshooting |
|---|---|
| The motion control function block conflicts with other motion control function blocks. | Make sure that other uniaxial motion control function blocks are not started or the execution of other uniaxial motion control function blocks is complete before the motion control function block is started. |

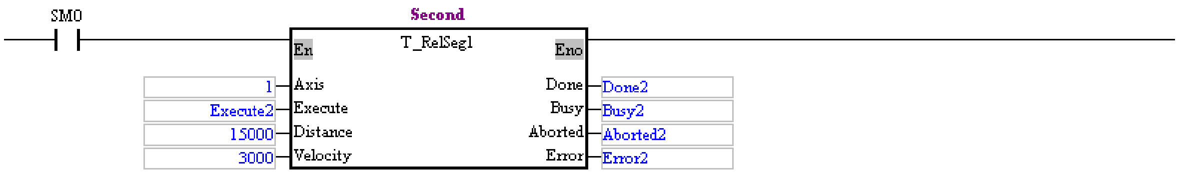
4. Example

Purposes:

- After the first single-speed motion is complete, the second single-speed motion will be executed.
- The second single-speed motion is executed before the execution of the first single-speed motion is complete.

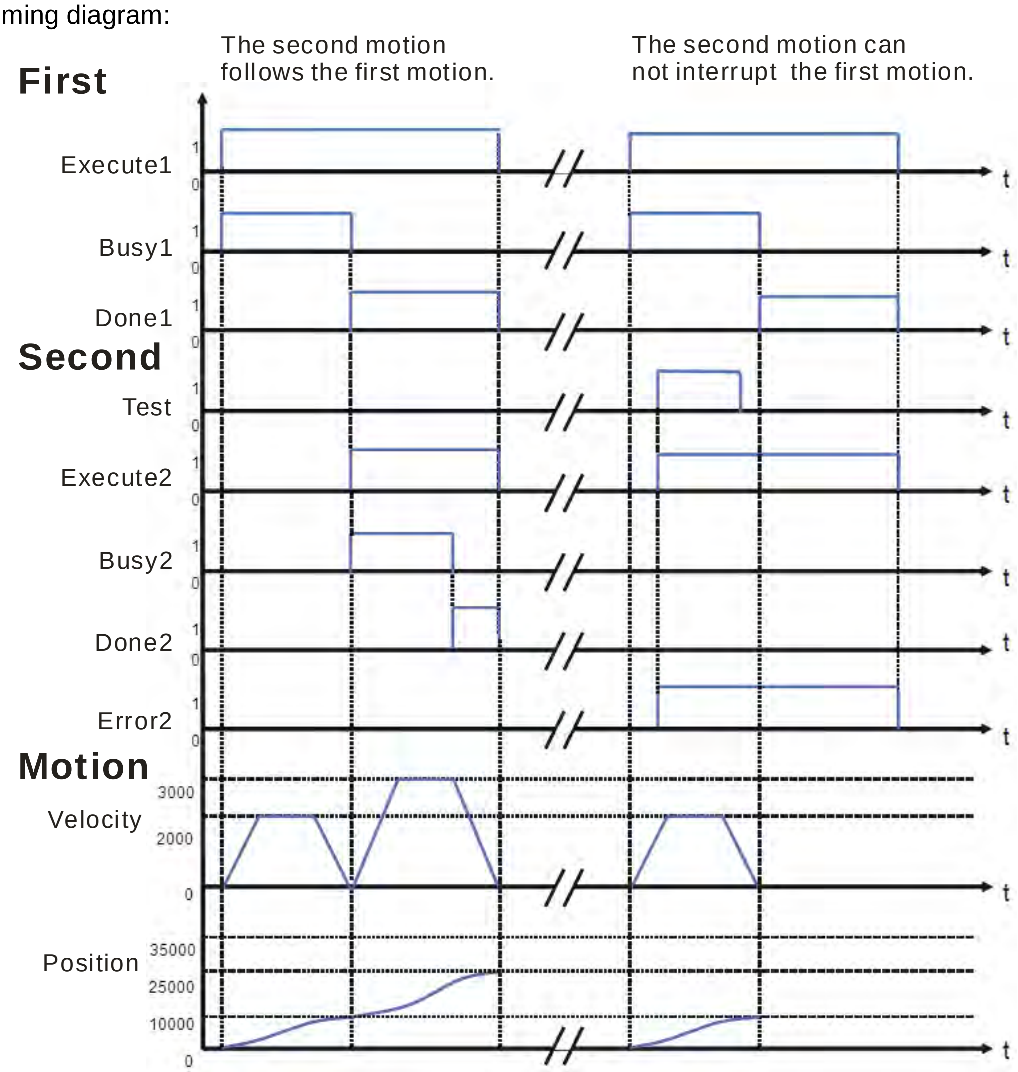
The motion control function block named FIRST is set so that the first axis moves at a speed of 2,000 pulses per second, and moves for 10,000 pulses. The motion control function block named SECOND is set so that the first axis moves at a speed of 3,000 pulses per second, and moves for 15,000 pulses.




- After the first single-speed motion is complete, the second single-speed motion will be executed.
  Steps:
  (a) Set Execute1 to True.
  (b) Wait for a transition in Done2's signal from low to high or a transition in Error2's signal from low to high.
- The second single-speed motion is executed before the execution of the first single-speed motion is complete.
  Steps:
  (a) Set Execute1 to True.
  (b) Set Test to ON when Busy1 is set to true.
  (c) Wait for a transition in Done2's signal from low to high or a transition in Error2's signal from low to high.

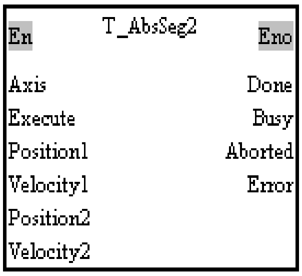
Timing diagram:




- After the first single-speed motion is complete, the second single-speed motion will be executed.
  When the motion control function block named FIRST is executed, the first axis moves for 10,000 pulses. After the execution of the motion control function block named FIRST is complete, the motion control function block named SECOND will be executed. When the motion control function block named SECOND is executed, the first axis moves for 15,000 pulses.
- The second single-speed motion is executed before the execution of the first single-speed motion is complete.
  When Error2 is set to True, the first axis moves for 10,000 pulses. The motion control function block named SECOND is invalid.

5. Modules which are supported
   The motion control function block T_RelSeg1 supports AH05PM-5A, AH15PM-5A, AH10PM-5A, and AH20MC-5A.

### 5.10.3 Absolute Two-speed Motion




1. Motion control function block
   The motion control function block T_AbsSeg2 is used to start absolute two-speed motion. The value of the Axis input pin indicates an axis number, and the value of the Velocity1 input pin indicates the speed of the first motion. The value of the Position1 input pin indicates the target position of the first motion, and the target position is an absolute position. The value of the Velocity2 input pin indicates the speed of the second motion. The value of the Position2 input pin indicates the target position of the second motion, and the target position is an absolute position.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | Motion is started when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Position1 | Absolute position of the first motion | DWORD | K-2,147,483,648~ K2,147,483,647 | The value of the Position1 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity1 | Target speed of the first motion | DWORD | K1~K2,147,483,647 | The value of the Velocity1 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Position2 | Absolute position of the second motion | DWORD | K-2,147,483,648~ K2,147,483,647 (If the value of the Position1 input pin is greater than 0, the value of the Position2 input pin must be greater than or equal to the value of the Position1 input pin. If the value of the Position1 input pin is less than or equal to 0, the value of the Position2 input pin must be less than or equal to the value of the Position1 input pin.) | The value of the Position2 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity2 | Target speed of the second motion | DWORD | K1~K2,147,483,647 | The value of the Velocity2 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **Output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal when motion is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |

| Output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | ● The execution of the motion control function block is interrupted by a command. | ● There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect.<br>● The axis specified is in motion before the motion control function block is executed. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

The number of pulses is a unit for the Position1 input pin/the Position2 input pin, and the number of pulses per second is a unit for the Velocity1 input pin/the Velocity2 input pin. Users can change the unit used by means of the motion control function block T_AxisSetting2.

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

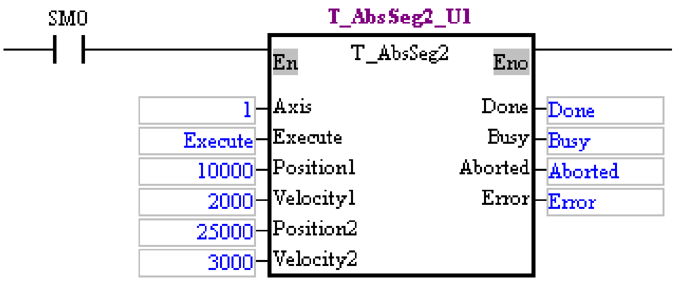
| Error | Troubleshooting |
|---|---|
| The motion control function block conflicts with other motion control function blocks. | Make sure that other uniaxial motion control function blocks are not started or the execution of other uniaxial motion control function blocks is complete before the motion control function block is started. |

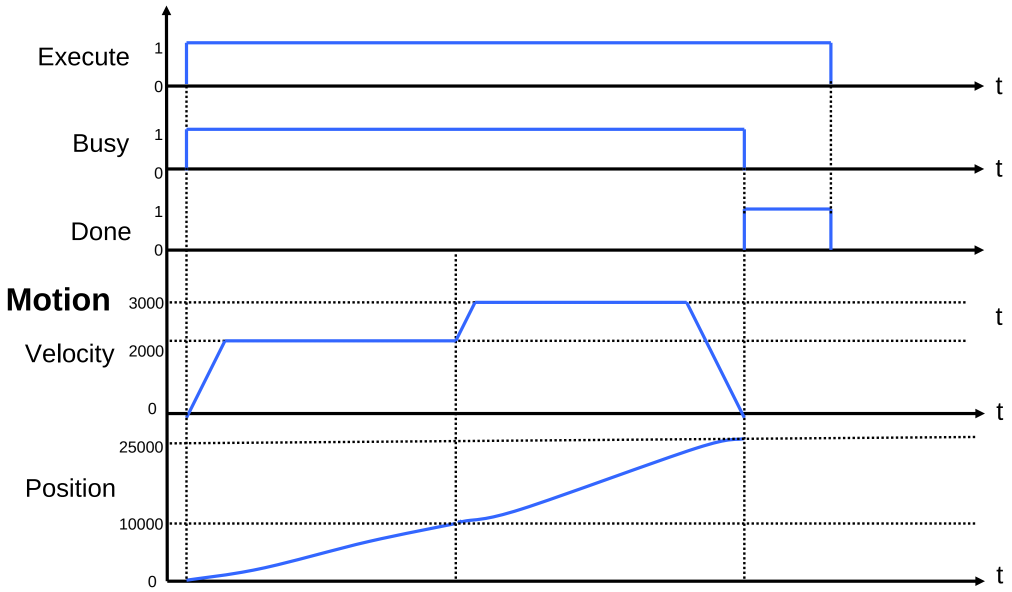
4. Example

   Purposes:

   - The motion control function block T_AbsSeg2 is used to start absolute two-speed motion of an axis.

     The first motion is set so that the first axis moves at a speed of 2,000 pulses per second, and moves for 10,000 pulses. The second motion is set so that the first axis moves at a speed of 3,000 pulses per second, and moves for 15,000 pulses.



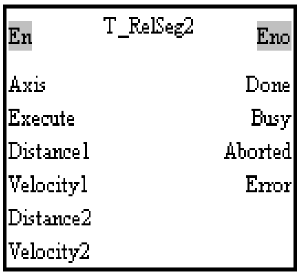




   After the motion control function block is started, the first axis moves for 10,000 pulses at a speed of 2,000 pulses per second, and moves for 15,000 pulses at a speed of 3,000 pulses per second.

5. Modules which are supported

   The motion control function block T_AbsSeg2 supports AH05PM-5A, AH15PM-5A, AH10PM-5A, and AH20MC-5A.

## 5.10.4 Relative Two-speed Motion




1. Motion control function block
   The motion control function block T_RelSeg2 is used to start relative two-speed motion. The value of the Axis input pin indicates an axis number, and the value of the Velocity1 input pin indicates the speed of the first motion. The value of the Distance1 input pin indicates the distance for which the first motion moves, and the distance is a relative distance. The value of the Velocity2 input pin indicates the speed of the second motion. The value of the Distance2 input pin indicates the distance for which the second motion moves, and the distance is a relative distance.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | Motion is started when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Distance1 | Relative distance for which the first motion moves | DWORD | K-2,147,483,646~ K2,147,483,646 | The value of the Distance1 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity1 | Target speed of the first motion | DWORD | K1~K2,147,483,647 | The value of the Velocity1 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Distance2 | Relative distance for which the second motion moves | DWORD | K-2,147,483,646~ K2,147,483,646 (If the value of the Distance1 input pin is a positive value, the value of the Distance2 input pin must be a positive value. If the value of the Distance1 input pin is a negative value, the value of the Distance2 input pin must be a negative value.) | The value of the Distance2 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity2 | Target speed of the second motion | DWORD | K1~K2,147,483,647 | The value of the Velocity2 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **Output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal when motion is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |

5

| Output pin | | | | |
|---|---|---|---|---|
| Name | Function | Data type | Time when there is a transition in an output pin's signal from low to high | Time when there is a transition in an output pin's signal from high to low |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | ● The execution of the motion control function block is interrupted by a command. | ● There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect.<br>● The axis specified is in motion before the motion control function block is executed. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

The number of pulses is a unit for the Distance1 input pin/the Distance2 input pin, and the number of pulses per second is a unit for the Velocity1 input pin/the Velocity2 input pin. Users can change the unit used by means of the motion control function block T_AxisSetting2.

3. Troubleshooting

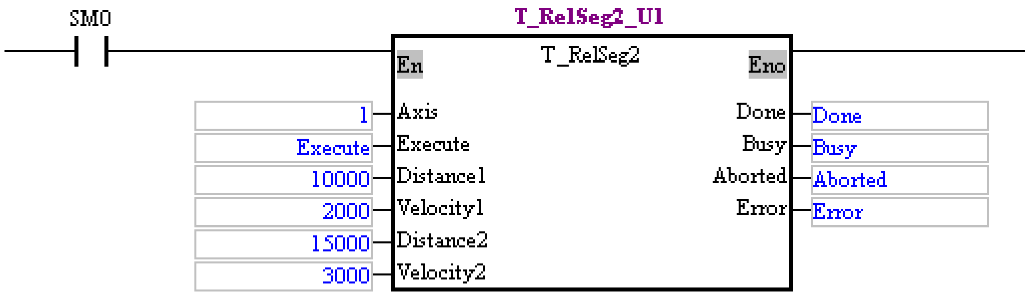
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

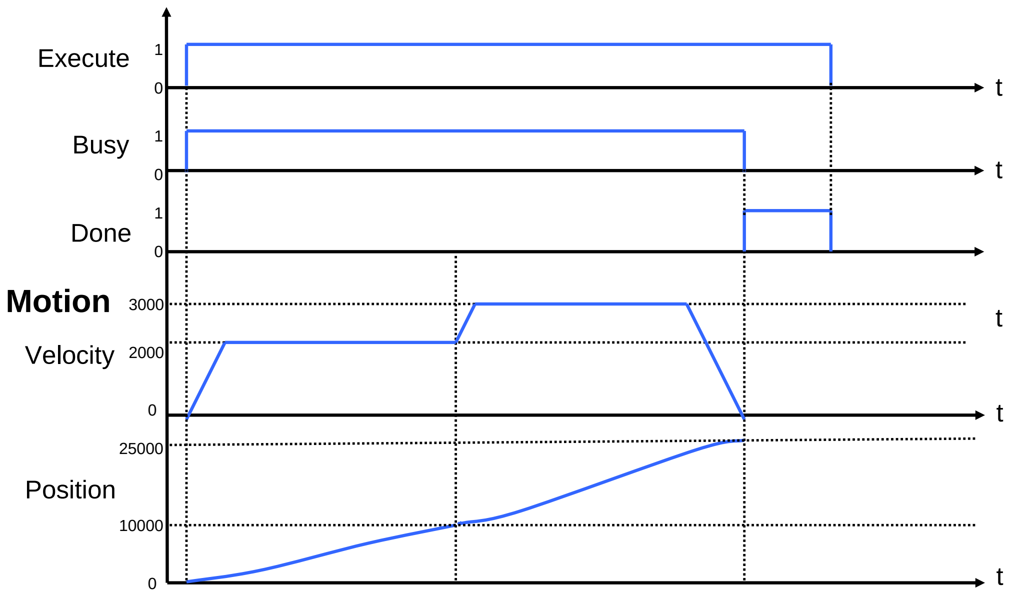
| Error | Troubleshooting |
|---|---|
| The motion control function block conflicts with other motion control function blocks. | Make sure that other uniaxial motion control function blocks are not started or the execution of other uniaxial motion control function blocks is complete before the motion control function block is started. |

4. Example

Purpose:

- The motion control function block T_AbsSeg2 is used to start relative two-speed motion of an axis.
  The first motion is set so that the first axis moves at a speed of 2,000 pulses per second, and moves for 10,000 pulses. The second motion is set so that the first axis moves at a speed of 3,000 pulses per second, and moves for 15,000 pulses.



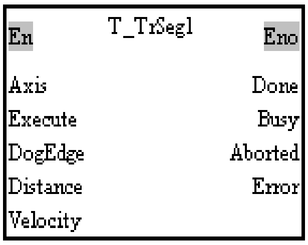




After the motion control function block is started, the first axis moves for 10,000 pulses at a speed of 2,000 pulses per second, and moves for 15,000 pulses at a speed of 3,000 pulses per second.

5. Modules which are supported

The motion control function block T_RelSeg2 supports AH05PM-5A, AH15PM-5A, AH10PM-5A, and AH20MC-5A.

## 5.10.5 Inserting Single-speed Motion




1. Motion control function block
   The motion control function block T_TrSeg1 is used to insert single-speed motion. The value of the Axis input pin indicates an axis number, and the value of the Velocity input pin indicates the speed of motion. The value of the DogEdge input pin indicates whether motion is triggered by a transition in DOG's signal from low to high or from high to low. The value of the Distance input pin indicates the distance for which motion moves, and the distance is a relative distance.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | Motion is started when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| DogEdge | Transition in DOG's signal from low to high or from high to low | BOOL | mcRising (True)/ mcFalling (False) | The value of the DogEdge input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Distance | Distance for which motion moves after a transition in DOG's signal from low to high or from high to low | DWORD | K-2,147,483,648~ K2,147,483,647 | The value of the Distance input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity | Target speed | DWORD | K1~K2,147,483,647 | The value of the Velocity input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| Output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal when motion is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |

| Output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The axis specified is in motion before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

The number of pulses is a unit for the Distance input pin, and the number of pulses per second is a unit for the Velocity input pin. Users can change the unit used by means of the motion control function block T_AxisSetting2. If the value of the DogEdge input pin is mcRising, motion will be triggered by a transition in DOG's signal from low to high. If the value of the DogEdge input pin is mcFalling, motion will be triggered by a transition in DOG's signal from high to low.

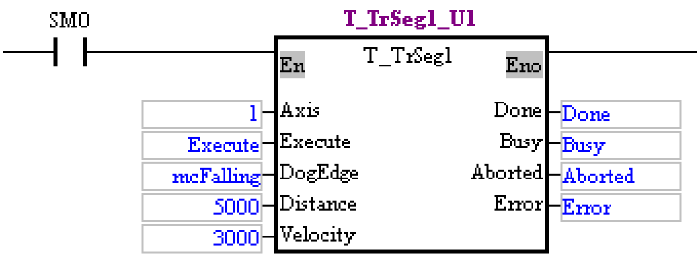
3. Troubleshooting

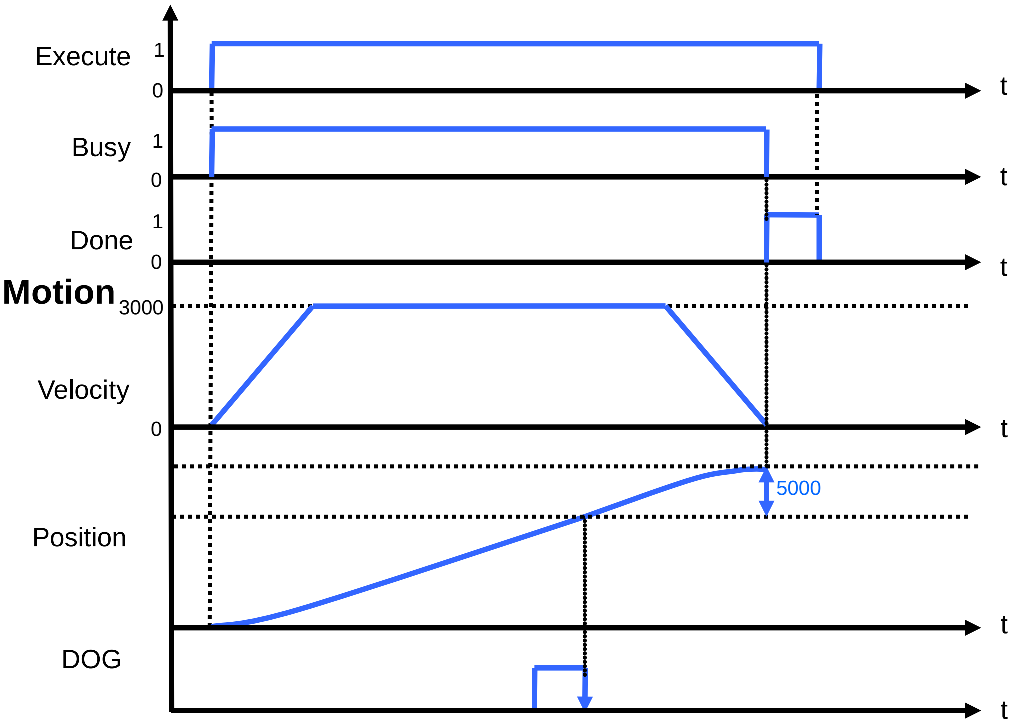
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |
| The motion control function block conflicts with other motion control function blocks. | Make sure that other uniaxial motion control function blocks are not started or the execution of other uniaxial motion control function blocks is complete before the motion control function block is started. |

4. Examples

Example 1:

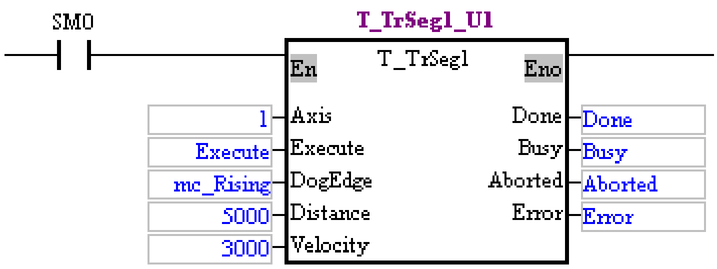
- The motion control function block T_TrSeg1 is used to insert single-speed motion which is triggered by a transition in DOG's signal from high to low.
  The motion control function block named T_TrSeg1_U1 is set so that the first axis moves at a speed of 3,000 pulses per second, and will move for 5,000 pulses after a transition in DOG's signal from high to low. After the first axis moves for 5,000 pulses, Done will be set to True.

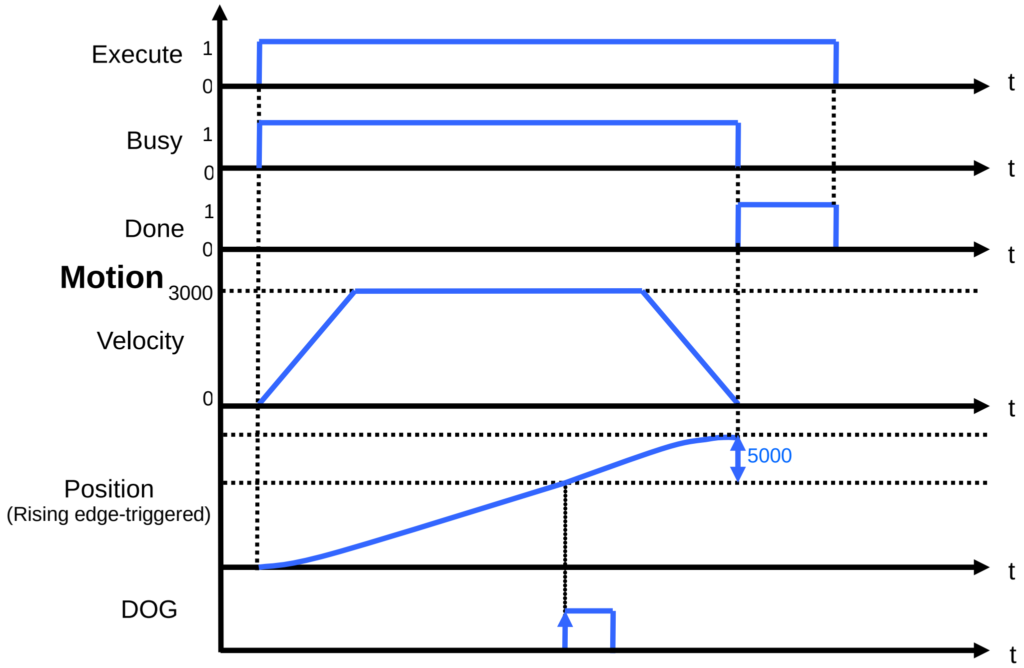





Example2:

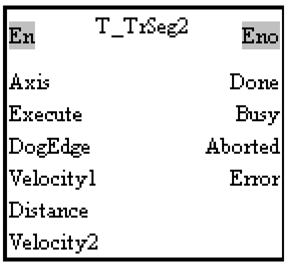
- The motion control function block T_TrSeg1 is used to insert single-speed motion which is triggered by a transition in DOG's signal from low to high.
  The motion control function block named T_TrSeg1_U1 is set so that the first axis moves at a speed of 3,000 pulses per second, and will move for 5,000 pulses after a transition in DOG's signal from low to high. After the first axis moves for 5,000 pulses, Done will be set to True.






5. Modules which are supported
   The motion control function block T_TrSeg1 supports AH05PM-5A, AH15PM-5A, AH10PM-5A, and AH20MC-5A.

## 5.10.6 Inserting Two-speed Motion




1. Motion control function block
   The motion control function block T_TrSeg2 is used to insert two-speed motion. The value of the Axis input pin indicates an axis number, and the value of the Velocity1 input pin indicates the speed of motion. The value of the DogEdge input pin indicates whether motion is triggered by a transition in DOG's signal from low to high or from high to low. The value of the Velocity2 input pin indicates the speed of the motion triggered by a transition in DOG's signal from low to high or from high to low. The value of the Distance input pin indicates the distance for which motion moves, and the distance is a relative distance.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | Motion is started when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| DogEdge | Transition in DOG's signal from low to high or from high to low | BOOL | mcRising (True)/ mcFalling (False) | The value of the DogEdge input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity1 | Target speed before a transition in DOG's signal from low to high or from high to low | DWORD | K1~K2,147,483,647 | The value of the Velocity1 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Distance | Distance for which motion moves after a transition in DOG's signal from low to high or from high to low | DWORD | K-2,147,483,648~ K2,147,483,647 | The value of the Distance input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity2 | Target speed after a transition in DOG's signal from low to high or from high to low | DWORD | K1~K2,147,483,647 | The value of the Velocity2 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **Output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal when motion is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |

| Output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The axis specified is in motion before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

The number of pulses is a unit for the Distance input pin, and the number of pulses per second is a unit for the Velocity1 input pin/the Velocity2 input pin. Users can change the unit used by means of the motion control function block T_AxisSetting2. If the value of the DogEdge input pin is mcRising, motion will be triggered by a transition in DOG's signal from low to high. If the value of the DogEdge input pin is mcFalling, motion will be triggered by a transition in DOG's signal from high to low.

3. Troubleshooting

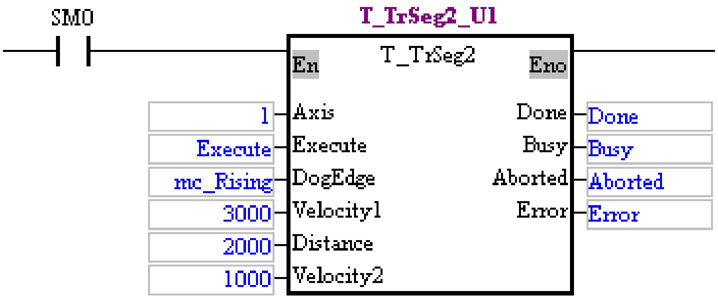
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

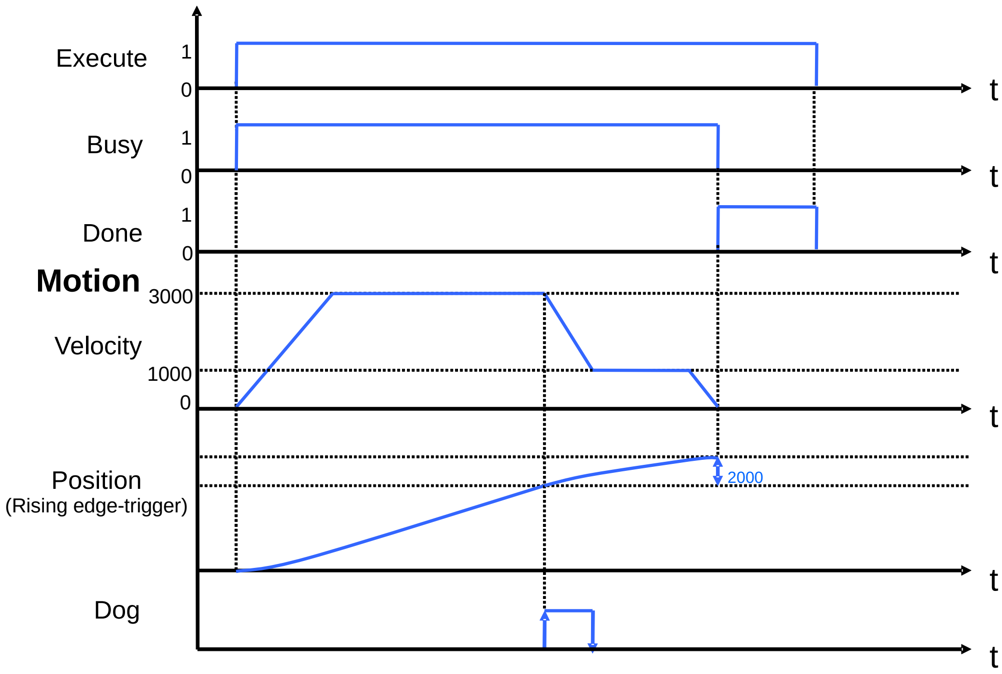
| Error | Troubleshooting |
|---|---|
| The motion control function block conflicts with other motion control function blocks. | Make sure that other uniaxial motion control function blocks are not started or the execution of other uniaxial motion control function blocks is complete before the motion control function block is started. |

4. Example

The motion control function block T_TrSeg2 is used to insert two-speed motion which is triggered by a transition in DOG’s signal from low to high.

The motion control function block named T_TrSeg2_U1 is set so that the first axis moves at a speed of 3,000 pulses per second, and will move for 2,000 pulses at a speed of 1,000 pulses per second after a transition in DOG’s signal from low to high.



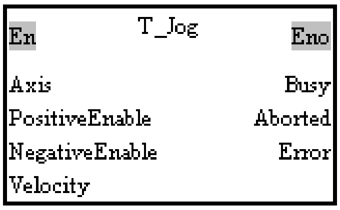




After the first axis moves for 2,000 pulses, Done will be set to True.

5. Modules which are supported

The motion control function block T_TrSeg2 supports AH05PM-5A, AH15PM-5A, AH10PM-5A, and AH20MC-5A.

### 5.10.7 JOG Motion




1. Motion control function block
   The motion control function block T_Jog is used to start JOG motion. The value of the Axis input pin indicates an axis number, and the value of the Velocity input pin indicates the speed of JOG motion. If the value of the PositiveEnable input pin is set to True, positive JOG motion will be started. If the value of the NegativeEnable input pin is set to True, negative JOG motion will be started.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| PositiveEnable | Enabling positive JOG motion | BOOL | True/False | ● If the PositiveEnable input pin and the NegativeEnable input pin are set to True simultaneously, positive JOG motion will be enabled, and the NegativeEnable input pin will be reset to False.<br>● If the PositiveEnable input pin is set to True after the NegativeEnable input pin is set to True, the NegativeEnable input pin will be reset to False, the negative JOG motion will stop, and the positive JOG motion will be enabled. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| NegativeEnable | Enabling negative JOG motion | BOOL | True/False | ● If the PositiveEnable input pin and the NegativeEnable input pin are set to True simultaneously, positive JOG motion will be enabled, and the NegativeEnable input pin will be reset to False.<br>● If the NegativeEnable input pin is set to True after the PositiveEnable input pin is set to True, the PositiveEnable input pin will be reset to False, the positive JOG motion will stop, and the negative JOG motion will be enabled. |
| Velocity | Target speed | DWORD | K1~K2,147,483,647 | ● When the motion control function block is executed, the value of the Velocity input pin is updated repeatedly. |
| **Output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the PositiveEnable input pin's signal from low to high or when there is a transition in the NegativeEnable input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when motion stops.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |

| Output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the PositiveEnable input pin's signal from high to low or when there is a transition in the NegativeEnable input pin's signal from high to low.<br>• If the PositiveEnable input pin and the NegativeEnable are set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The axis specified is in motion before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the PositiveEnable input pin's signal from high to low or when there is a transition in the NegativeEnable input pin's signal from high to low. |

The number of pulses per second is a unit for the Velocity input pin. Users can change the unit used by means of the motion control function block T_AxisSetting2.

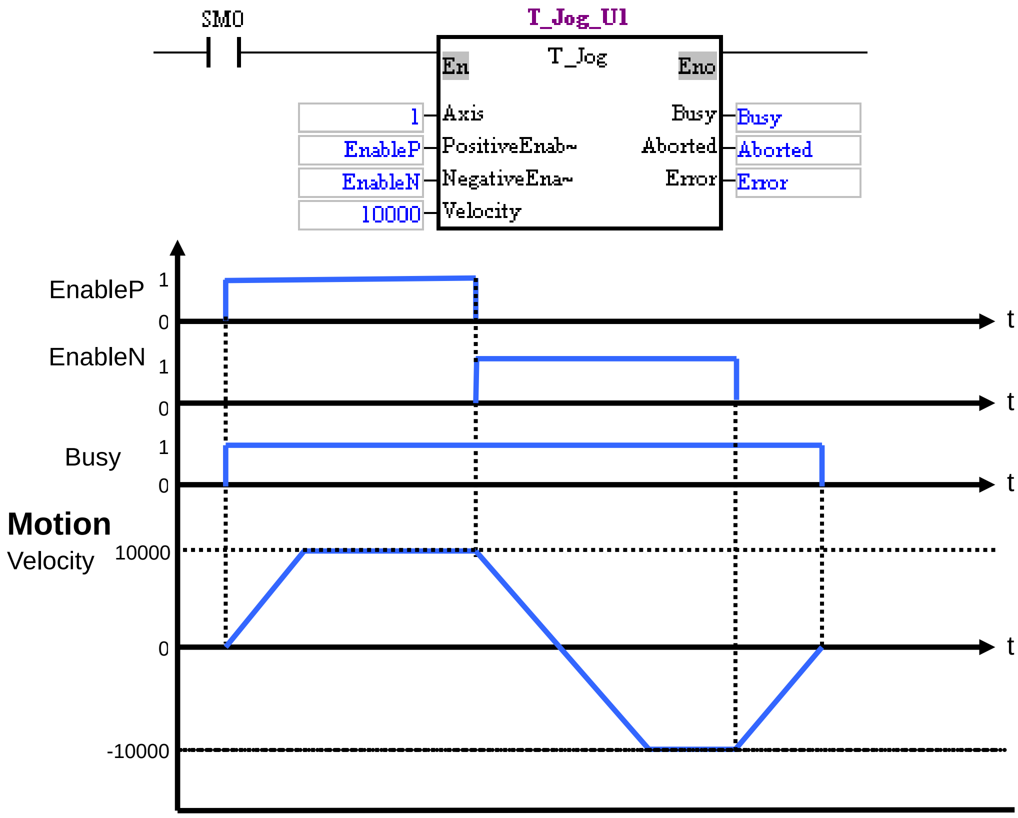
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |
| The motion control function block conflicts with other motion control function blocks. | Make sure that other uniaxial motion control function blocks are not started or the execution of other uniaxial motion control function blocks is complete before the motion control function block is started. |

4. Example

The motion control function block T_Jog is used to start JOG motion. Positive JOG motion is enabled by EnableP, and negative JOG motion is enabled by EnableN.

The first axis moves at a speed of 10,000 pulses per second. If EnableP is set to 1, the first axis will move in the positive direction. If EnableN is set to 1, the first axis will move in the negative direction.

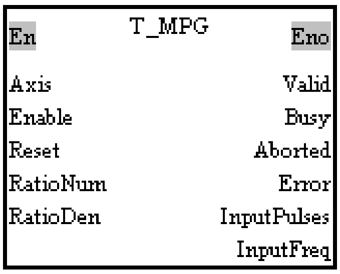



When EnableP is set to 1, the first axis moves at a speed of 10,000 pulses per second in the positive direction. When EnableN is set to 1, the first axis moves at a speed of 10,000 pulses per second in the negative direction. When EnableP and EnableN are not set to 1, the first axis stops moving.

5. Modules which are supported

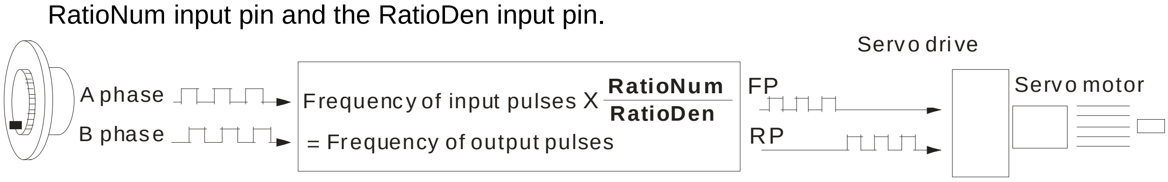
   The motion control function block T_Jog supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.10.8 Manual Pulse Generator Mode




1. Motion control function block

   The motion control function block T_MPG is used to enable a manual pulse generator mode. The value of the Axis input pin indicates an axis number. The motion of the axis specified follows the operation of a manual pulse generator. The relation between the position of the axis specified and the input pulses generated by the manual pulses used is determined by the RatioNum input pin and the RatioDen input pin.



Please refer to Chapter 2 for more information about wiring a manual pulse generator.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | Manual pulse generator mode | BOOL | True/False | - |
| Reset | Resetting the manual pulse generator used | BOOL | True/False | The value of the Reset input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| RatioNum | Numerator of an electronic gear ratio | WORD | K0~K32,767 | When the motion control function block is executed, the value of the RatioNum input pin is updated repeatedly. |
| RatioDen | Denominator of an electronic gear ratio | WORD | K1~K32,767 | When the motion control function block is executed, the value of the RatioDen input pin is updated repeatedly. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | ● There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Valid output pin's signal from high to low when motion stops.<br>● There is a transition in the Valid output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Valid output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |

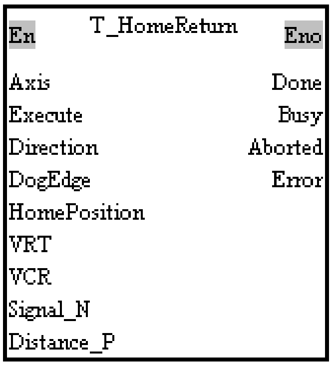
| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low.<br>• If the Enable input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The axis specified is in motion before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| **Value of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Update** |
| InputPulses | Number of pulses generated by the manual pulse generator used | DWORD | K-2,147,483,648~ K2,147,483,647 | When the motion control function block is executed, the value of the InputPulses output pin is updated repeatedly. |
| InputFreq | Frequency of pulses generated by the manual pulses generator used | DWORD | K0~K2,147,483,647 | When the motion control function block is executed, the value of the InputFreq output pin is updated repeatedly. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |
| The motion control function block conflicts with other motion control function blocks. | Make sure that other uniaxial motion control function blocks are not started or the execution of other uniaxial motion control function blocks is complete before the motion control function block is started. |

4. Modules which are supported
The motion control function block T_MPG supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

### 5.10.9 Returning Home




1. Motion control function block
The motion control function block T_HomeReturn is used to start motion of returning home. The value of the Axis input pin indicates an axis number, and the value of the Direction input pin indicates whether the axis specified returns home in the positive direction or in the negative direction. The value of the VRT input pin indicates the speed at which the axis specified returns home. The value of the DogEdge input pin indicates whether motion is triggered by a transition in DOG's signal from low to high or from high to low. The value of the VCR input pin indicates the speed to which the speed of the axis specified decreases. The value of the Signal_N input pin is the number of zero pulses. The value of the Distance_P is the number of supplementary pulses needed. After motion of returning home is complete, the value of the HomePosition input pin will be taken as the present position of the axis specified. Please refer to section 7.6 for more information about the normal mode of returning home.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K6 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | Motion is started when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Direction | Direction in which the axis specified returns home | BOOL | mcNegative (False)/ mcPositive (True) | The value of the Direction input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| DogEdge | Transition in DOG's signal from low to high or from high to low | BOOL | mcFalling (False)/ mcRising (True) | The value of the DogEdge input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| HomePosition | Home position | DWORD | K-2,147,483,648~ K2,147,483,647 | The value of the HomePosition input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| VRT | Speed at which the axis specified returns home | DWORD | K1~K1000000 Vbias<VRT≦Vmax | The value of the VRT input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| VCR | Speed to which the speed of the axis specified decreases | DWORD | K1~VRT | The value of the VCR input pin is valid when there is a transition in the Execute input pin's signal form low to high. |
| Signal_N | Number of zero pulses | WORD | K0~K32,767 (Only applicable to the first axis~the fourth axis) | The value of the Signal_N input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Distance_P | Number of supplementary pulses | WORD | K-32768~K32,767 | The value of the Distance_P input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

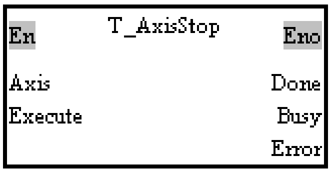
| | | | State of output pin | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when motion of returning home is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The axis specified is in motion before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |
| The motion control function block conflicts with other motion control function blocks. | Make sure that other uniaxial motion control function blocks are not started or the execution of other uniaxial motion control function blocks is complete before the motion control function block is started. |

4. Modules which are supported
   The motion control function block T_HomeReturn supports AH05PM-5A, AH15PM-5A, and AH10PM-5A.

## 5.10.10 Stopping Uniaxial Motion




1. Motion control function block
   The motion control function block T_AxisStop is used to stop the motion of the axis specified. The value of the Axis input pin indicates an axis number.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| Name | Function | Data type | Setting value | Time when a value is valid |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin’s signal from low to high. |
| Execute | Motion is stopped when there is a transition in the Execute input pin’s signal from low to high. | BOOL | True/False | - |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when the stopping of the motion of the axis specified is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The motion of the axis specified is not uniaxial motion, gear motion, or cam motion. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

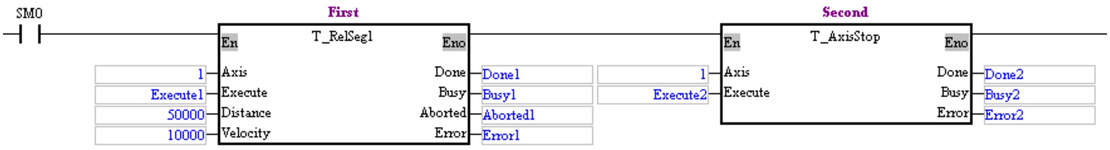
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |
| The motion control function block conflicts with other motion control function blocks. | Make sure that other uniaxial motion control function blocks are not started or the execution of other uniaxial motion control function blocks is complete before the motion control function block is started. |

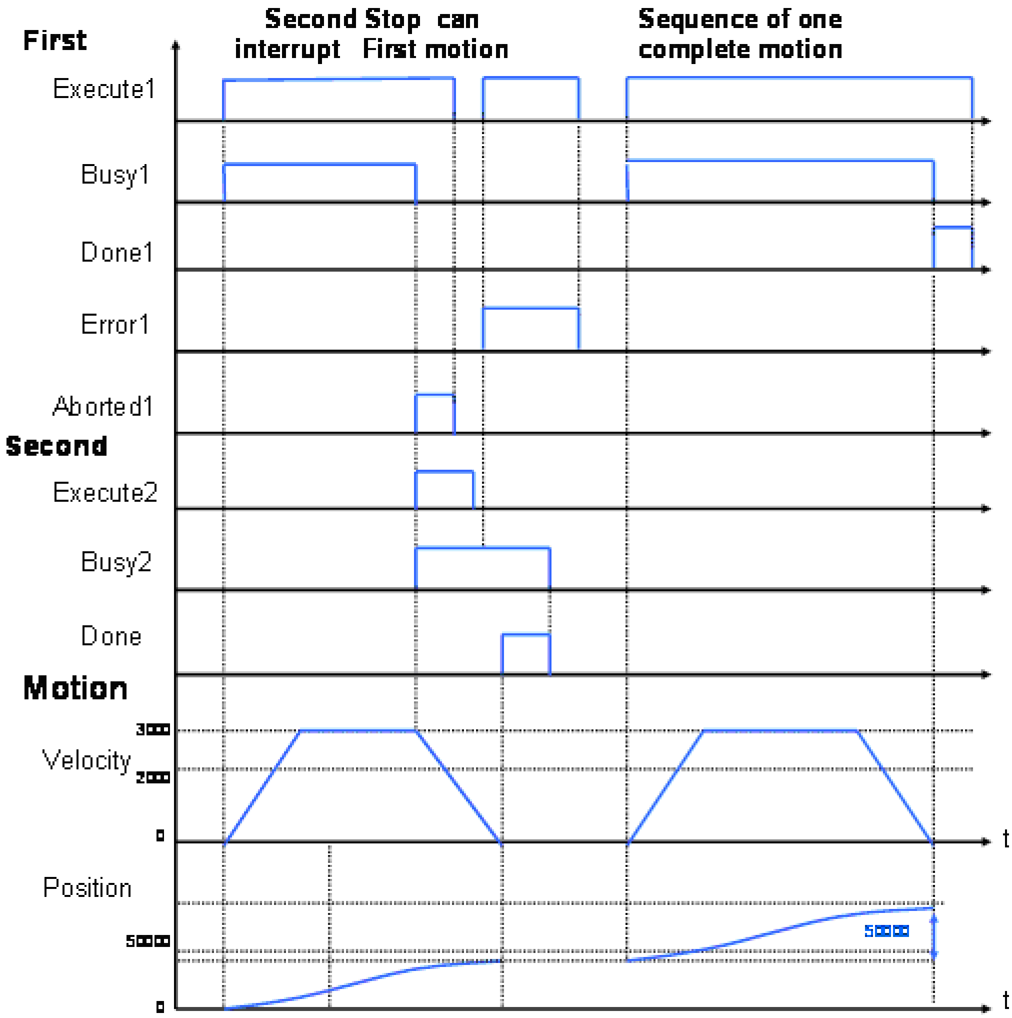
4. Example

The single-speed motion of an axis is started, and then the motion control function block T_AxisStop is used to stop the motion.

The motion control function block named First is used to start single-speed motion. It is set so that the first axis moves for 50,000 pulses at a speed of 10,000 per second. The motion control function block named Second is used to stop the motion of the first axis.




The motion control function block named First is started. Before Done 1 is set to True, Execute2 is used to start the motion control function block named Second.

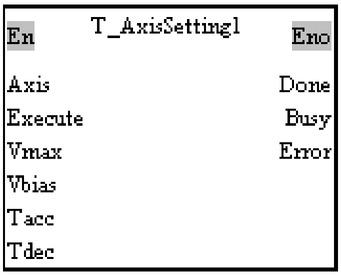



After the motion control function block named First is started, the first axis will move at a speed of 10,000 pulses per second. After the motion control function block named Second is started, Aborted1 will be set to True, Busy1 will be set to False, and the first axis will stop moving. When the motion control function block named Second is used to stop the motion of the first axis, no motion can be started. If any motion is started, an error will occur.

5. Modules which are supported

The motion control function block T_AxisStop supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

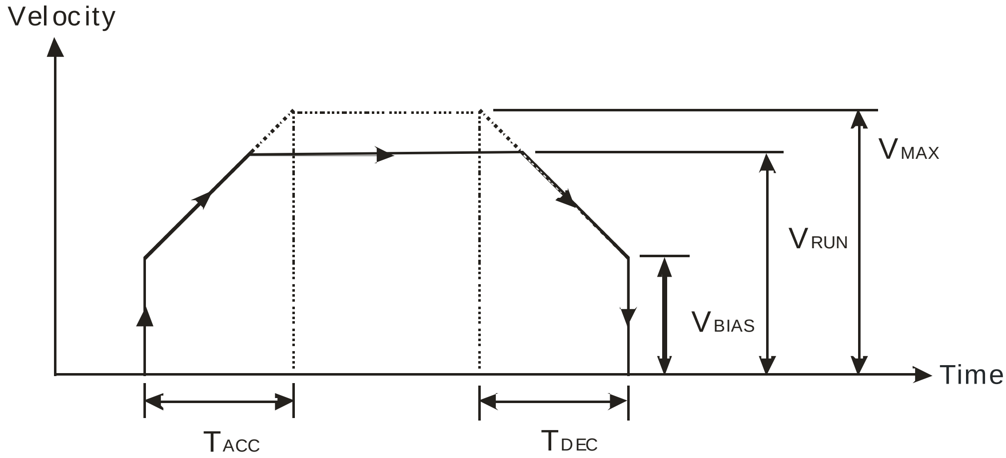
### 5.10.11 Parameter Setting I




1. Motion control function block

   The motion control function block T_AxisSetting1 is used to set motion parameters. The value of the Axis input pin indicates an axis number. Users can set the maximum speed of the axis specified, the start-up speed of the axis specified, the time it takes for the start-up speed of the axis specified to increase to its maximum speed, and the time it takes for the maximum speed of the axis specified to decrease to its start-up speed.

   The relation among the time it takes for the start-up speed of the axis specified to increase to its maximum speed, the time it takes for its maximum speed to decrease to its start-up speed, its start-up speed, its maximum speed, and its execution speed is shown below.




   $V_{RUN}$ is the execution speed of the axis specified. The axis specified moves according to the time it takes for its start-up speed to increase to its maximum speed, the time it takes for its maximum speed to decrease to its start-up speed, its start-up speed, and its maximum speed.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | Parameters are written when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Vmax | Maximum speed | DWORD | K1~K2,147,483,647 | The value of the Vmax input pin is valid when there is a transition in the Execute input pin's signal from low tot high. |
| Vbias | Start-up speed | DWORD | K0~K2,147,483,647 | The value of the Vbias input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Tacc | Acceleration time (Unit: ms) | WORD | K0~K32,767 | The value of the Tacc input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Tdec | Deceleration time (Unit: ms) | WORD | K0~K32,767 | The value of the Tdec input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when the writing of parameters is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is complete, the Done output pin will be set to False in the next cycle. |

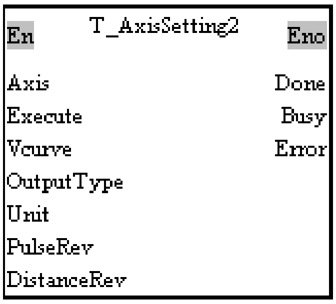
| Name | Function | Data type | State of output pin: Time when there is a transition in an output pin's signal from low to high | State of output pin: Time when there is a transition in an output pin's signal from high to low |
|---|---|---|---|---|
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_AxisSetting1 supports AH05PM-5A, AH10PM-5A, AH15PM-5A and AH20MC-5A.

## 5.10.12 Parameter Setting II




1. Motion control function block
   The motion control function block T_AxisSetting2 is used to set motion parameters. The value of the Axis input pin indicates an axis number. Users can set the velocity curve of the axis specified,

an output type, and a unit. The setting of a unit requires the number of pulses it takes for a motor to rotate once and the distance for which the axis specified moves when the motor rotates once.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | Parameters are written when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Vcurve | Velocity curve | BOOL | mcTrapezoid: False<br>mcSCurve: True | The value of the Vcurve input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| OutputType (AH20MC-5A is not supported.) | Output type | WORD | mcUD: 0<br>mcPD: 1<br>mcAB: 2<br>mc4AB: 3 | The value of the OutputType input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Unit | Unit | WORD | mcMotor: 0<br>mcMachine: 1<br>mcComp: 2 | The value of the Unit input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| PulseRev | Number of pulses it takes for a motor to rotate once | DWORD | K1~K2,147,483,647 | The value of the PulseRev input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| DistanceRev | Distance for which the axis specified moves when the motor used rotates once | DWORD | K1~K2,147,483,647 | The value of the DistanceRev input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

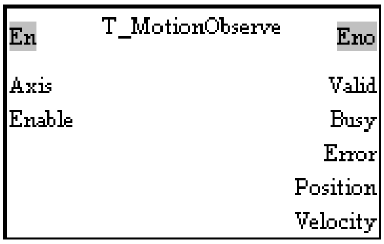
| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when the writing of parameters is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_AxisSetting2 supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

### 5.10.13 Reading the Present Position / Speed of an Axis




1. Motion control function block
   The motion control function block T_MotionObserve is used to read the present position/speed of an axis. The value of the Axis input pin indicates an axis number. After the motion control function block is started, users can read the present position of the axis specified through the Position output pin, and the speed of the axis specified through the Velocity output pin.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | Enabling the reading of the present position/speed of the axis specified | BOOL | True/False | - |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | ● There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low.<br>● There is a transition in the Valid output pin's signal from high to low when there is a transition in the Error input pin's signal from low to high. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

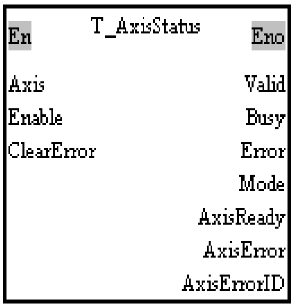
| Value of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Output range** | **Update** |
| Position | Present position (Pulse unit) | DWORD | K-2,147,483,648~ K2,147,483,647 | When the motion control function block is executed, the value of the Position output pin is updated repeatedly. |
| Velocity | Present speed (Pulse unit) | DWORD | K0~K2,147,483,647 | When the motion control function block is executed, the value of the Velocity output pin is updated repeatedly. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
The motion control function block T_MotionObserve supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.10.14 State of an Axis




1. Motion control function block
   The motion control function block is T_AxisStatus is used to read and clear the present erroneous state of an axis. The value of the Axis input pin indicates an axis number. Users can clear the present erroneous state of the axis specified by means of the ClearError input pin. The value of the AxisErrorID output pin indicates the present erroneous state of the axis specified.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | Enabling the displaying of the state of an axis | BOOL | True/False | - |
| ClearError | The erroneous state of the axis specified is cleared when there is a transition in the ClearError input pin's signal from low to high. | BOOL | True/False | The value of the ClearError input pin is valid when the motion control function block is executed. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | • There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low.<br>• There is a transition in the Valid output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| **Value of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Update** |
| Mode | Mode of motion | WORD | 16#0~16#32x (*1) | When the motion control function block is executed, the value of the Mode output pin is updated repeatedly. |
| AxisReady | Ready flag for the axis specified | BOOL | True/False | When the motion control function block is executed, the value of the AxisReady output pin is updated repeatedly. |
| AxisError | Axis error flag | BOOL | True/False | When the motion control function block is executed, the value of the AxisError output pin is updated repeatedly. |
| AxisErrorID | Error code | WORD | 16#0002~16#C4FF, 16#8001~16#8380 (*2) | When the motion control function block is executed, the value of the AxisErrorID output pin is updated repeatedly. |

*1: Value of the Mode output pin

| Value | Definition |
|---|---|
| 16#0 | Idle |
| 16#100 | Uniaxial motion is being stopped. |
| 16#101 | Absolute single-speed motion |
| 16#102 | Relative single-speed motion |
| 16#103 | Absolute two-speed motion |
| 16#104 | Relative two-speed motion |
| 16#105 | Inserting single-speed motion |
| 16#106 | Inserting two-speed motion |
| 16#107 | JOG motion |
| 16#108 | Manual pulse generator mode |
| 16#109 | Motion of returning home |
| 16#10A | Electronic gear motion |
| 16#10B | Electronic cam motion |
| 16#200 | G-code motion is being stopped. |
| 16#201 | Executing G-code motion |
| 16#300 | Multiaxial interpolation is being stopped. |
| 16#31x | Multiaxial absolute linear interpolation |
| 16#32x | Multiaxial relative linear interpolation |

*2: Value of the AxisErrorID output pin

| Value | Definition |
|---|---|
| 16#0002~16#C4FF | An error occurs in the AH500 series motion control module. |
| 16#8001~16#8380 | An error occurs in the ASD-A2-F series servo drive. |

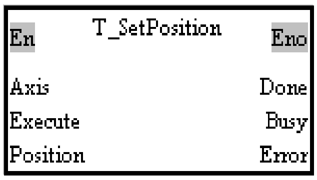
- Please refer to appendix A for more information about error codes.
- Error code in an ASD-A2-F series servo drive: The value of the AxisErrorID output pin is AL code+16#8000.

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
The motion control function block is T_AxisStatus supports AH05PM-5A, AH10PM-5A , AH15PM-5A, and AH20MC-5A.

## 5.10.15 Setting the Present Position of an Axis




1. Motion control function block
The motion control function block T_SetPosition is used to set the present position of an axis. The value of the Axis input pin indicates an axis number. Users can set the present position of the axis specified by means of the Position input pin.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | The present position of an axis is written when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Position | Present position of the axis specified | DWORD | K-2,147,483,648~ K2,147,483,647 | The value of the Position input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • The writing of a position is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

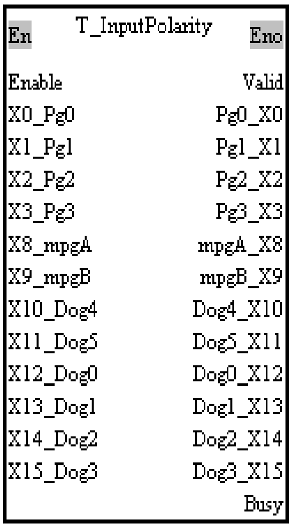
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_SetPosition supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

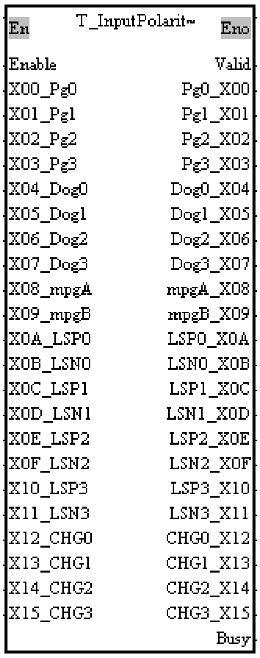
Note: To prevent errors from occurring, please avoid using the motion control function block to set the present position of the master axis involved in cam motion or gear motion.

## 5.10.16 Setting the Polarities of Input Terminals

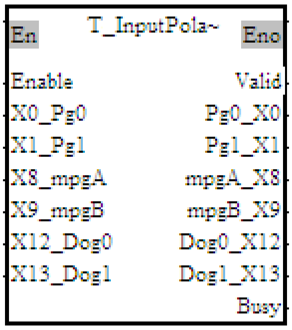
AH20MC-5A/AH10PM-5A




AH15PM-5A




AH05PM-5A




1. Motion control function block
   The motion control function block T_InputPolarity is used to set the polarities of the input terminals, and read the states of the input terminals in the AH500 series motion control module used. Users can set the polarities of the input terminals in the AH500 series motion control module used by means of input pins, and read the states of the input terminals in the AH500 series motion control module by means of output pins.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| Name | Function | Data type | Setting value | Time when a value is valid |
| Enable | Enabling the setting of the polarities of the input terminals, and the displaying of the states of the input terminals in the AH500 series motion control | BOOL | True/False | - |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| | module used | | | |
| X0_Pg0 | Polarity | BOOL | mcNO: False<br>mcNC: True | When the motion control function block is executed, the values of the input pins are updated repeatedly. |
| X1_Pg1 | Polarity | BOOL | | |
| X2_Pg2 | Polarity | BOOL | | |
| X3_Pg3 | Polarity | BOOL | | |
| X4_Dog0[*1] | Polarity | BOOL | | |
| X5_Dog1[*1] | Polarity | BOOL | | |
| X6_Dog2[*1] | Polarity | BOOL | | |
| X7_Dog3[*1] | Polarity | BOOL | | |
| X8_mpgA | Polarity | BOOL | | |
| X9_mpgB | Polarity | BOOL | | |
| X0A_LSP0[*1] | Polarity | BOOL | | |
| X0B_LSN0[*1] | Polarity | BOOL | | |
| X0C_LSP1[*1] | Polarity | BOOL | mcNO: False<br>mcNC: True | When the motion control function block is executed, the values of the input pins are updated repeatedly. |
| X0D_LSN1[*1] | Polarity | BOOL | | |
| X0E_LSP2[*1] | Polarity | BOOL | | |
| X0F_LSN2[*1] | Polarity | BOOL | | |
| X10_Dog4/<br>X10_LSP3[*1] | Polarity | BOOL | | |
| X11_Dog5/<br>X11_LSN3[*1] | Polarity | BOOL | | |
| X12_Dog0/<br>X12_CHG0[*1] | Polarity | BOOL | | |
| X13_Dog1/<br>X12_CHG1[*1] | Polarity | BOOL | | |
| X14_Dog2/<br>X12_CHG2[*1] | Polarity | BOOL | | |
| X15_Dog3/<br>X12_CHG3[*1] | Polarity | BOOL | | |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | • There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

<table>
<tr><th colspan="5">State of output pin</th></tr>
<tr><th>Name</th><th>Function</th><th>Data type</th><th>Time when there is a transition in an output pin’s signal from low to high</th><th>Time when there is a transition in an output pin’s signal from high to low</th></tr>
<tr><td>Pg0_X0</td><td>Polarity</td><td>BOOL</td><td rowspan="22">• When the values of input pins are set to True, and the input terminals are OFF, there are transitions in these output pins’ signals from low to high.<br>• When the values of input pins are set to False, and the input terminals are ON, there are transitions in these output pins’ signals from low to high.</td><td rowspan="22">• When the values of input pins are set to True, and the input terminals are ON, there are transitions in these output pins’ signals from high to low.<br>• When the values of input pins are set to False, and the input terminals are OFF, there are transitions in these output pins’ signals from high to low.<br>• There are transitions in these output pins’ signals from high to low when there is a transition in the Enable input pin’s signal from high to low.</td></tr>
<tr><td>Pg1_X1</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Pg2_X2</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Pg3_X3</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Dog0_X4[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Dog1_X5[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Dog2_X6[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Dog3_X7[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>mpgA_X8</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>mpgB_X9</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>LSP0_X0A[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>LSN0_X0B[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>LSP1_X0C[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>LSN1_X0D[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>LSP2_X0E[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>LSN2_X0F[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Dog4_X10/<br>LSP3_X10[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Dog5_X11/<br>LSN3_X11[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Dog0_X12/<br>CHG0_X12[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Dog1_X13/<br>CHG1_X13[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Dog2_X14/<br>CHG2_X14[*1]</td><td>Polarity</td><td>BOOL</td></tr>
<tr><td>Dog3_X15/<br>CHG3_X15[*1]</td><td>Polarity</td><td>BOOL</td></tr>
</table>

*1: It indicates a terminal of AH15PM-5A.

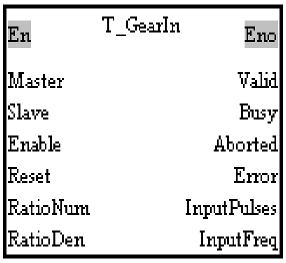
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported

The motion control function block T_InputPolarity supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

### 5.10.17 Electronic Gear Motion




1. Motion control function block
   The motion control function block T_GearIn is used to start electronic gear motion. The value of the Master input pin indicates a master axis, and the value of the Slave input pin indicates a slave axis. The motion of the slave axis specified follows the motion of the master axis specified. The value of the RatioNum input pin is the numerator of an electronic gear ratio. The value of the RatioDen input pin is the denominator of an electronic gear ration. The Reset input pin is used to clear the number of input pulses.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Master | Master axis number | WORD | 0~16, 200, 204, 208, 212, 216, 220 (*1) | The value of the Master input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Slave | Slave axis number | WORD | 1~16 | The value of the Slave input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | Enabling electronic gear motion | BOOL | True/False | - |
| Reset | Resetting the InputPulses output pin | BOOL | True/False | The value of the Reset input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| RatioNum | Numerator of an electronic gear ratio | DWORD | K-32,767~K32,767 | When the motion control function block is executed, the value of the RatioNum input pin is updated repeatedly. |
| RatioDen | Denominator of an electronic gear ratio | DWORD | K1~K32,767 | When the motion control function block is executed, the value of the RatioDen input pin is updated repeatedly. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | • There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Valid output pin's signal from high to low when motion stops.<br>• There is a transition in the Valid output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Valid output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Busy | The motion control function block is being executed. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low.<br>• If the Enable input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The axis specified is in motion before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

| Value of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Output range** | **Update** |
| InputPulses | Number of input pulses | DWORD | K-2,147,483,648~ K2,147,483,647 | When the motion control function block is executed, the value of the InputPulses output pin is updated repeatedly. |
| InputFreq | Frequency of input pulses | DWORD | K0~K2,147,483,647 | When the motion control function block is executed, the value of the InputFreq output pin is updated repeatedly. |

*1: Value of the Master input pin

| Value | Definition |
|---|---|
| 0 | Manual pulse generator |
| 1~16 | Motion axis 1~motion axis 16 |
| 200 | C200 |
| 204 | C204 |
| 208 | C208 |
| 212 | C212 |
| 216 | C216 |
| 220 | C220 |

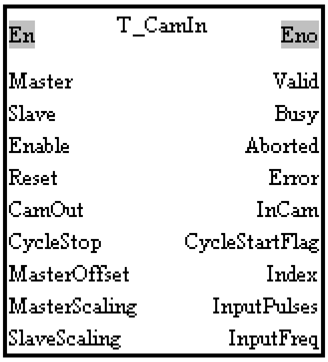
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported

The motion control function block T_GearIn supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.10.18 Electronic Cam Motion




1. Motion control function block

The motion control function block T_CamIn is used to start electronic cam motion. The value of the Master input pin indicates a master axis, and the value of the Slave input pin indicates a slave axis. The motion of the slave axis specified follows the motion of the master axis specified. The value of the MasterOffset input pin indicates the starting angle of the master axis specified. The Reset input pin is used to clear the number of input pulses. If the CamOut input pin is set to True, the slave axis specified will not mesh with the master axis specified. If the CycleStop input pin is set to True when the Enable input pin is reset, cam motion will not stop until a cycle is complete. Please refer to section 8.2.2 for more information.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Master | Master axis number | WORD | 0~16, 200, 204, 208, 212, 216, 220 (*1) | The value of the Master input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Slave | Slave axis number | WORD | 1~16 | The value of the Slave input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | Enabling electronic cam motion | BOOL | True/False | - |
| Reset | Resetting the InputPulses output pin | BOOL | True/False | The value of the Reset input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| CamOut | Not meshing with the master axis specified | BOOL | True/False | The value of the CamOut input pin is used when the motion control function block is executed. |
| CycleStop | Stopping a whole cycle | BOOL | True/False | The value of the CycleStop input pin is valid when there is a transition in the Enable input pin's signal from high to low. |
| MasterOffset | Starting angle of the axis specified (Unit: Pulse) | DWORD | K0~ K2,147,483,647 | The value of the MasterOffset input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| MasterScaling | Ratio which is used to reduce/enlarge the number of pulses sent by the master axis specified | FLOAT | 0.~650.00 (two decimal places) | The value of the MasterScaling input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| SlaveScaling | Ratio which is used to reduce/enlarge the number of pulses sent by the slave axis specified | FLOAT | 0.~650.00 (two decimal places) | The value of the SlaveScaling input pin is valid when there is a transition in the Enable input pin's signal from low to high. |

5

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | • There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Valid output pin's signal from high to low when motion stops.<br>• There is a transition in the Valid output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Valid output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Busy | The motion control function block is being executed. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low.<br>• If the Enable input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The cam chart created is incorrect.<br>• The axis specified is in motion before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| CycleStartFlag | Starting a cam cycle (The value of the CycleStartFlag output pin remains unchanged for one scan cycle.) | BOOL | • A cam cycle begins | • There is a transition in the CycleStartFlag output pin's signal from high to low in the scan cycle following a cam cycle. |
| InCam | The slave axis specified meshes with the master axis specified. | BOOL | • There is a transition in the InCam output pin's signal from low to high when there is a transition in the CamOut input pin's signal from low to high. | • There is a transition in the InCam output pin's signal from high to low when there is a transition in the CamOut input pin's signal from high to low. |
| **Value of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Update** |
| Index | Index of a point | DWORD | K1~K2047 | When the motion control function block is executed, the value of the Index output pin is updated repeatedly. |
| InputPulses | Number of input pulses | DWORD | K-2,147,483,648~ K2,147,483,647 | When the motion control function block is executed, the value of the InputPulses output pin is updated repeatedly. |
| InputFreq | Frequency of input pulses | DWORD | K0~K2,147,483,647 | When the motion control function block is executed, the value of the InputFreq output pin is updated repeatedly. |

*1: Value of the Master input pin

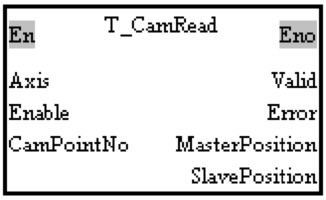
| Value | Definition |
|---|---|
| 0 | Manual pulse generator |
| 1~16 | Motion axis 1~motion axis 16 |
| 200 | C200 |
| 204 | C204 |
| 208 | C208 |
| 212 | C212 |
| 216 | C216 |
| 220 | C220 |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
The motion control function block T_CamIn supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

### 5.10.19 Reading a Cam Point




1. Motion control function block
The motion control function block T_CamRead is used to read a particular point in a cam chart. The value of the Axis input pin indicates an axis number. The value of the CamPoiontNo input pin indicates a cam point number. The value of the MasterPosition output pin indicates the position of the master axis specified, and the value of the SlavePosition output pin indicates the position of the slave axis specified.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | 1~16 | The value of the Axis input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | The reading of a cam point is enabled when there is a transition in the Enable input pin's signal from low to high. | BOOL | True/False | - |
| CamPointNo | Cam point number | DWORD | K0~K2046 | When the motion control function block is executed, the value of the CamPointNo input pin is updated repeatedly. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | ● There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low.<br>● There is a transition in the Valid output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high. |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| **Value of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| MasterPosition | Position of the master axis specified | DWORD | K-2,147,483,647~ K2,147,483,647 | When the motion control function block is executed, the value of the MasterPosition output pin is updated repeatedly. |
| SlavePosition | Position of the slave axis specified | DWORD | K-2,147,483,647~ K2,147,483,647 | When the motion control function block is executed, the value of the SlavePosition output pin is updated repeatedly. |

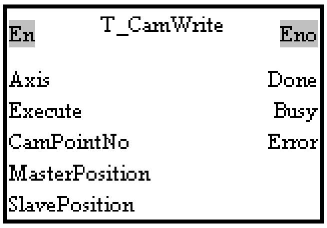
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported

   The motion control function block T_CamRea supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.10.20 Writing a Cam Point




1. Motion control function block
   The motion control function block T_CamWrite is used to modify a particular point in a cam chart. The value of the Axis input pin indicates an axis number. The value of the CamPoiontNo input pin indicates a cam point number. The value of the MasterPosition indicates the position of the master axis specified, and the value of the SlavePosition indicates the position of the slave axis specified.
   Note: If users want to modify all the points in a cam chart, the pair of coordinates (0, 0) will need to be written after the last point is modified.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | 1~16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | The writing of a cam point is enabled when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| CamPointNo | Cam point number | DWORD | K0~2046 | The value of the CamPointNo input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| MasterPosition | Position of the master axis specified | DWORD | K-2,147,483,647~ K2,147,483,647 | The value of the MasterPosition input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| SlavePosition | Position of the slave axis specified | DWORD | K-2,147,483,647~ K2,147,483,647 | The value of the SlavePosition input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

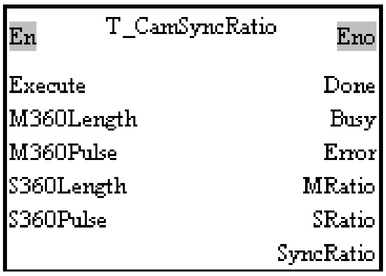
| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when the writing of a cam point is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The slave axis specified meshes with the master axis specified before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
The motion control function block T_CamWrite supports AH05PM-5A, AH15PM-5A, AH10PM-5A, and AH20MC-5A.

## 5.10.21 Calculating a Synchronization Ratio




1. Motion control function block
   The motion control function block T_CamSyncRatio is used to calculate a synchronization ratio. A synchronization ration is calculated by means of the M360Length input pin, the M360Pulse input pin, the S360Length input pin, and the S360Pulse input pin. (The value of the M360Length input pin indicates physical quantity, and the value of the M360Pulse input pin indicates the number of pulses. The value of the S360Length input pin indicates physical quantity, and the value of the S360Pulse input pin indicates the number of pulses.)
2. Input pins/Output pins

| **Input pin** | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | Enabling the calculation of a synchronization ratio | BOOL | True/False | - |
| M360Length | Distance for which the master axis specified moves in a cycle | DWORD | K1~K2,147,483,647 | The value of the M360Length input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| M360Pulse | Number of pulses for which the master axis specified moves in a cycle | DWORD | K1~K2,147,483,647 | The value of the M360Pulse input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| S360Length | Distance for which the slave axis specified moves in a cycle | DWORD | K1~K2,147,483,647 | The value of the S360Length input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| S360Pulse | Number of pulses for which the slave axis specified moves in a cycle | DWORD | K1~K2,147,483,647 | The value of the S360Pulse input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal when the calculation of a synchronization ratio is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The slave axis specified meshes with the master axis specified before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |
| **Value of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| MRatio | Ratio of the distance for which a master axis moves to the number of pulses for which the master axis moves | DWORD | K-2,147,483,647~ K2,147,483,647 | The value of the MRatio output pin is valid when there is a transition in the Done output pin's signal from low to high. |

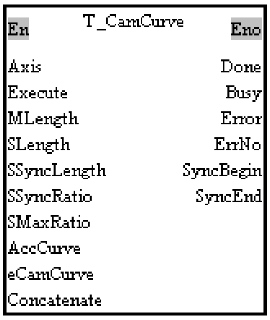
| Value of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| SRatio | Ratio of the distance for which a slave axis moves to the number of pulses for which the slave axis moves | DWORD | K-2,147,483,647~ K2,147,483,647 | The value of the SRatio output pin is valid when there is a transition in the Done output pin's signal from low to high. |
| SyncRatio | Synchronization ratio | DWORD | K-2,147,483,647~ K2,147,483,647 | The value of the SyncRatio output pin is valid when there is a transition in the Done output pin's signal from low to high. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
The motion control function block T_CamSyncRatio supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.10.22 Creating a Cam Curve




1. Motion control function block
The motion control function block T_CamCurve is used to create a cam curve. The value of the Axis input pin indicates an axis number. The value of the MLength input pin, the value of the SLength input pin, the value of the SSyncRatio input pin, and the value of the SMaxRatio input pin indicate the physical quantity needed to generate a cam curve. The value of the AccCurve input pin and the value of the eCamCurve determine a cam curve type. Please refer to section 8.4.2 for more information.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | 1~16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | The creation of a cam curve is enabled when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| MLength | Distance for which the master axis specified moves | DWORD | K1~K2,147,483,647 | The value of the MLength input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| SLength | Distance for which the slave axis specified moves | DWORD | K1~K2,147,483,647 | The value of the SLength input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| SSyncLength | Distance for which the slave axis specified is synchronized with the master axis specified | DWORD | K1~K2,147,483,647 | The value of the SSyncLength input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| SSyncRatio | Synchronization ratio of the speed of the slave axis specified to the speed of the master axis specified | FLOAT | $1.1755\times10^{-38}$~ $3.4028\times10^{+38}$ | The value of the SSyncRatio input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| SMaxRatio | Maximum ratio of the speed of the slave axis to the speed of the master axis specified | FLOAT | $1.1755\times10^{-38}$~ $3.4028\times10^{+38}$ | - |
| AccCurve | Acceleration curve | WORD | 0~3 (*1) | The value of the SSyncRatio input pin is valid when there is a transition in the AccCurve input pin's signal from low to high. |
| eCamCurve | Cam curve | WORD | 0~5 (*2) | |
| Concatenate | Concatenation | BOOL | True/False | The value of the Concatenate input pin is valid when there is a transition in the AccCurve input pin's signal from low to high. |

5

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | ● There is a transition in the Done output pin's signal when the creation of a cam curve is complete. | ● There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when the execution of the motion control function block is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |
| **Value of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| ErrNo | Error code | WORD | 0~2 | When the motion control function block is executed, the value of the ErrNo output pin is updated repeatedly. |
| SyncBegin | Starting point of synchronization | DWORD | K0~K2,147,483,647 | When the motion control function block is executed, the value of the SyncBegin output pin is updated repeatedly. |

| Value of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| SyncEnd | Terminal point of synchronization | DWORD | K0~K2,147,483,647 | When the motion control function block is executed, the value of the SyncEnd output pin is updated repeatedly. |

*1: Value of the AccCurve input pin

| Value | Definition |
|---|---|
| 0 | Uniform curve |
| 1 | Uniform acceleration curve |
| 2 | SingleHypot curve |
| 3 | Cycloid |

*2: Value of the eCamCurve input pin

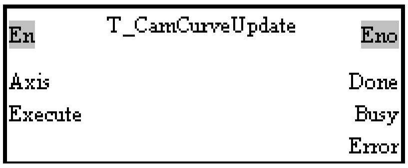
| Value | Definition |
|---|---|
| 0 | leftCAM |
| 1 | midCAMall |
| 2 | midCAMbegin |
| 3 | midCAMend |
| 5 | rightCAM |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
The motion control function block T_CamCurve supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.10.23 Updating a Cam Curve




1. Motion control function block
The motion control function block T_CamCurveUpdate is used to update a cam chart so that the cam curve in the next can cycle is the cam curve created by means of the motion control function block T_CamCurve. The value of the Axis input pin indicates an axis number.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | 1~16 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | When there is a transition in the Execute input pin's signal from low to high, the update of a cam curve is | BOOL | True/False | - |

| | | | | |
|---|---|---|---|---|
| | enabled. | | | |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal when the update of a cam curve is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

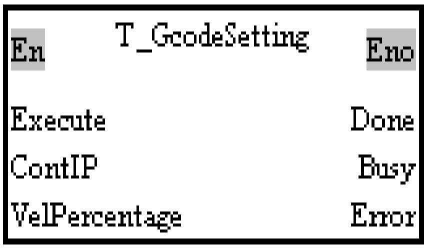
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported

The motion control function block T_CamCurveUpdate supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

# 5.11 Multiaxial Motion Control Function Blocks

## 5.11.1 Setting the Parameters of G-code Motion




1. Motion control function block
   The motion control function block T_GcodeSetting is used to set the parameters of G-code motion. The value of the ContIP input pin indicates the minimum speed to which the speed of continuous interpolation decreases. If the speed of G-code motion is less than the speed indicated by the value of the ContIP input pin, the G-code motion will move at the speed indicated by the value of the ContIP input pin. The value of the VelPercentage input pin indicates the percentage for the values of the speed parameters of G-codes.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | The parameters of G-code motion are set when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| ContIP | Minimum speed to which the speed of continuous interpolation decreases | DWORD | K0~K500000 | The value of the ContIP input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| VelPercentage | Percentage for the values of the speed parameters of G-codes | WORD | K0~K65,535 | The value of the VelPercentage input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal when the setting of the parameters of G-code motion is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

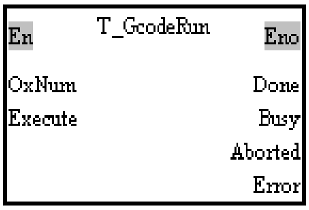
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported

The motion control function block T_GcodeSetting supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.11.2 Executing G-code Motion




1. Motion control function block
   The motion control function block T_GcodeRun is used to set and execute an Ox motion subroutine. The value of the OxNum indicates an Ox motion subroutine number.
2. Input pins/Output pins

| **Input pin** | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| OxNum | Ox motion subroutine number | WORD | OX0~OX99: 0~99<br>SD card: 100~199 | The value of the OxNum input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | An Ox motion subroutine is executed when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the function block is complete. | BOOL | • There is a transition in the Done output pin's signal when the execution of an Ox motion subroutine is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect.<br>● The slave axis specified meshes with the master axis specified before the motion control function block is executed. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | ● The execution of the motion control function block is interrupted by a command. | ● There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

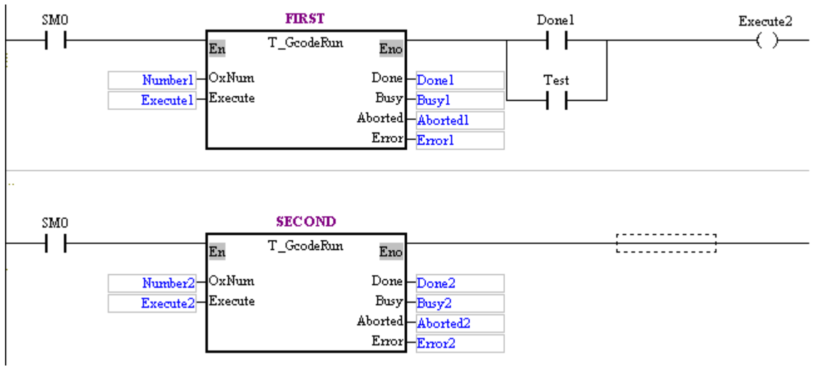
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Example

Purposes:

- After the first G-code motion is complete, the second G-code motion will be executed.
- The second G-code motion is executed before the execution of the first G-code motion is complete.

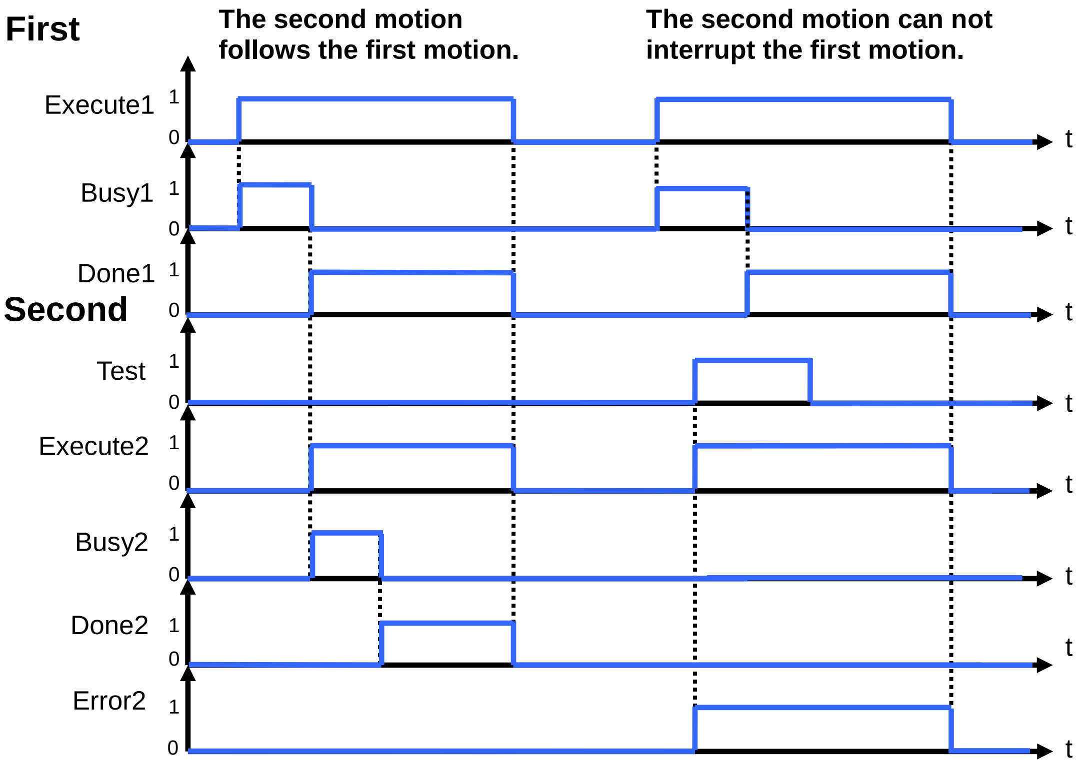
The motion control function block named FIRST and the motion control function block named SECOND are set so that two different Ox motion subroutines are executed.




- After the first G-code motion is complete, the second G-code motion will be executed.
  Steps:
  (a) Set Execute1 to True.
  (b) Wait for a transition in Done2's signal from low to high or a transition in Error2's signal from low to high.
- The second G-code motion is executed before the execution of the first G-code motion is complete.
  Steps:
  (a) Set Execute1 to True.
  (b) Set Test to ON when Busy1 is set to True.
  (c) Wait for a transition in Done2's signal from low to high or a transition in Error2's signal from low to high.

Timing diagram:

**Number1 = Number2**




5. Modules which are supported
   The motion control function block T_GcodeRun supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

### 5.11.3 Stopping G-code Motion


1. Motion control function block
   The motion control function block T_GcodeStop is used to stop the execution of an Ox motion subroutine.
2. Input pins/Output pins

| **Input pin** | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | G-code motion is stopped when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the function block is complete. | BOOL | ● There is a transition in the Done output pin's signal from low to high when the stopping of G-code motion is complete. | ● There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

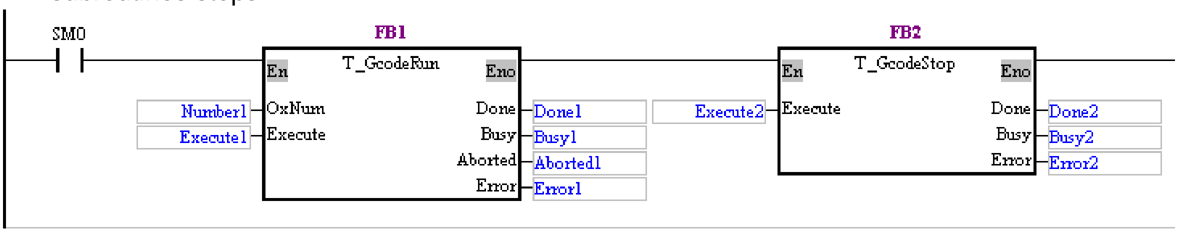
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Example

Purpose:

● The execution of an Ox motion subroutine stops before it is complete.

The motion control function block named FB1 is set so that an Ox motion subroutine is executed. The motion control function block named FB2 is set so that the execution of the Ox motion subroutines stops.




Steps:

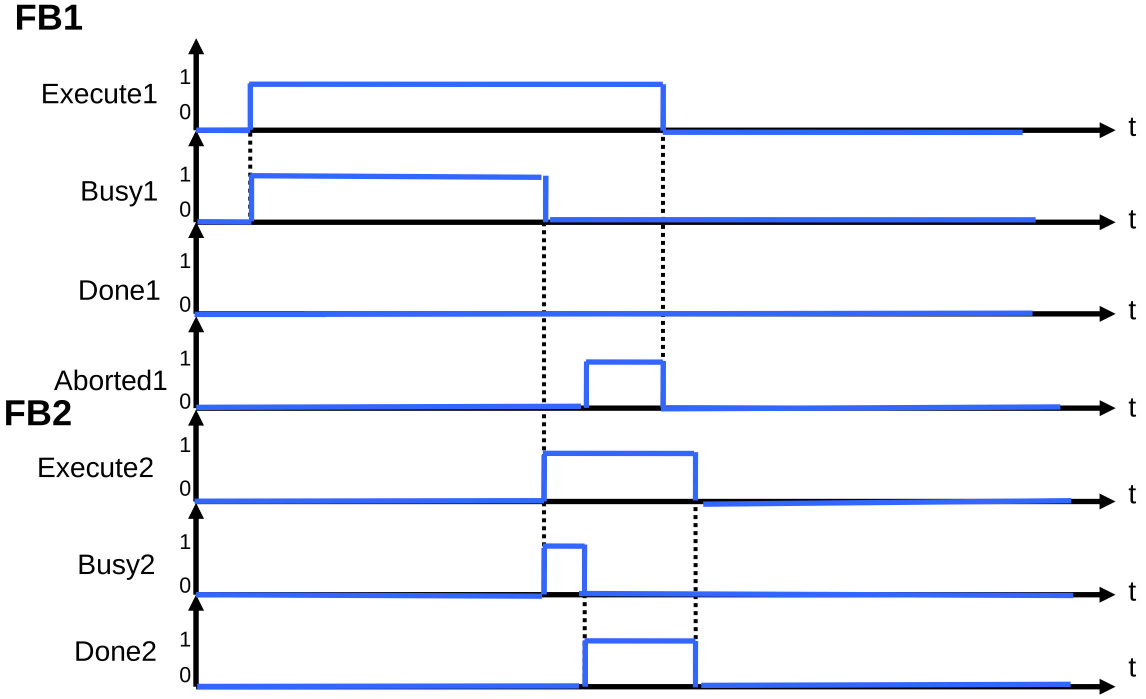
(a) Set Execute1 to True.

(b) Execute the G-codes in the Ox motion subroutine specified.

(c) Set Execute2 to True before the execution of the G-codes in the Ox motion subroutine specified is complete.

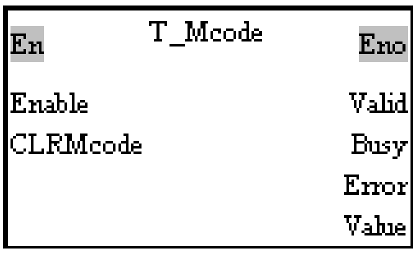
(d) Stop the execution of the Ox motion subroutine specified, and set Aborted1 to True.

Timing diagram:




5. Modules which are supported
   The motion control function block T_GcodeStop supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.11.4 Reading an M-code




1. Motion control function block
   The motion control function block T_Mcode is used to read an M-code, and clear the M-code specified. The CLRMcode input pin is used to clear the M-code specified.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Enable | Enabling the reading of an M-code | WORD | True/False | - |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| CLRMcode | An M-code is cleared when there is a transition in the CLRMcode input pin's signal from low to high, and the Enable input pin is set to True. | BOOL | True/False | The value of the CLRMcode input pin is valid when the motion control function block is executed. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | ● There is a transition in the Valid output pin's signal from low to high when an M-code is executed. | ● There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from low to high.<br>● There is a transition in the CLRMcode input pin's signal from low to high when the Valid output pin is set to True. There is a transition in the Valid output pin's signal from high to low when there is a transition in the CLRMcode input pin's signal from low to high. |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

| Value of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| Value | When the Valid output pin is set to True, the value of the Value of output pin indicates the M-code which is executed. | WORD | K0~4096 | When the Valid output pin is set to True, the value of the Value of output pin is updated repeatedly. |

3. Troubleshooting

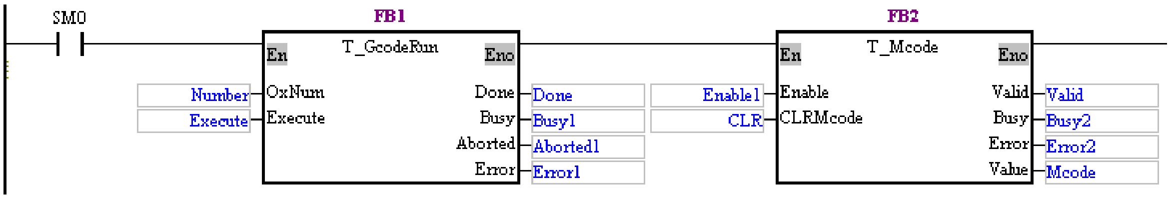
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Example

Purpose:

- When an Ox motion subroutine is executed, the motion control function block T_Mcode is used to check the status of an M-code. If an M-code is executed, the motion control function block T_Mcode will be used to clear the M-code.

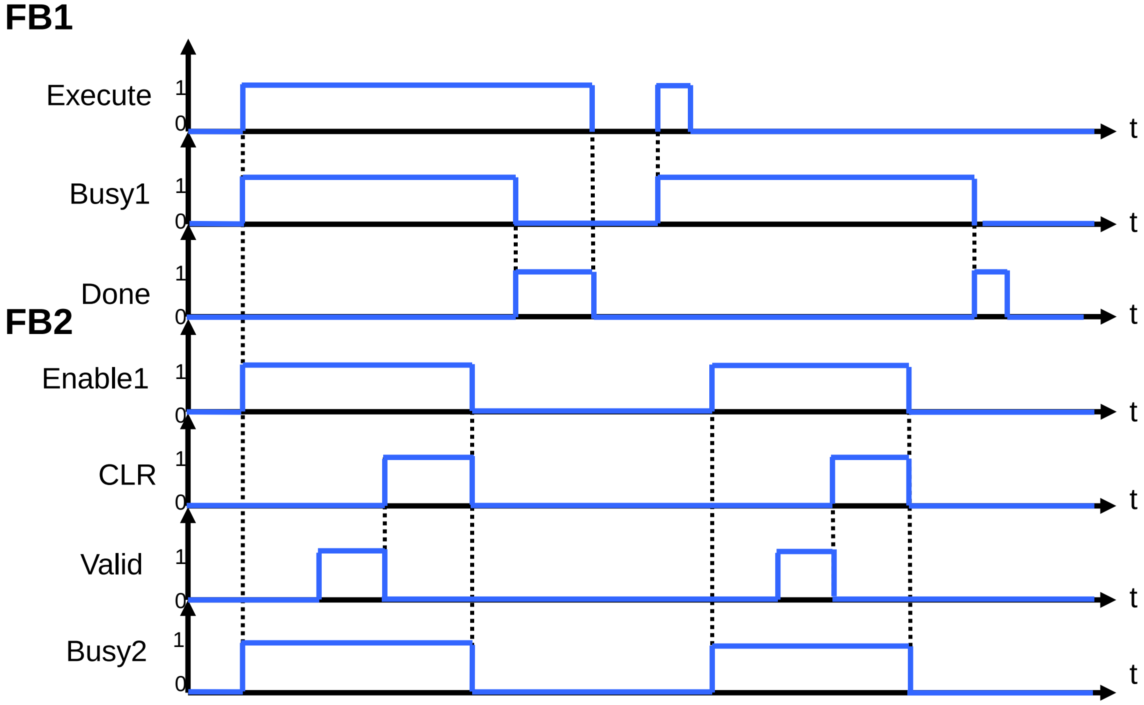
The motion control function block named FB1 is set so that an Ox motion subroutine is executed.
The motion control function block named FB2 is set so that the status of an M-code is checked.




Steps:

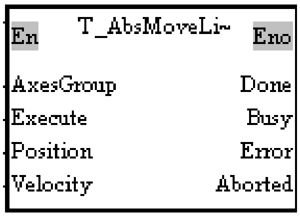
(a) Set Execute to True.
(b) Execute the G-codes in the Ox motion subroutine specified.
(c) Set Execute1 to True before the execution of the G-codes in the Ox motion subroutine specified is complete.
(d) Check the status of the M-code which is being executed.
(e) When an M-code is executed, Valid is set to True.
(f) CLR is used to clear the M-code which is executed.

Timing diagram:




5. Modules which are supported
   The motion control function block T_Mcode supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

### 5.11.5 Multiaxial Absolute Linear Interpolation




1. Motion control function block
   The motion control function block T_AbsMoveLinear is used to start multiaxial absolute linear interpolation. Users can set the axes which execute interpolation by means of the AxesGroup input pin, set the target positions of the axes specified by means of the Position input pin, and set the speed of the axes specified by means of the Velocity input pin.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | Motion is started when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| AxesGroup | Axes which execute interpolation | WORD[6] | [_,_,_,_,_,_]<br>0: Not setting axes<br>n: Adding the n$^{th}$ axis (n is in the range of 1 to 16.)<br>(The first cell must be set.) | The value of the AxesGroup input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Position | Target positions | DWORD[6] | [_,_,_,_,_,_]<br>K-2,147,483,648~ K2,147,483,647 | The value of the Position input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity | Speed of interpolation | DWORD | K1~K2,147,483,647 | The value of the Velocity input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the function block is complete. | BOOL | ● There is a transition in the Done output pin's signal from low to high when multiaxial absolute linear interpolation is complete. | ● There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |

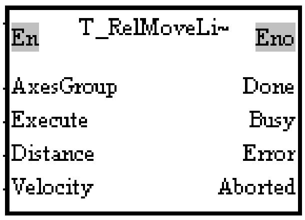
| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
The motion control function block T_AbsMoveLinear supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.11.6 Multiaxial Relative Linear Interpolation




1. Motion control function block
The motion control function block T_RelMoveLinear is used to start multiaxial relative linear interpolation. Users can set the axes which execute interpolation by means of the AxesGroup input pin, set the distances for which the axes specified move by means of the Distance input pin, and set the speed of the axes specified by means of the Velocity input pin.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| AxesGroup | Axes which execute interpolation | WORD[6] | [_,_,_,_,_]<br>0: Not setting axes<br>n: Adding the nth axis (n is in the range of 1 to 16.)<br>(The first cell must be set.) | The value of the AxesGroup input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | Motion is started when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Distance | Distances for which the axes specified move | DWORD[6] | [_,_,_,_,_,_]<br>K-2,147,483,648~ K2,147,483,647 | The value of the Distance input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Velocity | Speed of interpolation | DWORD | K1~K2,147,483,647 | The value of the Velocity input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when multiaxial relative linear interpolation is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |

5

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_RelMoveLinear supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.11.7 Stopping Multiaxial Linear Interpolation


1. Motion control function block
   The motion control function block T_GroupStop is used to stop multiaxial linear interpolation. Users can set the axes which execute interpolation by means of the AxesGroup input pin.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | Linear interpolation is stopped when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| AxesGroup | Axes which execute interpolation | WORD[6] | [_,_,_,_,_]<br>0: Not setting axes<br>n: Adding the $n^{th}$ axis (n is in the range of 1 to 16.)<br>(The first cell must be set.) | The value of the AxesGroup input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when the stopping of multiaxial linear interpolation is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

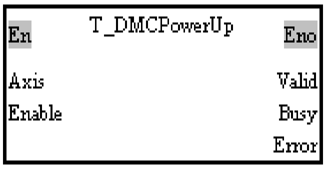
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported

   The motion control function block T_GroupStop supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

# 5.12 Network Function Blocks

## 5.12.1 Starting / Stopping a Servo Drive




1. Motion control function block
   The motion control function block T_DMCPowerUp is used to start or stop the servo drive specified on a DMCNET. The value of the Axis input pin indicates an axis number.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | 1~12 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Enable | The servo drive specified on a DMCNET is started when there is a transition in the Enable input pin's signal from low to high. The servo drive specified on a DMCNET is stopped when there is a transition in the Enable input pin's signal from high to low. | BOOL | True/False | - |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | ● There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Valid output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

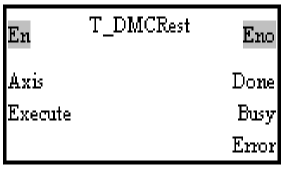
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported

   The motion control function block T_DMCPowerUp supports AH20MC-5A.

### 5.12.2 Resetting a Servo Drive




1. Motion control function block

   The motion control function block T_DMCRest is used when a network is abnormal. After a network is reset by the motion control function block T_DMCRest, users will have to use the motion control function block T_DMCControlInit to connect the motion control module and the servo drive which are used to the network.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | 1~12 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

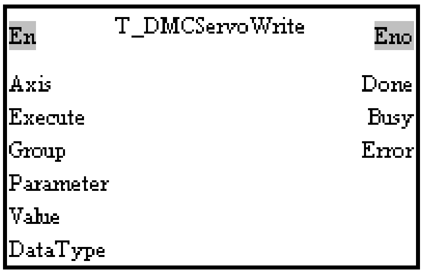
| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | A network is reset when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when the resetting of a network is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_DMCRest supports AH20MC-5A.

### 5.12.3 Writing the Value of a Parameter into a Servo Drive




1. Motion control function block
   The motion control function block T_DMCServoWrite is used to write the value of a parameter into the servo drive specified on a DMCNET. The value of the Axis input pin indicates an axis number, the value of the Group input pin indicates a group number, the value of the Parameter input pin indicates a parameter number, the value of the DataType input pin indicates a data type, and the value of the Value input pin indicates the value written into the servo drive specified.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | 1~12 | The value of the Axis input pin is valid when there is a transition in the Execute input pin’s signal from low to high. |
| Execute | The value of a parameter is written into a servo drive when there is a transition in the Execute input pin’s signal from low to high. | BOOL | True/False | - |
| Group | Group number (Please refer to ASDA-A2 Series User Manual for more details.) | WORD | 0~9 | The value of the Group input pin is valid when there is a transition in the Execute input pin’s signal from low to high. |
| Parameter | Parameter number (Please refer to ASDA-A2 Series User Manual for more details.) | WORD | 0~99 | The value of the Parameter input pin is valid when there is a transition in the Execute input pin’s signal from low to high. |
| Value | Value of a parameter (Please refer to ASDA-A2 Series User Manual for more details.) | DWORD | K-2,147,483,647~ K2,147,483,647 | The value of the Value input pin is valid when there is a transition in the Execute input pin’s signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| DataType | Data type | BOOL | mc16bits: False<br>mc32bits: True | The value of the DataType input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the function block is complete. | BOOL | ● There is a transition in the Done input pin's signal from low to high when the writing of the value of a parameter into the servo drive specified is complete. | ● There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

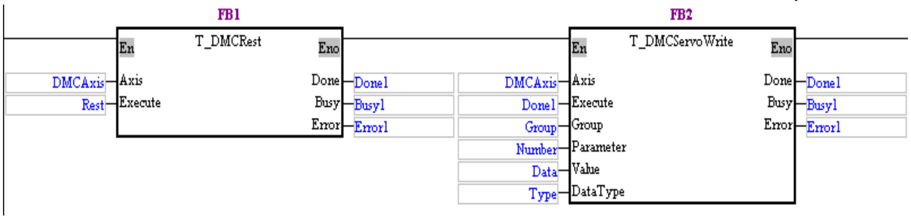
4. Example

Purpose:

- Users can reset the servo drive specified on a DMCNET by means of the motion control function block T_DMCRest, and then write the value of a parameter into the servo drive by means of the motion control function block T_DMCServoWrite.

The motion control function block named FB1 is set so that the servo drive specified is reset. The value of the Group input pin in the motion control function block named FB2 indicates a group number, the value of the Parameter input pin in the motion control function block named

FB2 indicates a parameter number, and the value of the Value input pin in the motion control function block named FB2 indicates the value written into the servo drive specified.



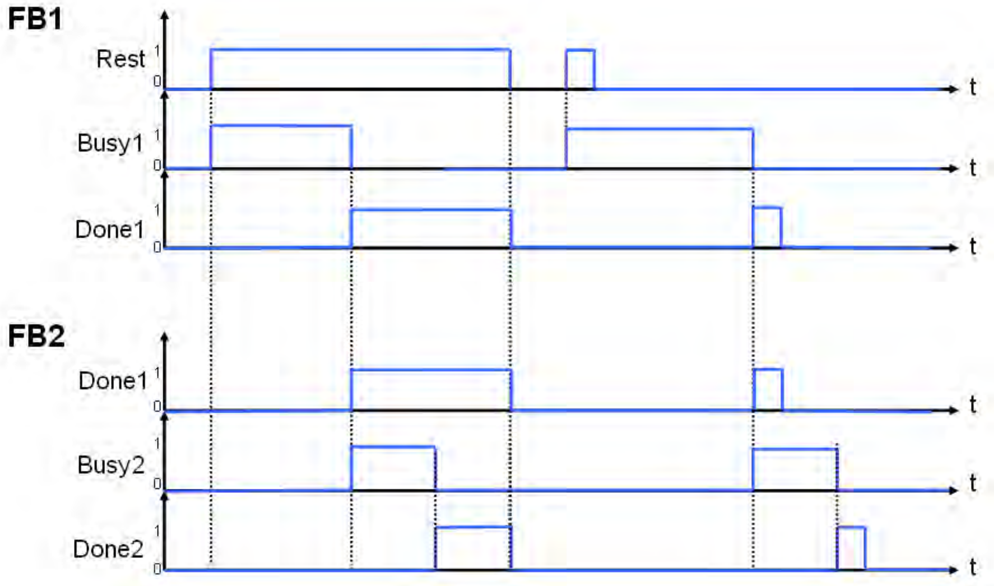

Steps:

(a) Reset the servo drive specified by means of the motion control function block named FB1.

(b) After the servo drive specified is reset, the motion control function block named FB2 will be executed automatically.

(c) After the execution of the motion control function block name FB2 is complete, Done1 will be set to True.

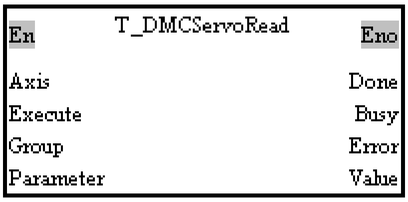
Timing diagram:




5. Modules which are supported

   The motion control function block T_DMCServoWrite supports AH20MC-5A.

## 5.12.4 Reading the Value of a Parameter from a Servo Drive




1. Motion control function block

   The motion control function block T_DMCServoRead is used to read the value of a parameter from the servo drive specified on a DMCNET. The value of the Axis input pin indicates an axis number, the value of the Group input pin indicates a group number, the value of the Parameter input pin indicates a parameter number, and the value of the Value input pin indicates the value read from the servo drive specified.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | 1~12 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | The value of a parameter is read from a servo drive when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Group | Group number | WORD | 0~9 | The value of the Group input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Parameter | Parameter number | WORD | 0~99 | The value of the Parameter input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the function block is complete. | BOOL | ● There is a transition in the Done input pin's signal from low to high when the reading of the value of a parameter from the servo drive specified is complete. | ● There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |
| **Value of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| Value | Value of a parameter | DWORD | K-2,147,483,647~ K2,147,483,647 | The value of the Value of output pin is valid when there is a transition in the Done output pin's signal from low to high. |

3. Troubleshooting

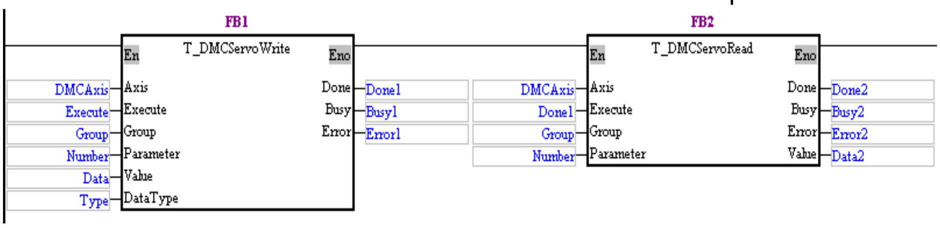
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Example

Purpose:

- Users write the value of a parameter into the servo drive specified by means of the motion control function block T_DMCServoWrite, and then read the value written into the servo drive by means of the motion control function block T_DMCServoRead.

The motion control function block named FB1 is set so that the value of a parameter is written into the servo drive specified. The motion control function block named FB2 is set so that the value written into the servo drive is read. The value of the Group input pin in the motion control function block named FB1/FB2 indicates a group number, and the value of the Parameter input pin in the motion control function block named FB1/FB2 indicates a parameter number.



Steps:

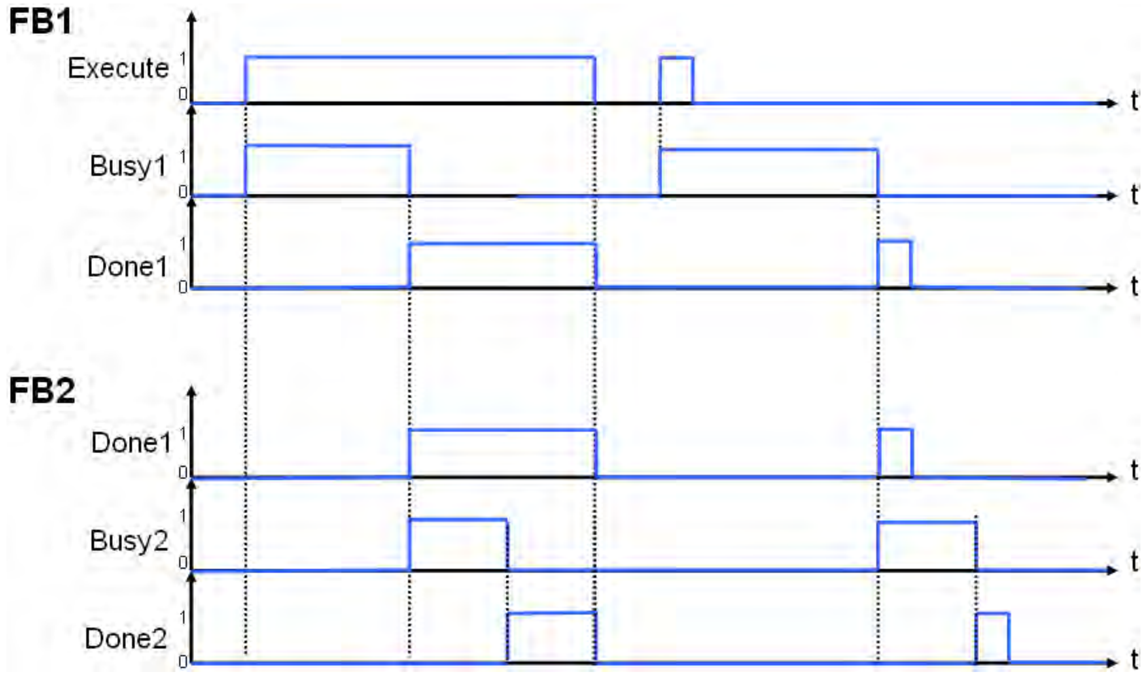
(a) Write the value of a parameter into the servo drive specified by means of the motion control function block named FB1.

(b) After the execution of the motion control function block named FB1 is complete, the motion control function block named FB2 will be executed automatically.

(c) After the execution of the motion control function block name FB2 is complete, Done2 will be set to True. The value of the Value of output pin in the motion control function block named FB2 is the value read from the servo drive specified.

(d) The value of the Value of output pin in the motion control function block named FB2 should be the same as the value of the Value input pin in the motion control function block named FB1.

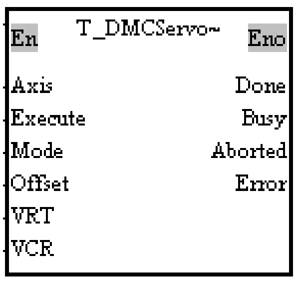
Timing diagram:




5. Modules which are supported

The motion control function block T_DMCServoRead supports AH20MC-5A.

### 5.12.5 Instructing a Servo Drive to Return Home




1. Motion control function block

The motion control function block T_DMCServoHoming is used to instruct the servo drive specified on a DMCNET to return home. The value of the Axis input pin indicates an axis number, and the value of the Mode input pin indicates a mode of returning home. After the servo drive specified returns home, the value of the Offset input pin will indicate an offset.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | 1~12 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Execute | Motion is started when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Mode | Mode of returning home | WORD | 1~35 | The value of the Mode input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Offset | Offset | WORD | K-32,767~K32,767 | The value of the Offset input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| VRT | Speed at which the servo drive specified returns home<br>Unit: RPM | DWORD | K1~K2000 | The value of the VRT input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| VCR | Speed to which the speed of the servo drive specified decreases<br>Unit: RPM | DWORD | K1~K500 | The value of the VCR input pin is valid when there is a transition in the Execute input pin's signal form low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | ● There is a transition in the Done output pin's signal from low to high when motion of returning home is complete. | ● There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | ● The execution of the motion control function block is interrupted by a command. | ● There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

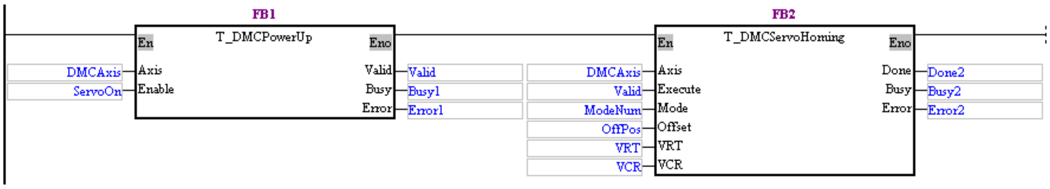
| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Example

Purpose:

- The motion control function block T_DMCPowerUp is used to start the servo drive specified, and then the motion control function block T_DMCServoHoming is used to instruct the servo drive to return home in the way specified.

The motion control function block named FB1 is set to that the servo drive specified is started. The motion control function block named FB2 is set so that the servo drive returns home.



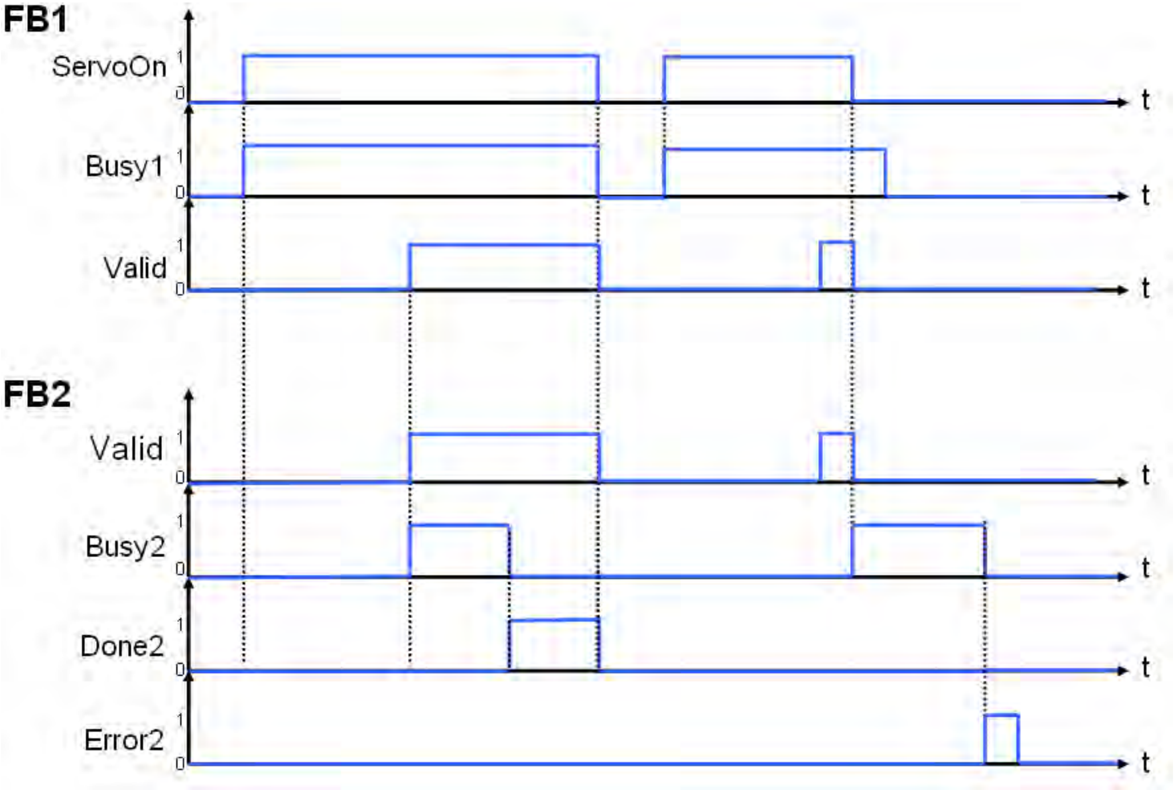

Steps:

(a) After the execution of the motion control function block named FB1 is complete, the servo drive specified will be started.

(b) The motion control function block named FB2 is executed automatically.

(c) ModeNum determines the mode of returning home.

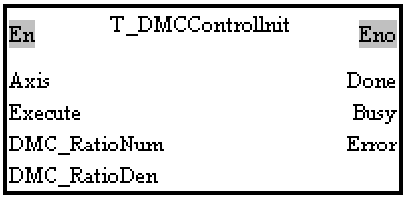
Timing diagram:




5. Modules which are supported

   The motion control function block T_DMCServoHoming supports AH20MC-5A.

## 5.12.6 Initializing a Servo Drive




1. Motion control function block

   The motion control function block T_DMCControlInit is used to initialize the servo drive specified on a DMCNET. The value of the Axis input pin indicates an axis number. The value of the DMC-RatioNum is the numerator of an electronic gear ratio. The value of the DMC-RatioDen is the denominator of an electronic gear ratio.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | 1~12 | The value of the Axis input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | The servo drive specified is initialized when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| DMC_RatioNum | Numerator of an electronic gear ratio | WORD | K1~K32,767 | The value of the DMC_RatioNum input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| DMC_RatioDen | Denominator of an electronic gear ratio | WORD | K1~K32,767 | The value of the DMC_RatioDen input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | ● There is a transition in the Done output pin's signal from low to high when the initialization of the servo drive specified is complete. | ● There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from |

5

| | | | | |
|---|---|---|---|---|
| | | | | low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

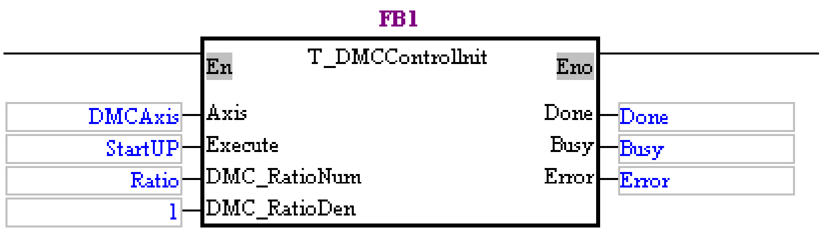
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

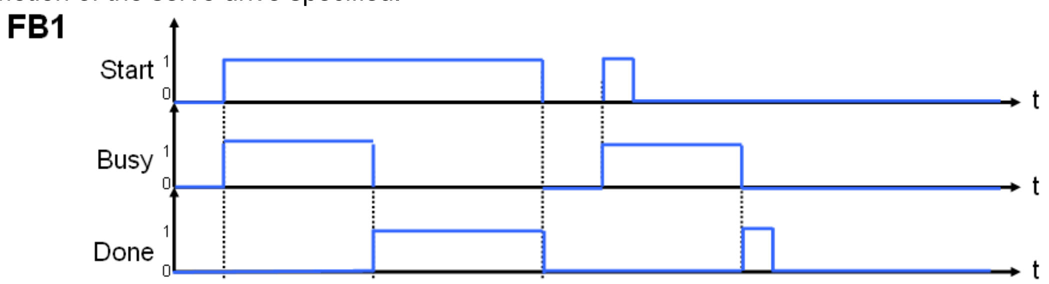
4. Example

Purpose:

- The motion control function block T_DMCControlInit is used to initialize the servo drive specified on a DMCNET, and set an electronic gear ratio.



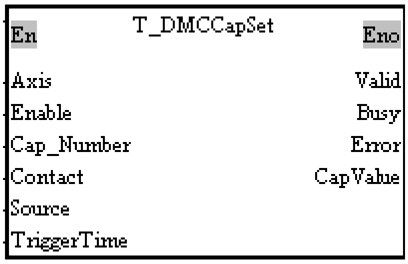

The motion control function block T_DMCControlInit can be used to set an electronic gear ratio. After the execution of the motion control function block T_DMCControlInit is complete, a uniaxial motion control function block or a multiaxial motion control function block can be used to start motion of the servo drive specified.




5. Modules which are supported

The motion control function block T_DMCControlInit supports AH20MC-5A.

### 5.12.7 Instructing a Servo Drive to Capture Values




1. Motion control function block
   The motion control function block T_DMCCapSet is used to instruct the servo drive specified on a DMCNET to capture values. The value of the Axis input pin indicates an axis number. The value of the CAP_Number input pin is the number of values which will be captured. Users can set a capture signal by means of the Contact input pin, and set the source of the values which will be captured. The value of the TriggerTime input pin indicates a minimum time interval, and the value of the CapValue of output pin is the value which is captured.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Axis | Motion axis number | WORD | K1~K12 | The value of the Axis input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | The servo drive is instructed to capture values when there is a transition in the Enable input pin's signal from low to high. | BOOL | True/False | - |
| Cap_Number | Number of values captured | WORD | K1~K400 | The value of the Cap_Number input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Contact | Setting a capture signal | BOOL | True/False | The value of the Contact input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Source | Source of the values captured | WORD | K0~K3 (*1) | The value of the Source input pin is valid when there is a transition in the Enable input pin's signal from low to high. |

| **Input pin** | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| TriggerTime | Minimum time interval (Unit: ms) | WORD | K0~K1000 | The value of the TriggerTime input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | ● There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect.<br>● The source specified has been occupied. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| **Value of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| CapValue | Vale which is captured | DWORD | K0~K2,147,483,647 | When the Valid output pin is set to True, the value of the CapValue of output pin is updated repeatedly. |

*1: Value of the Source input pin

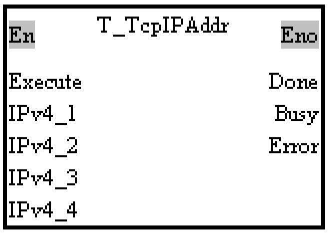
| Value | Definition |
|---|---|
| 0 | Invalid |
| 1 | Auxiliary encoder |
| 2 | Pulse command |
| 3 | Main encoder |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_DMCCapSet supports AH20MC-5A.

## 5.12.8 Setting an Ethernet IP Address




1. Motion control function block
   The motion control function block T_TcpIPAddr is used to set the Ethernet IP address of the module used. Users can set an IP address by means of the IPv4_1 input pin, the IPv4_2 input pin, the IPv4_3 input pin, and the IPv4_4 input pin.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | The values of parameters are written when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| IPv4_1 | First byte of an IP address | WORD | K0~255 | The value of the IPv4_1 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| IPv4_2 | Second byte of an IP address | WORD | K0~255 | The value of the IPv4_2 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| IPv4_3 | Third byte of an IP address | WORD | K0~255 | The value of the IPv4_3 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| IPv4_4 | Fourth byte of an IP address | WORD | K0~K255 | The value of the IPv4_4 input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | ● There is a transition in the Done output pin's signal from low to high when the setting of the Ethernet IP address of the module used is complete. | ● There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>● If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

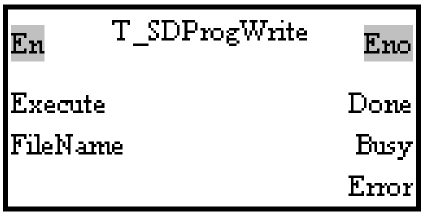
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_TcpIPAddr supports AH10PM-5A, AH15PM-5A, and AH20MC-5A.

# 5.13 Other Motion Control Function Blocks

## 5.13.1 Backing a Main Program up onto an SD Card




1. Motion control function block
   The motion control function block T_SDProgWrite is used to back a main program up onto an SD card. The value of the FileName input pin indicates a filename.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | A main program is backed up when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| FileName | Filename | WORD | K0~4095 | The value of the FileName input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when the making of a backup of a main program is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |

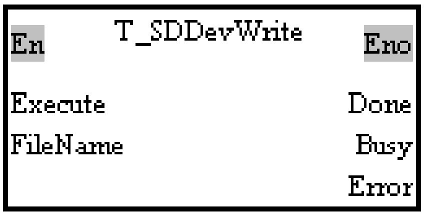
| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_SDProgWrite supports AH10PM-5A, AH15PM, and AH20MC-5A.

### 5.13.2 Backing the Values in Devices up onto an SD Card




1. Motion control function block
   The motion control function block T_SDDevWrite is used to back the values in the devices in a module up onto an SD card. The value of the FileName input pin indicates a filename.

2. Input pins/Output pins

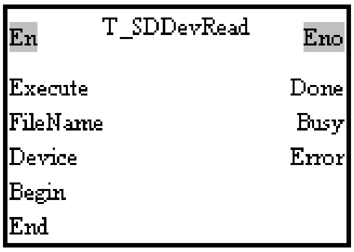
| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | The values in the devices in a module are backed up when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| FileName | Filename | WORD | K0~K4095 | The value of the FileName input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when the making of a backup of the values in the devices in a module is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
The motion control function block T_SDDevWrite supports AH10PM-5A, AH15PM, and AH20MC-5A.

### 5.13.3 Restoring the Values in Devices in an SD Card




1. Motion control function block
The motion control function block T_SDDevRead is used to read the values in the devices specified from the file specified in an SD card. The value of the FileName input pin indicates a filename, and the value of the Device input pin indicates a device type. The value of the Begin input pin indicates a starting device, and the value of the End input pin indicates a terminal device.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Execute | The restoration of the values in devices in an SD card is enabled when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| FileName | Filename | WORD | K0~4095 | The value of the FileName input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Device | Device type | WORD | mcSD_M (0): M device<br>mcSD_D (5): D device<br>mcSD_W (6): W device | The value of the Device input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| Begin | Starting device | WORD | M: K0~4,096<br>D: K0~9,999<br>W: K0~65,535 | The value of the Begin input pin is valid when there is a transition in the Execute input pin's signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| End | Terminal device | WORD | M: K 0~4,096<br>D: K0~9,999<br>W: K0~65,535 | The value of the End input pin is valid when there is a transition in the Execute input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal from low to high when the restoration of the values in devices in an SD card is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |

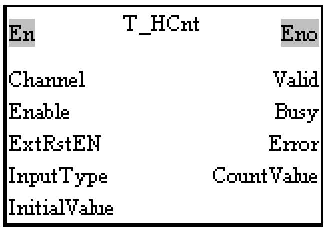
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported

The motion control function block T_SDDevRead supports AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.13.4 High-speed Counter




1. Motion control function block
   The motion control function block T_HCnt is used to start a high-speed counter. The value of the Channel input pin indicates a counter number, and the value of the InputType input pin indicates an input pulse type. The ExtRstEN input pin is used to set an external reset switch. The value of the InitialValue input pin is the initial value in the counter specified, and the value of the CountValue of output pin is the value in the counter specified.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Channel | Counter number | WORD | 0~5 (*1) | The value of the Channel input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | Eanbling the counter specified | BOOL | True/False | - |
| ExtRstEN | Enabling an external reset input terminal | BOOL | True/False | The value of the ExtRstEN input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| InputType | Input pulse type | WORD | mcUD: 0<br>mcPD: 1<br>mcAB: 2<br>mc4AB: 3 | When the motion control function block is executed, the value of the InputType input pin is updated repeatedly. |
| InitialValue | Initial value in the counter specified | DWORD | K0~K2,147,483,647 | The value of the InitialValue input pin is valid when there is a transition in the Enable input pin's signal from low to high. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | ● There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect.<br>● The source specified has been occupied. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

| Value of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| CountValue | Value in the counter specified | DWORD | K0~2,147,483,647 | When the Valid output pin is set to True, the value of the CountValue of output pin is updated repeatedly. |

*1: Value of the Channel input pin

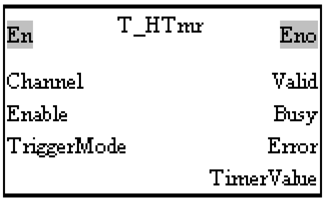
| Value | Definition | Terminal |
|---|---|---|
| 0 | C200 | X0.8, X0.9 |
| 1 | C204 | X0.10, X0.11 |
| 2 | C208 | X0.12, X0.13 |
| 3 | C212 | X0.14, X0.15 |
| 4 | C216 | X0.12, X0.13 |
| 5 | C220 | X0.14, X0.15 |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
The motion control function block T_HCnt supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.13.5 High-speed Timer




1. Motion control function block
   The motion control function block T_HTmr is used to start a high-speed timer. The value of the Channel input pin indicates a timer number, the value of the TriggerMode indicates a mode of triggering the measurement of time, and the value of the TimerValue of output pin the value in the timer specified.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Channel | Timer number | WORD | 0~3 (*1) | The value of the Channel input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | Eanbling the timer specified | BOOL | True/False | - |
| TriggerMode | Mode of triggering the measurement of timer | BOOL | mcUp_Down: False<br>mcUp_Up: True | When the motion control function block is executed, the value of the TriggerMode input pin is updated repeatedly. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | • There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect.<br>● The source specified has been occupied. | There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

| Value of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| TimerValue | Value in the timer specified | DWORD | K0~K2,147,483,647 | When the motion control function block is executed, the value of the TimerValue of output pin is updated repeatedly. If there is no trigger, the value in the timer specified will remain unchanged. |

*1: Value of the Channel input pin

| Value | Definition | Terminal |
|---|---|---|
| 0 | C200 | X0.0 |
| 1 | C204 | X0.1 |
| 2 | C208 | X0.2 |
| 3 | C212 | X0.3 |

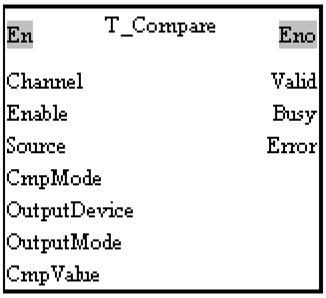
3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported

The motion control function block T_HTmr supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

### 5.13.6 Setting High-speed Comparison




1. Motion control function block
   The motion control function block T_Compare is used to start high-speed comparison. The value of the Channel input pin indicates a comparator number, the value of the Source input pin indicates a source, the value of the CmpMode input pin indicates a comparison condition, and the value of the OutputDevice indicates an output device.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Channel | Comparator number | WORD | 0~7 | The value of the Channel input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | The motion control function block is enabled when there is a transition in the Enable input pin's signal from low to high. | BOOL | True/False | - |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Source | Source | WORD | mcCmpAxis1 (0): Present position of the first axis<br>mcCmpAxis2 (1): Present position of the second axis<br>mcCmpAxis3 (2): Present position of the third axis<br>mcCmpAxis4 (3): Present position of the fourth axis<br>mcCmpC200 (4): Present value in C200<br>mcCmpC204 (5): Present value in C204<br>mcCmpC208 (6): Present value in C208<br>mcCmpC212 (7): Present value in C212 | The value of the Source input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| CmpMode | Comparison condition | WORD | 0: =<br>1: ≧<br>2: ≦ | The value of the CmpMode input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| OutputDevice | Output device | WORD | mcCmpY8 (0): Y0.8<br>mcCmpY9 (1): Y0.9<br>mcCmpY10 (2): Y0.10<br>mcCmpY11 (3): Y0.11<br>mcCmpRstC200 (4): C200<br>mcCmpRstC204 (5): C204<br>mcCmpRstC208 (6): C208<br>mcCmpRstC212 (7): C212 | The value of the OutputDevice input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| OutputMode | Output mode | BOOL | mcCmpSet: True<br>mcCmpRst: False | The value of the OutputMode input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| CmpValue | Value with which a source is compared | DWORD | K-2,147,483,647~ K2,147,483,647 | The value of the CmpValue input pin is valid when there is a transition in the Enable input pin's signal from low to high. |

5

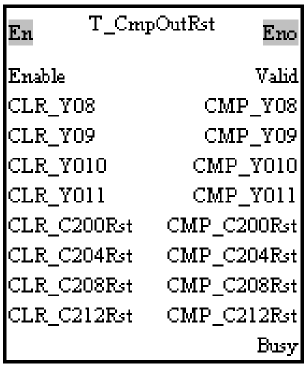
| State of output pin | | | | |
|---|---|---|---|---|
| Name | Function | Data type | Time when there is a transition in an output pin's signal from low to high | Time when there is a transition in an output pin's signal from high to low |
| Valid | An output value is valid. | BOOL | • There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The source specified has been occupied. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |
| The comparator specified has been used. | Use another comparator. |

4. Modules which are supported
The motion control function block T_Compare supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.13.7 Resetting High-speed Comparison




1. Motion control function block
   The motion control function block T_CmpOutRst is used to reset high-speed comparison, and check the comparison conditions used. CLR_Y08, CLR_Y09, CLR_Y010, CLR_Y011, CLR_C200Rst, CLR_C204Rst, CLR_C208Rst, and CLR_C212Rst determine the output devices which will be reset.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Enable | The motion control function block is enabled when there is a transition in the Enable input pin's signal from low to high. | BOOL | True/False | - |
| CLR_Y08<br>CLR_Y09<br>CLR_Y010<br>CLR_Y011<br>CLR_C200Rst<br>CLR_C204Rst<br>CLR_C208Rst<br>CLR_C212Rst | Resetting the output devices Y0.8, Y0.9, Y0.10, Y0.11, C200, C204, C208, and C212 | BOOL | True/False | When the motion control function block is executed, the values of these input pins are updated repeatedly. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | • There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Busy | The motion control function block is being executed. | BOOL | ● There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | ● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>● There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | ● Input values are incorrect.<br>● The source specified has been occupied. | ● There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

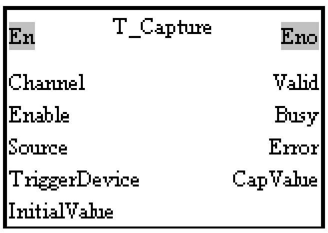
| Value of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| CMP_Y08 | States of the output devices Y0.8, Y0.9, Y0.10, Y0.11, C200, C204, C208, and C212 | BOOL | True/False | When the Valid output pin is set to True, the values of these output pins are updated repeatedly. |
| CMP_Y09 | | | | |
| CMP_Y010 | | | | |
| CMP_Y011 | | | | |
| CMP_C200Rst | | | | |
| CMP_C204Rst | | | | |
| CMP_C208Rst | | | | |
| CMP_C212Rst | | | | |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_CmpOutRst supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.13.8 Setting High-speed Capture



1. Motion control function block
   The motion control function block T_Capture is used to start high-speed capture. The value of the Channel input pin indicates a capturer number. The value of the Source input pin indicates a source, the value of the TriggerDevice input pin indicates the device which triggers the capture of a value, the value of the InitialValue input pin is an initial value, and the value of the CapValue of output pin is the value captured.
2. Input pins/Output pins

| **Input pin** | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Channel | Capturer number | WORD | 0~7 | The value of the Channel input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | The motion control function block is enabled when there is a transition in the Enable input pin's signal from low to high. | BOOL | True/False | - |
| Source | Source | WORD | mcCmpAxis1 (0): Present position of the first axis<br>mcCmpAxis2 (1): Present position of the second axis<br>mcCmpAxis3 (2): Present position of the third axis<br>mcCmpAxis4 (3):Present position of the fourth axis<br>mcCmpC200 (4): Present value in C200<br>mcCmpC204 (5): Present value in C204<br>mcCmpC208 (6): Present value in C208<br>mcCmpC212 (7): Present value in C212 | The value of the Source input pin is valid when there is a transition in the Enable input pin's signal from low to high. |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| TriggerDevice | Device which triggers the capture of a value | WORD | mcCapX0 (0): X0.0<br>mcCapX1 (1): X0.1<br>mcCapX2 (2): X0.2<br>mcCapX3 (3): X0.3<br>mcCapX8 (8): X0.8<br>mcCapX9 (9): X0.9<br>mcCapX10 (10): X0.10<br>mcCapX11 (11): X0.11<br>mcCapX12 (12): X0.12<br>mcCapX13 (13): X0.13<br>mcCapX14 (14): X0.14<br>mcCapX15 (15): X0.15 | The value of the TriggerDevice input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| InitialValue | Initial value | DWORD | K-2,147,483,648~ K2,147,483,647 | The value of the InitialValue input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | • There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

5

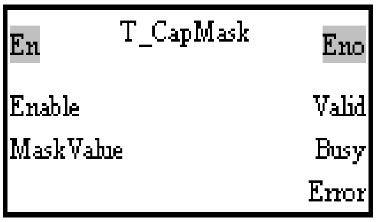
| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The source specified has been occupied. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| **Value of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Time when a value is valid** |
| CapValue | Value which is captured | DWORD | K-2,147,483,648~ K2,147,483,647 | When the motion control function block is executed, the value of the CapValue of output pin is updated repeatedly. If there is no trigger, the value captured will remain unchanged. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |
| The capturer specified has been used. | Use another capturer. |

4. Modules which are supported
   The motion control function block T_Capture supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.13.9 High-speed Masking




1. Motion control function block
   The motion control function block T_CapMask is used to start high-speed masking. The MaskValue input pin determines the range which will be masked.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Enable | The motion control function block is enabled when there is a transition in the Enable input pin's signal from low to high. | BOOL | True/False | - |
| MaskValue | Range which is masked | DWORD | K1~2,147,483,647 | When the motion control function block is executed, the value of the MaskValue input pin is updated repeatedly. |

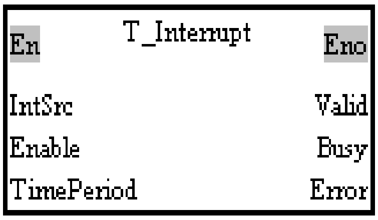
| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Valid | An output value is valid. | BOOL | • There is a transition in the Valid output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The source specified has been occupied. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_CapMask supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

## 5.13.10 Setting an Interrupt




1. Motion control function block
   The motion control function block T_Interrupt is used to set the trigger for an interrupt subroutine. The value of the IntSrc input pin indicates the trigger for an interrupt subroutine. If the interrupt set is a time interrupt, the value of the TimePeriod input pin indicates the cycle of the interrupt.
2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| IntSrc | Trigger for an interrupt subroutine | WORD | IntTimer 0<br>IntX8:1<br>IntX9:2<br>IntX10:3<br>IntX11:4<br>IntX12:5<br>IntX13:6<br>IntX14:7<br>IntX15:8 | The value of the IntSrc input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Enable | The motion control function block is enabled when there is a transition in the Enable input pin's signal from low to high. | BOOL | True/False | - |
| TimePeriod | Cycle of a time interrupt<br>(Unit: ms)<br>(Not applicable to terminal interrupts) | WORD | K1~K65,535 | When the motion control function block is executed, the value of the TimePeriod input pin is updated repeatedly. |

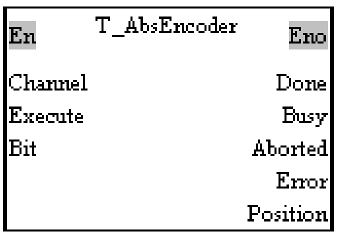
| State of output pin | | | | |
|---|---|---|---|---|
| Name | Function | Data type | Time when there is a transition in an output pin's signal from low to high | Time when there is a transition in an output pin's signal from high to low |
| Valid | An interrupt is enabled. | BOOL | • There is a transition in the Valid output pin's signal from low to high when an interrupt is enabled. | • There is a transition in the Valid output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Enable input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The source specified has been occupied. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Enable input pin's signal from high to low. |

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_Interrupt supports AH05PM-5A, AH10PM-5A, AH15PM-5A, and AH20MC-5A.

### 5.13.11 Absolute Encoder




1. Motion control function block
   The motion control function block T_AbsEncoder is used to start the reading of the position of an absolute encoder.

2. Input pins/Output pins

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Time when a value is valid** |
| Channel | Group number | WORD | K1~K4 (*1) | The value of the Channel input pin is valid when there is a transition in the Enable input pin's signal from low to high. |
| Execute | The reading of the position of an absolute encoder is enabled when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | - |
| Bit | Resolution of an absolute encoder | DWORD | K1~K32 (*2) | When the motion control function block is executed, the value of the Bit input pin is updated repeatedly. |
| **State of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Done | The execution of the motion control function block is complete. | BOOL | • There is a transition in the Done output pin's signal when the reading of the position of an absolute encoder is complete. | • There is a transition in the Done output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when motion is complete, the Done output pin will be set to False in the next cycle. |
| Busy | The motion control function block is being executed. | BOOL | • There is a transition in the Busy output pin's signal from low to high when there is a transition in the Execute input pin's signal from low to high. | • There is a transition in the Busy output pin's signal from high to low when there is a transition in the Done output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Error output pin's signal from low to high.<br>• There is a transition in the Busy output pin's signal from high to low when there is a transition in the Aborted output pin's signal from low to high. |

| State of output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Time when there is a transition in an output pin's signal from low to high** | **Time when there is a transition in an output pin's signal from high to low** |
| Aborted | The execution of the motion control function block is interrupted by a command. | BOOL | • The execution of the motion control function block is interrupted by a command. | • There is a transition in the Aborted output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low.<br>• If the Execute input pin is set to False when the execution of the motion control function block is interrupted, the Aborted output pin will be set to False in the next cycle. |
| Error | An error occurs in the motion control function block. | BOOL | • Input values are incorrect.<br>• The axis specified is in motion before the motion control function block is executed. | • There is a transition in the Error output pin's signal from high to low when there is a transition in the Execute input pin's signal from high to low. |
| **Value of output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Update** |
| Position | Present position of an encoder | DWORD | K0~K2,147,483,647 | When there is a transition in the Done output pin's signal from low to high, the value of the Position output pin is updated. |

*1: Wiring hardware

| Group number | T+ | T- | D+ | D- |
|---|---|---|---|---|
| **1** | Y0.0+ | Y0.0- | X0.0+ | X0.0- |
| **2** | Y0.2+ | Y0.2- | X0.1+ | X0.1- |
| **3** | Y0.4+ | Y0.4- | X0.2+ | X0.2- |
| **4** | Y0.6+ | Y0.6- | X0.3+ | X0.3- |

*2: Setting the resolution of an encoder

Specifications for an SSI encoder:

| Item | Specification |
|---|---|
| **Resolution per rotation** | 8192 (13 bits) |
| **Number of rotations** | 4096 (12 bits) |

Resolution of an encoder: Resolution per rotation+Number of rotations+1=13+12+1=26

3. Troubleshooting

| Error | Troubleshooting |
|---|---|
| The values of input pins in the motion control function block are incorrect. | Check whether the values of the input pins are in the ranges allowed. |

4. Modules which are supported
   The motion control function block T_AbsEncoder supports AH10PM-5A.

**MEMO**

# Chapter 6 Data Transmission

## Table of Contents

## 6.1 Functions

Users can set the way in which an AH500 series CPU module exchange data with an AH500 series motion control module.

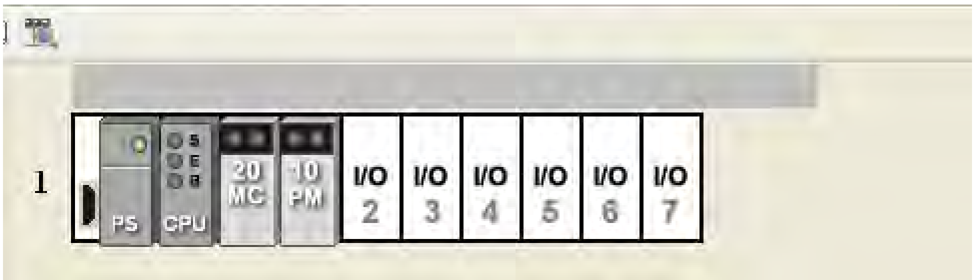
1. The AH500 series CPU module writes values in 400 word devices in the AH500 series motion control module.
2. The AH500 series motion control module writes data into 400 word devices in the AH500 series CPU module.
3. The AH500 series CPU module writes data into 400 bit devices in the AH500 series motion control module.
4. The AH500 series motion control module writes data into 400 bit devices into the AH500 series CPU module.

The users can control or monitor the AH500 series motion control module by means of the 400 word devices and the 400 bit devices in the AH500 series motion control module. The users can write values into registers in the AH500 series motion control module by means of the program in the AH500 series motion control module, and the values can be written into the AH500 series CPU module. The users can write values into registers in the AH500 series CPU module by means of the program in the AH500 series CPU module, and the values can be written into the AH500 series motion control module.

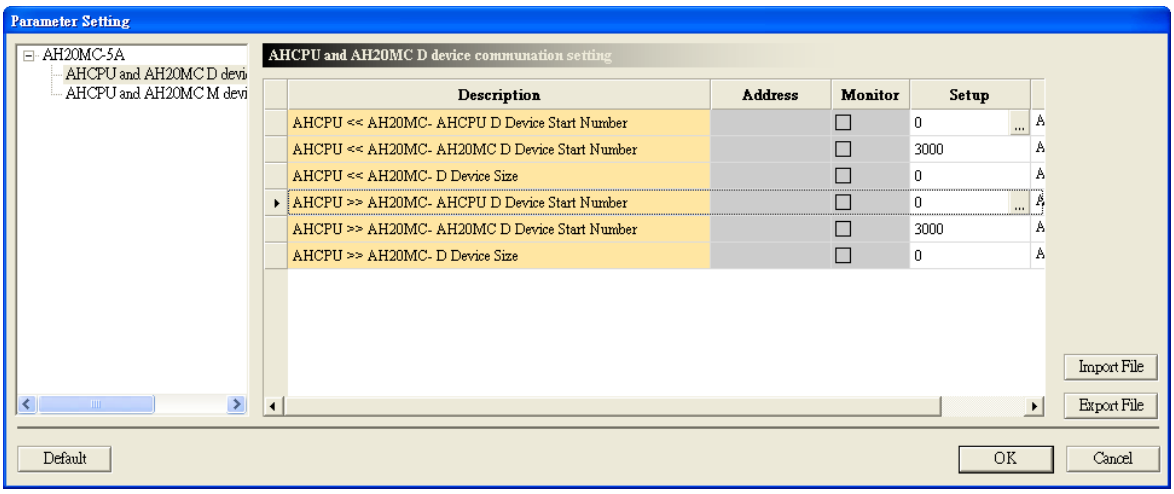
The following sections are about AH20MC-5A.

## 6.2 Parameters

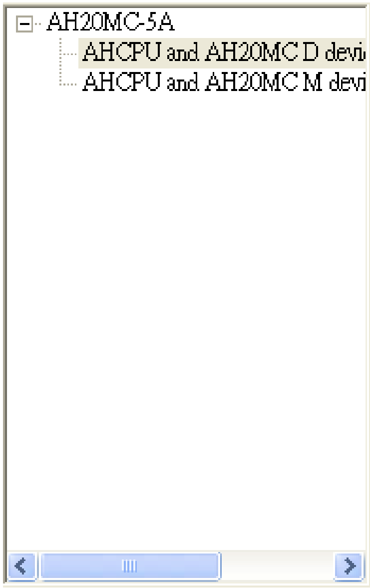
The **HWCONFIG** window in ISPSoft is shown below.




After users double-click AH20MC-5A in the **HWCONFIG** window, the **Parameter Setting** window will appear.



There is a parameter list at the left side of the **Parameter Setting** window. **AHCPU and AH20MC D device communication setting** and **AHCPU and AH20MC M device communication setting** are on the list.




The items at the right side of the **Parameter Setting** window are detailed parameters.

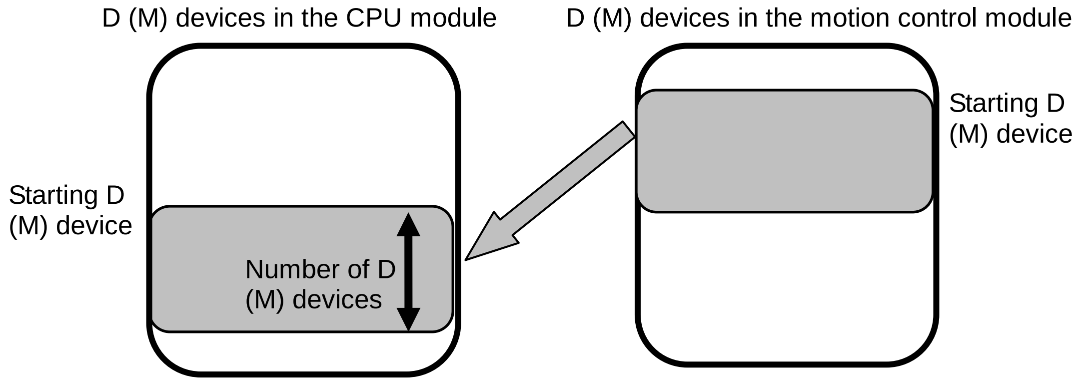
AHCPU and AH20MC D device communation setting

| Description | Address | Monitor | Setup |
|---|---|---|---|
| AHCPU << AH20MC- AHCPU D Device Start Number | | ☐ | 0 |
| AHCPU << AH20MC- AH20MC D Device Start Number | | ☐ | 3000 |
| AHCPU << AH20MC- D Device Size | | ☐ | 0 |
| AHCPU >> AH20MC- AHCPU D Device Start Number | | ☐ | 0 |
| AHCPU >> AH20MC- AH20MC D Device Start Number | | ☐ | 3000 |
| AHCPU >> AH20MC- D Device Size | | ☐ | 0 |

The detailed parameters at the right side of the **Parameter Setting** window are described below.

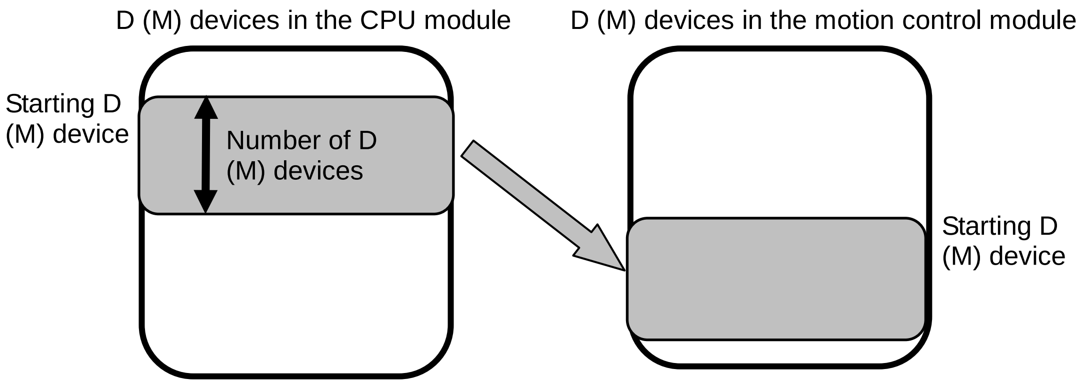
| | | | |
|---|---|---|---|
| AHCPU << AH20MC- AHCPU D Device Start Number | | ☐ | 0 |
| AHCPU << AH20MC- AH20MC D Device Start Number | | ☐ | 3000 |
| AHCPU << AH20MC- D Device Size | | ☐ | 0 |

- **AHCPU<<AH20MC-AHCPU D (M) Device Start Number**: After the AH500 series CPU module reads the values in D (M) devices in AH20MC-5A, the values will be stored in the devices starting from a device in the AH500 series CPU module.
- **AHCPU<<AH20MC-AH20MC D (M) Device Start Number**: The AH500 series CPU module reads the values in the D (M) devices starting from a D (M) device in AH20MC-5A.
- **AHCPU<<AH20MC-D (M) Device Size**: The AH500 series CPU module reads the values in a certain number of D (M) devices in AH20MC-5A.




| | | |
|---|---|---|
| AHCPU >> AH20MC- AHCPU D Device Start Number | ☐ | 0 |
| AHCPU >> AH20MC- AH20MC D Device Start Number | ☐ | 3000 |
| AHCPU >> AH20MC- D Device Size | ☐ | 0 |

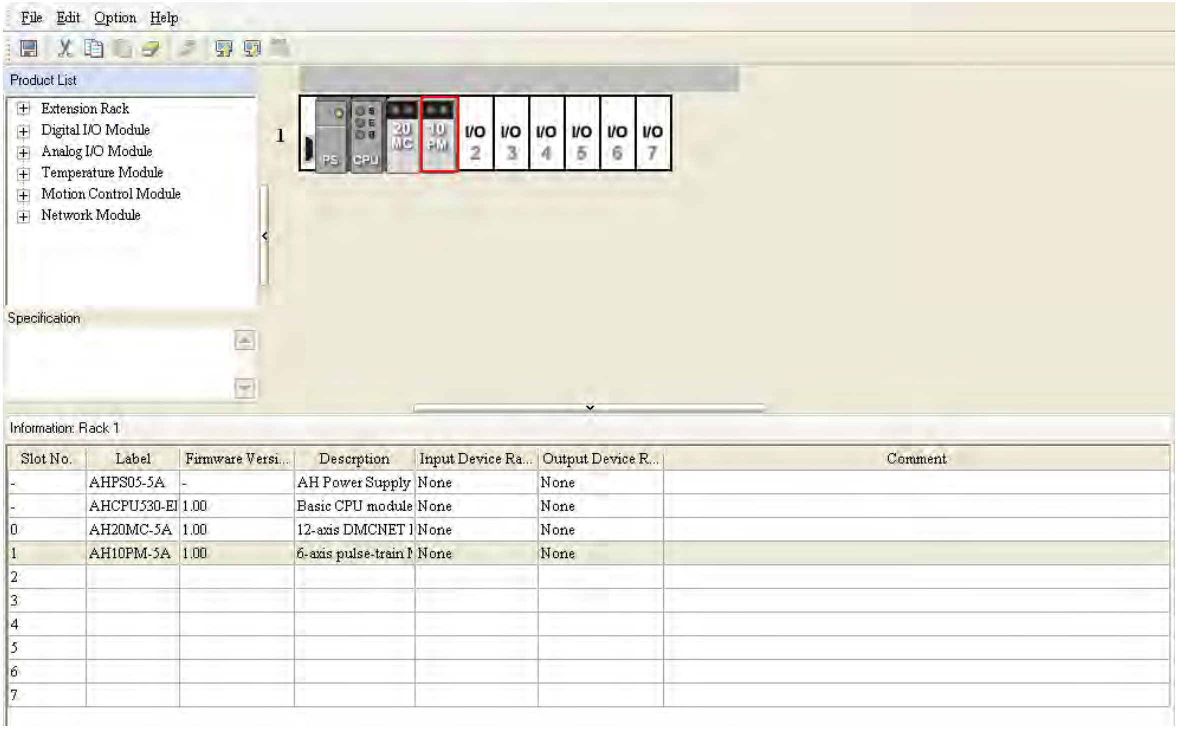
- **AHCPU>>AH20MC-AHCPU D (M) Device Start Number**: The values in the devices starting from a device in the AH500 series CPU module is written into D (M) devices in AH20MC-5A.
- **AHCPU>>AH20MC-AH20MC D (M) Device Start Number**: The AH500 series CPU module writes values into the D (M) devices starting from a D (M) device in AH20MC-5A.
- **AHCPU>>AH20MC-D (M) Device Size**: The AH500 series CPU module writes values into a certain number of D (M) devices in AH20MC-5A.



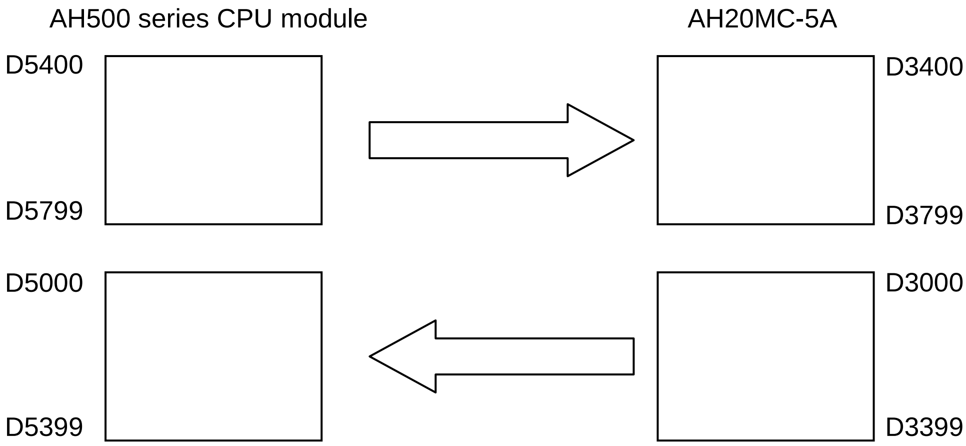
## 6.3 Usage

The steps of using ISPSoft are as follows.

1. Click **I/O Scan** on the toolbar in the **HWCONFIG** window.




2. Set the number of values which will be exchanged, and the devices in which the values exchanged will be stored.



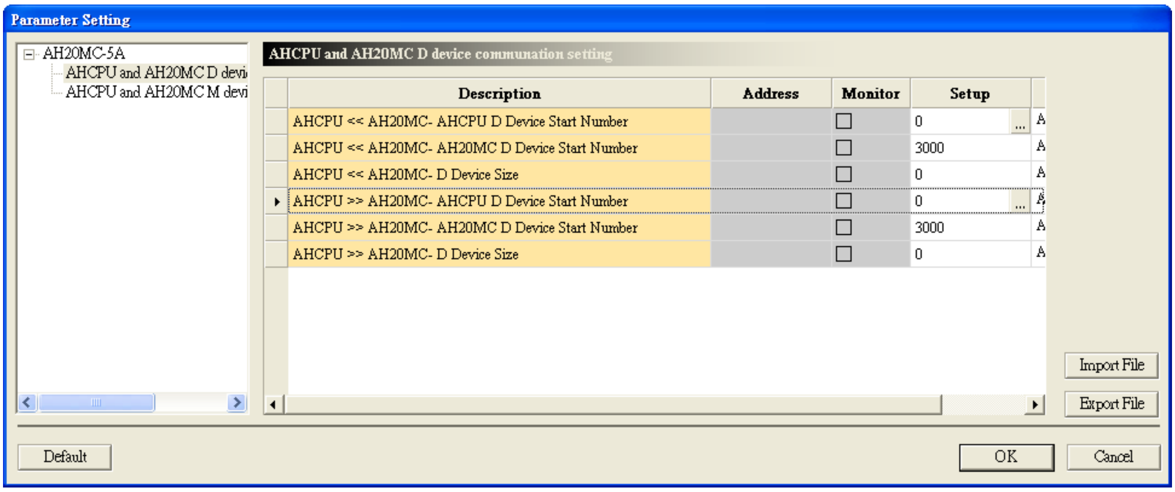

The AH500 series CPU module writes values into 400 devices in AH20MC-5A, and reads values in 400 D devices in AH20MC-5A. The values in D5400~D5799 in the AH500 series CPU module are written into D3400~D3799 in AH20MC-5A. The values in D3000~D3399 in AH20MC are read, and stored in D5000~D5399 in the AH500 series CPU module.

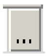


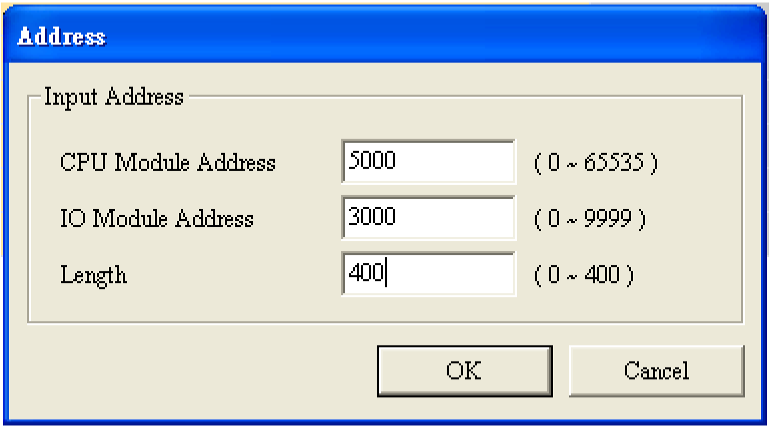
D3399

3. HWCONFIG

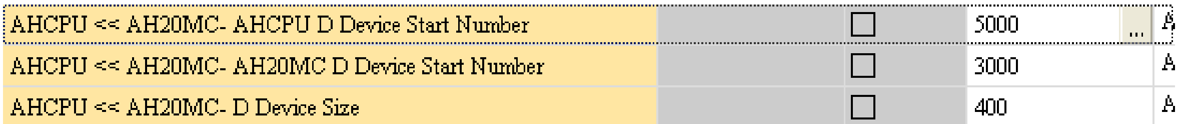
After users double-click AH20MC-5A in the **HWCONFIG** window, the **Parameter Setting** window will appear.



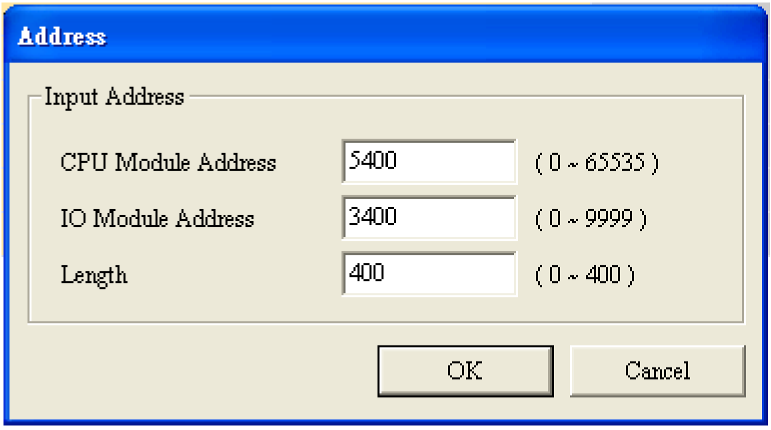

Click  in the **Setup** cell for **AHCPU<<AH20MC-AHCPU D (M) Device Start Number**. In the **Address** window, type 5000 in the **CPU Module Address** box, type 3000 in the **IO Module Address** box, and type 400 in the **Length** box.




Click **OK** in the **Address** window.



Click ... in the **Setup** cell for **AHCPU>>AH20MC-AHCPU D (M) Device Start Number**. In the **Address** window, type 5400 in the **CPU Module Address** box, type 3400 in the **IO Module Address** box, and type 400 in the **Length** box.




Click **OK** in the **Address** window.

| | | | |
|---|---|---|---|
| AHCPU >> AH20MC- AHCPU D Device Start Number | ☐ | 5400 | ... |
| AHCPU >> AH20MC- AH20MC D Device Start Number | ☐ | 3400 | |
| AHCPU >> AH20MC- D Device Size | ☐ | 400 | |

4. Program created in PMSoft

   After AH20MC-5A exchanging values with the AH500 series CPU module, it can write the values gotten from the AH500 series CPU module into SR registers by means of a program created in PMSoft. The program below is about single-speed motion.


The values in D3401 and D3402 are written into SR1023 and SR1024. The values in SR1023 and SR1024 indicate the target position of the first axis. The value in D3403 is written into SR1030. The value in SR1030 indicates an operation command. Besides, the value in D3000 is determined by the state of SM1048. The AH500 series CPU module can judge whether the single-speed motion is complete by means of the value in D3000.

**MEMO**

# Chapter 7 Uniaxial Motion

## Table of Contents

# 7.1 Functions of Uniaxial Motion

The special data registers for motion axes are described below

| SR number (1+N)th axis[*4] HW[*1] | SR number (1+N)th axis[*4] LW[*1] | Function | Setting range | Factory setting |
|---|---|---|---|---|
| - | SR1000+100*N | Setting the parameters of the axis specified | Bit 0~bit 15 | 16#0 |
| - | SR1001+100*N | Compensation value for the axis specified | Users can set SR1001+100*N according to their needs. | 16#0 |
| SR1003+100*N | SR1002+100*N | Number of pulses it takes for the motor of the axis specified to rotate once (A) | 1~2,147,483,647 pulses/revolution | K2,000 |
| SR1005+100*N | SR1004+100*N | Distance generated after the motor of the axis specified rotate once (B) | 1~2,147,483,647[*2] | K1,000 |
| SR1007+100*N | SR1006+100*N | Maximum speed ($V_{MAX}$) at which the axis specified rotates | 0~2,147,483,647[*3] | K10,500,000 |
| SR1009+100*N | SR1008+100*N | Start-up speed ($V_{BIAS}$) at which the axis specified rotates | 0~2,147,483,647[*3] | K0 |
| SR1011+100*N | SR1010+100*N | JOG speed ($V_{JOG}$) at which the axis specified rotates | 0~2,147,483,647[*3] | K5,000 |
| SR1013+100*N | SR1012+100*N | Speed ($V_{RT}$) at which the axis specified returns home | 0~2,147,483,647[*3] | K50,000 |
| SR1015+100*N | SR1014+100*N | Speed ($V_{CR}$) to which the speed of the axis specified decreases when the axis returns home | 0~2,147,483,647[*3] | K1,000 |
| - | SR1016+100*N | Number of PG0 pulses for the axis specified | 0~32,767 pulses | K0 |
| - | SR1017+100*N | Supplementary pulses for the axis specified | -32,768~+32,767 pulses | K0 |

| SR number (1+N)th axis[*4] HW[*1] | SR number (1+N)th axis[*4] LW[*1] | Function | Setting range | Factory setting |
|---|---|---|---|---|
| SR1019+100*N | SR1018+100*N | Home position of the axis specified | 0~±999,999[*1] | K0 |
| - | SR1020+100*N | Time ($T_{ACC}$) it takes for the axis specified to accelerate | 10~32,767 ms | K5,100 |
| - | SR1021+100*N | Time ($T_{DEC}$) it takes for the axis specified to decelerate | 10~32,767 ms | K5,100 |
| SR1023+100*N | SR1022+100*N | Target position of the axis specified (P (I)) | -2,147,483,648~+2,147,483,647[*1] | K0 |
| SR1025+100*N | SR1024+100*N | Speed at which the axis specified rotates (V (I)) | 0~2,147,483,647[*1] | K1,000 |
| SR1027+100*N | SR1026+100*N | Target position of the axis specified (P (II)) | -2,147,483,648~+2,147,483,647[*1] | K0 |
| SR1029+100*N | SR1028+100*N | Speed at which the axis specified rotates(V (II)) | 0~2,147,483,647[*2] | K2,000 |
| - | SR1030+100*N | Operation command | Bit 0~bit 15 | 16#0 |
| - | SR1031+100*N | Mode of operation | Bit 0~bit 15 | 16#0 |
| SR1033+100*N | SR1032+100*N | Present command position of the axis specified (Pulse) | -2,147,483,648~+2,147,483,647[*1] | K0 |
| SR1035+100*N | SR1034+100*N | Present command speed of the axis specified (PPS) | 0~2,147,483,647 PPS | K0 |
| SR1037+100*N | SR1036+100*N | Present command position of the axis specified (Unit[*3]) | -2,147,483,648~+2,147,483,647[*1] | K0 |
| SR1039+100*N | SR1038+100*N | Present command speed of the axis specified (Unit[*3]) | 0~2,147,483,647 PPS | K0 |
| - | SR1040+100*N | State of the axis specified | Bit 0~bit 15 | 16#0 |

| SR number (1+N)th axis*4 | | Function | Setting range | Factory setting |
|---|---|---|---|---|
| HW*1 | LW*1 | | | |
| - | SR1041+100*N | Axis error code | Please refer to the error code tables in appendix A. | 16#0 |
| - | SR1042+100*N | Electronic gear ratio of the axis specified (Numerator) | 1~32,767 | K1 |
| - | SR1043+100*N | Electronic gear ratio of the axis specified (Denominator) | 1~32,767 | K1 |
| SR1045+100*N | SR1044+100*N | Frequency of pulses generated by the manual pulse generator for the axis specified | Frequency of pulses generated by the manual pulse generator for the axis specified | K0 |
| SR1047+100*N | SR1046+100*N | Number of pulses generated by the manual pulse generator for the axis specified | Number of pulses generated by the manual pulse generator for the axis specified | K0 |
| - | SR1048+100*N | Response speed of the manual pulse generator for the axis specified | Response speed of the manual pulse generator for the axis specified | K5 |
| - | SR1049+100*N | Mode of stopping Ox0~Ox99 | Users can set SR1049+100*N according to their needs. | K0 |
| SR1051+100*N | SR1050+100*N | Electrical zero of the axis specified | Users can set (SR1051+100*N, SR1050+100*N) according to their needs. | K0 |
| - | SR1052+100*N | Setting an Ox motion subroutine number | Users can set SR1052+100*N according to their needs. | K0 |
| - | SR1053+100*N | Step address in the Ox motion subroutine at which an error occurs | Users can set SR1053+100*N according to their needs. | K0 |
| SR1069+100*N | SR1068+100*N | Present position of the encoder specified on a DMCNET*5 | The value displayed in (SR1069+100*N, SR1068+100*N) is a value set in a Delta ASDA-A2 series servo drive. | K0 |

*1. HW: High word; LW: Low word

*2. Unit: μm/rev, mdeg/rev, and $10^{-4}$ inches/rev

*3. The unit used varies with the setting of bit 0 and bit 1 in SR1000+100*N.

*4. N is in the range of 0 to 15.
*5. Only AH20MC-5A is supported.

The special data registers related to uniaxial motion are described below.

<table>
<tr><th colspan="2">SR number</th><th rowspan="3">Function</th><th colspan="8">Motion</th></tr>
<tr><th colspan="2">(1+N)th axis</th><th rowspan="2">JOG motion</th><th rowspan="2">Returning home</th><th rowspan="2">Single-speed motion</th><th rowspan="2">Inserting single-speed motion</th><th rowspan="2">Two-speed motion</th><th rowspan="2">Inserting two-speed motion</th><th rowspan="2">Variable motion</th><th rowspan="2">Manual pulse generator mode</th></tr>
<tr><th>HW</th><th>LW</th></tr>
<tr><td>-</td><td>SR1000+100*N</td><td>Setting the parameters of the axis specified</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td></tr>
<tr><td>SR1003+100*N</td><td>SR1002+100*N</td><td>Number of pulses it takes for the motor of the axis specified to rotate once (A)</td><td colspan="8" rowspan="2">If the unit selected is a motor unit, users do not need to set SR1002+100*N and SR1003+100*N. If the unit selected is a mechanical unit or a compound unit, users need to set SR1002+100*N and SR1003+100*N.</td></tr>
<tr><td>SR1005+100*N</td><td>SR1004+100*N</td><td>Distance generated after the motor of the axis specified rotate once (B)</td></tr>
<tr><td>SR1007+100*N</td><td>SR1006+100*N</td><td>Maximum speed ($V_{MAX}$) at which the axis specified rotates</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td></tr>
<tr><td>SR1009+100*N</td><td>SR1008+100*N</td><td>Start-up speed ($V_{BIAS}$) at which the axis specified rotates</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td></tr>
<tr><td>SR1011+100*N</td><td>SR1010+100*N</td><td>JOG speed ($V_{JOG}$) at which the axis specified rotates</td><td>◎</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>SR1013+100*N</td><td>SR1012+100*N</td><td>Speed ($V_{RT}$) at which the axis specified returns home</td><td rowspan="4">-</td><td rowspan="4">◎</td><td rowspan="4">-</td><td rowspan="4">-</td><td rowspan="4">-</td><td rowspan="4">-</td><td rowspan="4">-</td><td rowspan="4">-</td></tr>
<tr><td>SR1015+100*N</td><td>SR1014+100*N</td><td>Speed ($V_{CR}$) to which the speed of the axis specified decreases when the axis returns home</td></tr>
<tr><td>-</td><td>SR1016+100*N</td><td>Number of PG0 pulses for the axis specified</td></tr>
<tr><td>-</td><td>SR1017+100*N</td><td>Supplementary pulses for the axis specified</td></tr>
<tr><td>SR1019+100*N</td><td>SR1018+100*N</td><td>Home position of the axis specified</td><td>-</td><td>◎</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>-</td><td>SR1020+100*N</td><td>Time ($T_{ACC}$) it takes for the axis specified to accelerate</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>◎</td><td>-</td></tr>
</table>

| SR number (1+N)th axis HW | SR number (1+N)th axis LW | Function | Motion: JOG motion | Returning home | Single-speed motion | Inserting single-speed motion | Two-speed motion | Inserting two-speed motion | Variable motion | Manual pulse generator mode |
|---|---|---|---|---|---|---|---|---|---|---|
| - | SR1021+100*N | Time ($T_{DEC}$) it takes for the axis specified to decelerate | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | - |
| SR1023+100*N | SR1022+100*N | Target position of the axis specified (P (I)) | - | - | ◎ | ◎ | ◎ | ◎ | - | ◎ |
| SR1025+100*N | SR1024+100*N | Speed at which the axis specified rotates (V (I)) | - | - | ◎ | ◎ | ◎ | ◎ | ◎ | - |
| SR1027+100*N | SR1026+100*N | Target position of the axis specified (P (II)) | - | - | - | - | ◎ | ◎ | - | ◎ |
| SR1029+100*N | SR1028+100*N | Speed at which the axis specified rotates (V (II)) | - | - | - | - | ◎ | ◎ | - | - |
| - | SR1030+100*N | Operation command | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ |
| - | SR1031+100*N | Mode of operation | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ |
| SR1033+100*N | SR1032+100*N | Present command position of the axis specified (Pulse) | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ |
| SR1035+100*N | SR1034+100*N | Present command speed of the axis specified (PPS) | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ |
| SR1037+100*N | SR1036+100*N | Present command position of the axis specified (Unit) | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ |
| SR1039+100*N | SR1038+100*N | Present command speed of the axis specified (Unit) | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ |
| - | SR1042+100*N | Electronic gear ratio of the axis specified (Numerator) | - | - | - | - | - | - | - | ◎ |
| - | SR1043+100*N | Electronic gear ratio of the axis specified (Denominator) | - | - | - | - | - | - | - | ◎ |
| SR1045+100*N | SR1044+100*N | Frequency of pulses generated by the manual pulse generator for the axis specified | - | - | - | - | - | - | - | ◎ |

| SR number | | Function | Motion | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| (1+N)th axis | | | JOG motion | Returning home | Single-speed motion | Inserting single-speed motion | Two-speed motion | Inserting two-speed motion | Variable motion | Manual pulse generator mode |
| **HW** | **LW** | | | | | | | | | |
| SR1047+100*N | SR1046+100*N | Number of pulses generated by the manual pulse generator for the axis specified | - | - | - | - | - | - | - | ◎ |
| - | SR1048+100*N | Response speed of the manual pulse generator for the axis specified | - | - | - | - | - | - | - | ◎ |
| - | SR1049+100*N | Mode of stopping Ox0~Ox99 | - | - | - | - | - | - | - | - |
| SR1051+100*N | SR1050+100*N | Electrical zero of the axis specified | - | - | - | - | - | - | - | - |
| - | SR1052+100*N | Setting an Ox motion subroutine number | - | - | - | - | - | - | - | - |
| - | SR1053+100*N | Step address in the Ox motion subroutine at which an error occurs | - | - | - | - | - | - | - | - |
| SR1069+100*N | SR1068+100*N | Present position of the encoder specified on a DMCNET*1 | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ | ◎ |

*1. Only AH20MC-5A is supported.

Common special data registers are described below.

1. Setting the parameters of the axis specified

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1000+100*N |

[Description]

| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1000+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | Curve | Mode of triggering the calculation of the target position | Relative/Absolute coordinates | Direction in which the motor used rotates | Mode of triggering the return to home | Mode of returning home | Direction in which the axis specified returns home | - | - | Output type (positive logic) | | - | | Unit | |

Bit 0~bit 15 in SR1000+100*N are described below.

- Bit 0 and b1 in SR1000+100*N: Unit

| b1 | b0 | Unit | Description |
|---|---|---|---|
| 0 | 0 | Motor unit | A pulse is a unit. |
| 0 | 1 | Mechanical unit | A micrometer, 10-4 inches, or a degree is a unit. |
| 1 | 0 | Compound unit | Position: A micrometer, 10-4 inches, or a degree is a unit. (Mechanical unit)<br>Speed: A pulse is a unit. (Motor unit) |
| 1 | 1 | | |

| | Motor unit | Compound unit | Mechanical unit |
|---|---|---|---|
| **Position** | pulse | µm | |
| | pulse | mdeg | |
| | pulse | $10^{-4}$ inches | |
| **Speed** | pulse/second | | centimeter/minute |
| | pulse/second | | 10 degrees/minute |
| | pulse/second | | inch/minute |

- Position: Home position of the axis specified, target position of the axis specified (P (I)), target position of the axis specified (P (II)), and present command position of the axis specified
- Speed: Maximum speed ($V_{MAX}$) at which the axis specified rotates, start-up speed ($V_{BIAS}$) at which the axis specified rotates, JOG speed ($V_{JOG}$) at which the axis specified rotates, speed ($V_{RT}$) at which the axis specified returns home, speed ($V_{CR}$) to which the speed of the axis specified decreases when the axis returns home, speed at which the axis specified rotates (V (I)), and speed at which the axis specified rotates (V (II))

- Example 1:
  Bit [1:0]=00⇨Motor unit
  Position: Pulse
  Speed: Pulse/second (PPS)
  Target position of the axis specified (P (I)): 10,000 pulses
  Speed at which the axis specified rotates: 10K PPS
  After the AH500 series motion control module sends 10,000 pulses, the axis specified can move to the target position specified. (The frequency of pulses is 10K PPS.) The distance for which the axis specified can move after a pulse is sent is calculated according to the physical quantity used.
- Example 2:
  Bit [1:0]=01⇨Mechanical unit
  Position: µm
  Speed: Centimeter/minute
  N=0
  (SR1003, SR1002)=1,000 (pulses/revolution)
  (SR1005, SR1004)=100 (micrometers/revolution)
  P (I)=10,000 (micrometers)
  V (I)=6 (centimeters/minute)
  The number of pulses sent by the AH500 series motion control module and the frequency of pulses are calculated below.

$$\text{Distance} = \underbrace{\frac{\text{Distance}}{\text{Revolution}}}_{\text{B}} \times \underbrace{\frac{\text{Revolution}}{\text{Number of pulses}}}_{1/\text{A}} \times \text{Number of pulses}$$

  Number of pulses it takes for the axis specified to move to the target position specified $= \frac{\text{P (I)}\,\mu\text{m}}{\text{B}/\text{A}} = \text{P (I)} \times \frac{\text{A}}{\text{B}} = 100{,}000\ \text{(pulses)}$

  Speed at which the axis specified rotates (V (I)): 6 (centimeters/minute)=60,000/60 (micrometers/second)

$$\text{Speed} = \frac{\text{Distance}}{\text{Time}} = \underbrace{\frac{\text{Distance}}{\text{Revolution}}}_{\text{B}} \times \underbrace{\frac{\text{Revolution}}{\text{Number of pulses}}}_{1/\text{A}} \times \underbrace{\frac{\text{Number of pulses}}{\text{Time}}}_{\text{PPS,pulse/sec}}$$

  The frequency of pulses calculated by the AH500 series motion control module

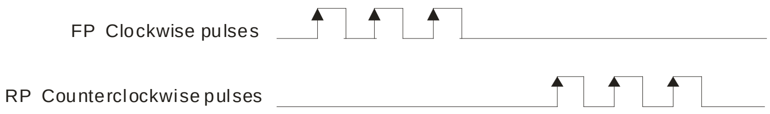
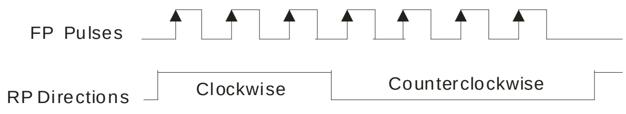
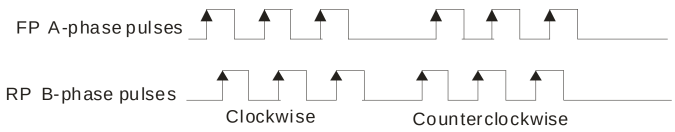
$$= V(I) \times \frac{10^4}{60} \times \frac{A}{B} = \frac{60{,}000}{60} \times \frac{1{,}000}{100} = 10{,}000 \quad \text{(PPS)}$$

- Example 3
  Bit [1:0]=10 or 11⇨Compound unit
  Position: Micrometer
  Speed: Pulse/second (PPS)
  N=0
  (SR1003, SR1002)=2,000 (pulses/revolution)
  (SR1005, SR1004)=100 (micrometers/revolution)
  P (I)=10,000 (micrometers)
  V (I)=10K (PPS)
  The number of pulses sent by the AH500 series motion control module is calculated below.
  Number of pulses it takes for the axis specified to move to the target position specified

$$= \frac{\text{P (I)}\,\mu\text{m}}{\text{B}/\text{A}} = \text{P (I)} \times \frac{\text{A}}{\text{B}} = 200{,}000\ \text{(pulses)}$$

7

- Bit 4 and bit 5 in SR1000+100*N: Output type

<table>
<tr><th>b5</th><th>b4</th><th>Output type (positive logic)</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>
</td><td>Counting up/down</td></tr>
<tr><td>0</td><td>1</td><td>
</td><td>Pulses+Directions</td></tr>
<tr><td>1</td><td>0</td><td rowspan="2">
</td><td>A/B-phase pulses</td></tr>
<tr><td>1</td><td>1</td><td>Four times the frequency of A/B-phase pulses</td></tr>
</table>

- Bit 8 in SR1000+100*N: Direction in which the axis specified returns home
- Bit 9 in SR1000+100*N: Mode of returning home
- Bit 10 in SR1000+100*N: Mode of triggering the return to home
  Please refer to section 7.6 for more information about bit 8, bit 9, and bit 10 in SR1000+100*N.
- Bit 11 in SR1000+100*N: Direction in which the motor used rotates
  (1) Bit 11=0: When the motor rotates clockwise, the value indicating the present position of the axis increases.
  (2) Bit 11=1: When the motor rotates clockwise, the value indicating the present position of the axis decreases.
- Bit 12 in SR1000+100*N: Relative/Absolute coordinates
  (1) Bit 12=0: Absolute coordinates
  (2) Bit 12=1: Relative coordinates
- Bit 13 in SR1000+100*N: Mode of triggering the calculation of the target position
  Please refer to section 7.1 fore more information. (The setting of bit 13 in SR1000+100*N is applicable to inserting single-speed motion and inserting two-speed motion.)
- Bit 14 in SR1000+100*N: Curve
  (1) Bit 14=0: Trapezoid curve
  (2) Bit 14=1: S curve

2. Number of pulses it takes for the motor of the axis specified to rotate once (A)

| (1+N)$^{th}$ axis | |
|---|---|
| **HW** | **LW** |
| SR1003+100*N | SR1002+100*N |

[Description]

- Owing to the fact that users can set an electronic gear ratio for a servo drive, the number of pulses it takes for a servo motor to rotate once is not necessarily equal to the number of pulses which will be generated after an encoder rotates once. The relation between the number of pulses it takes for a servo drive to rotate once and an electronic gear ratio is described below.
  Number of pulses it takes for a motor to rotate once (A) x Electronic gear ratio (CMX/CDV) = Number of pulses which will be generated after an encoder rotates once
  Example: If the number of pulses it takes for a motor to rotate once is 20,000, and the resolution of Delta ASDA-A2 series servo drive is 1,280,000, the gear ratio which should be set is 128/2.
- The unit used is determined by bit 0 and bit 1 in SR1000+100*N. If the unit selected is a mechanical unit or a compound unit, users need to set SR1002+100*N and SR1003+100*N. If

the unit selected is a motor unit, users do not need to set SR1002+100*N and SR1003+100*N.

3. Distance generated after the motor of the axis specified rotate once (B)

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1005+100*N | SR1004+100*N |

[Description]

- Three units are available. They are µm/revolution, mdeg/revolution, and $10^{-4}$ inches/revolution. The unit used is determined by bit 0 and bit 1 in SR1000+100*N. The value in (SR SR1005+100*N, SR1004+100*N) is in the range of 1 to 2,147,483,647.
- The unit used is determined by bit 0 and bit 1 in SR1000+100*N. If the unit selected is a mechanical unit or a compound unit, users need to set SR1004+100*N and SR1005+100*N. If the unit selected is a motor unit, users do not need to set SR1004+100*N and SR1005+100*N.

4. Maximum speed ($V_{MAX}$) at which the axis specified rotates

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1007+100*N | SR1006+100*N |

[Description]

- Users can set the maximum speed of motion. The value in (SR1007+100*N, SR1006+100*N) is in the range of 0 to 2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- The frequency of pulses generated by motion is in the range of 10 PPS to 1000K PPS. If the value in (SR1007+100*N, SR1006+100*N) is greater than 1000K, the frequency of pulses generated will be 1000K PPS. If the value in (SR1007+100*N, SR1006+100*N) is less than 10, the frequency of pulses generated will be 10 PPS.

5. Start-up speed ($V_{BIAS}$) at which the axis specified rotates

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1009+100*N | SR1008+100*N |

[Description]

- Users can set the start-up speed of motion. The value in (SR1009+100*N, SR1008+100*N) is in the range of 0 to 2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- The frequency of pulses generated by motion is in the range of 0 PPS to 1000K PPS. If the value in (SR1009+100*N, SR1008+100*N) is greater than 1000K, the frequency of pulses generated will be 1000K PPS. If the value in (SR1009+100*N, SR1008+100*N) is less than 0, the frequency of pulses generated will be 0 PPS.
- If a stepper motor system is used, the start-up speed that users set must be greater than the motor resonance frequency generated.

6. Time ($T_{ACC}$) it takes for the axis specified to accelerate

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1020+100*N |

[Description]

- Users can set the times it takes for the speed of the axis specified to increase from its start-up speed to its maximum speed. The value in SR1020+100*N is in the range of 0 to 32,767. A millisecond is a unit.
- If the value in SR1020+100*N is less than 10, it will be counted as 10. If the value in

SR1020+100*N is greater than 32,767, it will be counted as 32,767.

- If users want to have a complete S curve, the maximum speed which is set must be the same as the speed at which the axis specified operates.

7. Time ($T_{DEC}$) it takes for the axis specified to decelerate

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1021+100*N |

[Description]

- Users can set the times it takes for the speed of the axis specified to decrease from its maximum speed to its start-up speed. The value in SR1021+100*N is in the range of 0 to 32,767. A millisecond is a unit.
- If the value in SR1021+100*N is less than 10, it will be counted as 10. If the value in SR1021+100*N is greater than 32,767, it will be counted as 32,767.
- If users want to have a complete S curve, the maximum speed which is set must be the same as the speed at which the axis specified operates.

8. Present command position of the axis specified (Pulse)

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1033+100*N | SR1032+100*N |

[Description]

- The value in (SR1033+100*N, SR1032+100*N) is in the range of -2,147,483,648 to +2,147,483,647.
- The present command position of the axis specified is indicated by the number of pulses. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.) After the axis specified returns home, the value in (SR1019+100*N, SR1018+100*N) will be written into (SR1033+100*N, SR1032+100*N).

9. Present command speed of the axis specified (PPS)

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1035+100*N |

[Description]

- The value in SR1035+100*N is in the range of 0 to 2,147,483,647.

10. Present command position of the axis specified (Unit)

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1037+100*N | SR1036+100*N |

[Description]

- The value in (SR1037+100*N, SR1036+100*N) is in the range of -2,147,483,648 to +2,147,483,647.
- The unit used is determined by bit 0 and bit 1 in SR1000+100*N. After the axis specified returns home, the value in (SR1019+100*N, SR1018+100*N) will be written into (SR1037+100*N, SR1036+100*N).

11. Present command speed of the axis specified (Unit)

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1039+100*N | SR1038+100*N |

[Description]

- The value in (SR1039+100*N, SR1038+100*N) is in the range of 0 to 2,147,483,647.
- The unit used is determined by bit 0 and bit 1 in SR1000+100*N.

12. Operation command

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1030+100*N |

[Description]

| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1030+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | The execution of the Ox motion subroutine set starts. | A mode of inserting two-speed motion is activated. | A mode of two-speed motion is activated. | A mode of inserting single-speed motion is activated. | A mode of single-speed motion is activated. | - | A mode of triggering the return to home is activated. | A manual pulse generator is operated. | A mode of variable motion is activated. | The axis specified operates in JOG- mode. | The axis specified operates in JOG+ mode. | - | The motion of the axis specified is stopped by software. |

[Description]

- If a bit in SR1030+100*N is turned from OFF to ON when (SM1048+100*N) is ON, motion will be activated.
- When bit 0 in SR1030+100*N is turned from OFF to ON, motion decelerates and stops.
- Please refer to section 7.2~section 7.10 for more information.

13. Mode of operation

| (1+N)$^{th}$ axis | |
|---|---|
| **HW** | **LW** |
| - | SR1031+100*N |

[Description]

| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1031+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| Restoring the module to the factory settings | - | - | - | - | - | - | - | - | - | - | - | - | Mode of sending a CLR signal | - | - |

Bit 0~bit 15 in SR1031+100*N are described below.

- Bit 2: Mode of sending a CLR signal
  (1) Bit 2=0: After the axis returns home, the CLR output will send a 130 millisecond signal to the servo drive, and the present position of the servo drive which is stored in a register in the servo drive will be cleared.
  (2) Bit 2=1: The CLR output functions as a general output.
- Bit 15: Restoring the module to the factory settings
  (1) Bit 15=1: The values of parameters are restored to factory settings.

## 7.2 Introduction of Uniaxial Motion

1. There are eight modes of motion.

| | |
|---|---|
| 1. Returning home | 5. Two-speed motion |
| 2. JOG motion | 6. Inserting two-speed motion |
| 3. Single-speed motion | 7. Variable motion |
| 4. Inserting single-speed motion | 8. Manual pulse generator mode |

2. If more than one mode of motion is activated, they will be executed in particular order.

| | |
|---|---|
| 1. Stopping the motion of the axis specified by software. | 6. Variable motion |
| 2. Returning home | 7. Single-speed motion |
| 3. Positive JOG motion | 8. Inserting single-speed motion |
| 4. Negative JOG motion | 9. Two-speed motion |
| 5. Manual pulse generator mode | 10. Inserting two-speed motion |

If a mode of motion is activated when another mode of motion is executed, the AH500 series motion control module will continue executing the original mode.

3. Uniaxial motion is controlled by SR1030+100*N. After the parameters related to motion are set, the motion can be started.

| (1+N)th axis | | Operation command |
|---|---|---|
| **HW** | **LW** | |
| - | SR1030+100*N | |

[Description]

| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1030+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | The execution of the Ox motion subroutine set starts. | A mode of inserting two-speed motion is activated. | A mode of two-speed motion is activated. | A mode of inserting single-speed motion is activated. | A mode of single-speed motion is activated. | - | A mode of triggering the return to home is activated. | A manual pulse generator is operated. | A mode of variable motion is activated. | The axis specified operates in JOG- mode. | The axis specified operates in JOG+ mode. | - | The motion of the axis specified is stopped by software. |

# 7.3 Introduction of JOG Motion

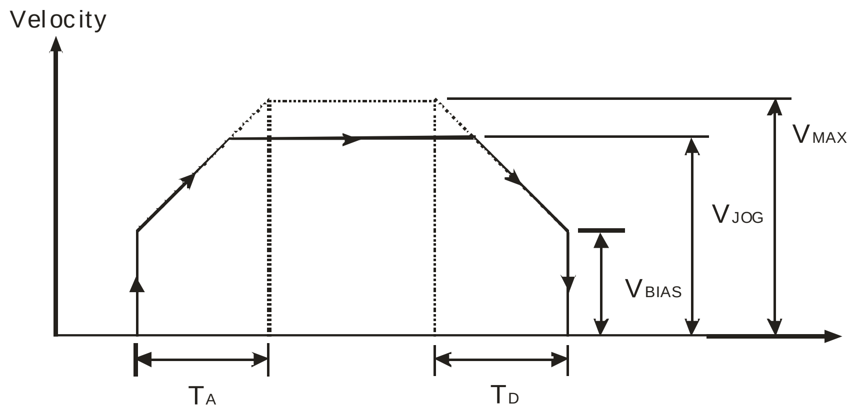
## 7.3.1 Related Special Data Registers

1. JOG speed ($V_{JOG}$) at which the axis specified rotates

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1011+100*N | SR1010+100*N |

[Description]

- Users can set the JOG speed ($V_{JOG}$) at which the axis specified rotates. The value in (SR1011+100*N, SR1010+100*N) is in the range of 0 to 2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- The frequency of pulses generated by motion is in the range of 10 PPS to 1000K PPS. If the value in (SR1009+100*N, SR1008+100*N) is greater than 1000K, the frequency of pulses generated will be 1000K PPS. If the value in (SR1009+100*N, SR1008+100*N) is less than 10, the frequency of pulses generated will be 10 PPS.
- $V_{MAX}$>$V_{JOG}$>$V_{BIAS}$
  If the $V_{JOG}$ set is greater than the $V_{MAX}$ set, the actual $V_{JOG}$ will be equal to the $V_{MAX}$.
  If the $V_{JOG}$ set is less than the $V_{BIAS}$ set, an error will occur.
  When an axis operates, users can modify the JOG speed of the axis. If the value in (SR1011+100*N, SR1010+100*N) is 0, the JOG motion of the axis specified will be stopped, and will needs to be started again. If JOG motion is started when the value in (SR1011+100*N, SR1010+100*N) is 0, an error will occur.




2. Operation command

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1030+100*N |

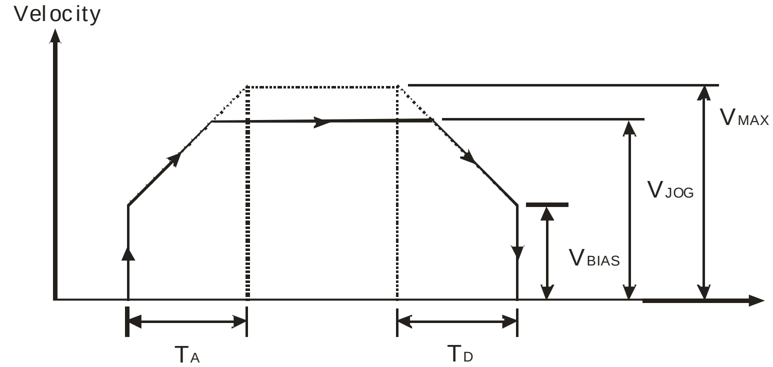
[Description]

| **Special data register** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1030+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | The execution of the Ox motion subroutine set starts. | A mode of inserting two-speed motion is activated. | A mode of two-speed motion is activated. | A mode of inserting single-speed motion is activated. | A mode of single-speed motion is activated. | - | A mode of triggering the return to home is activated. | A manual pulse generator is operated. | A mode of variable motion is activated. | The axis specified operates in JOG- mode. | The axis specified operates in JOG+ mode. | - | The motion of the axis specified is stopped by software. |

- Bit 2=1: Positive JOG motion
- Bit3=1: Negative JOG motion

### 7.3.2 Operation

- When bit 2 in SR1030+100*N is ON, clockwise pulses are generated at the JOG speed set.
- When bit 3 in SR1030+100*N is ON, counterclockwise pulses are generated at the JOG speed set.
- After a JOG mode is activated, the AH500 series motion control module used will execute JOG motion. The speed of JOG motion can be modified when the JOG motion is executed. If the value in (SR1011+100*N, SR1010+100*N) is 0, the JOG motion of the axis specified will be stopped, and will needs to be started again. If JOG motion is started when the value in (SR1011+100*N, SR1010+100*N) is 0, an error will occur.




# 7.4 Introduction of Variable Motion

## 7.4.1 Related Special Data Registers

1. Speed at which the axis specified rotates (V (I))

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1025+100*N | SR1024+100*N |

[Description]

- The value in (SR1025+100*N, SR1024+100*N) is in the range of 0 to +2,147,483,647. The unit used is determined by bit 0 and bit 1 in SR1000+100*N.
- The frequency of pulses generated by motion is in the range of 10 PPS to 1000K PPS. If the value in (SR1025+100*N, SR1024+100*N) is greater than 1000K, the frequency of pulses generated will be 1000K PPS. If the value in (SR1025+100*N, SR1024+100*N) is less than 10, the frequency of pulses generated will be 10 PPS.
- $V_{MAX} > V(I) > V_{BIAS}$
- When bit 4 in SR1030+100*N is ON, the speed at which the axis specified rotates (V (I)) can be changed.

2. Operation command

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1030+100*N |

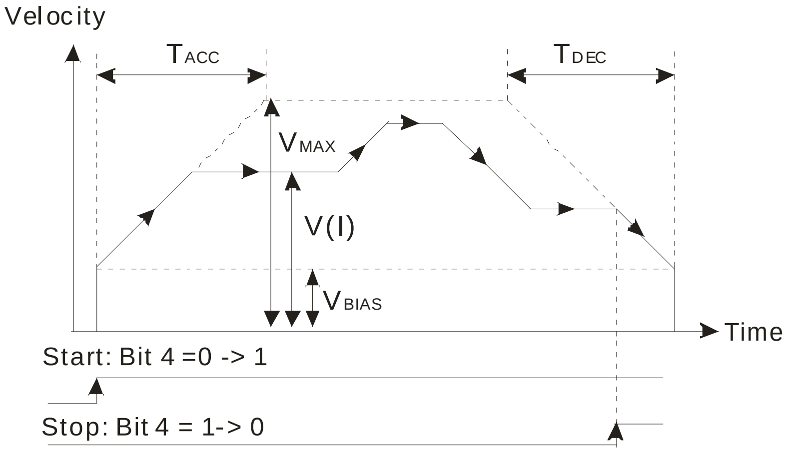
[Description]

| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1030+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | The execution of the Ox motion subroutine set starts. | A mode of inserting two-speed motion is activated. | A mode of two-speed motion is activated. | A mode of inserting single-speed motion is activated. | A mode of single-speed motion is activated. | - | A mode of triggering the return to home is activated. | A manual pulse generator is operated. | A mode of variable motion is activated. | The axis specified operates in JOG- mode. | The axis specified operates in JOG+ mode. | - | The motion of the axis specified is stopped by software. |

- If users want to activate a mode of variable motion, they have to set bit 4 in SR1030+100*N to 1.

### 7.4.2 Operation

- After bit 4 in SR1030+100*N is set to 1, the AH500 series motion control module will execute variable motion. AH10PM-5A, AH15PM-5A, and AH05PM-5A send pulses by pulse generators. AH20MC-5A sends pulses to a servo drive by means of the DMCNET.
- After a mode of variable motion is activated, the $V_{BIAS}$ of the axis specified will increase to its V (I). When the axis operates, users can change its V (I) at will. The AH500 series motion control module accelerates or decelerates according to the V (I) set.
- Diagram




## 7.5 Introduction of a Manual Pulse Generator Mode

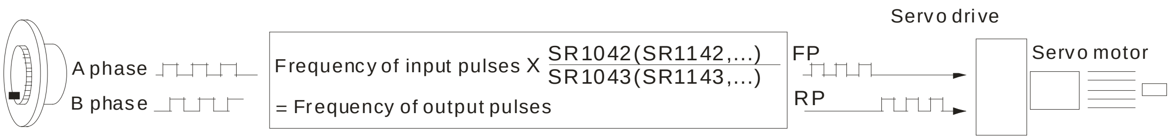
### 7.5.1 Related Special Data Registers

1. Electronic gear ratio of the axis specified

| $(1+N)^{th}$ axis | | Electronic gear ratio |
|---|---|---|
| **HW** | **LW** | |
| - | SR1042+100*N | Electronic gear ratio (Numerator) |
| - | SR1043+100*N | Electronic gear ratio (Denominator) |

[Description]

- If bit 5 in SR1030+100*N is set to ON, a manual pulse generator mode will be activated.
- A manual pulse generator generates A/B-phase pulses that are sent to an input terminal. The relation between the position of the axis specified and the input pulses generated by the manual pulses used is shown below.




- The speed output is determined by the frequency of input pulses generated by a manual pulse generator and an electronic gear ratio.

2. Frequency of pulses generated by the manual pulse generator for the axis specified

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1045+100*N | SR1044+100*N |

[Description]

The value in (SR1045+100*N, SR1044+100*N) indicates the frequency of pulses generated by the manual pulse generator for the axis specified. It does not vary with the values in SR1042+100*N and SR1043+100*N.

3. Number of pulses generated by the manual pulse generator for the axis specified

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1047+100*N | SR1046+100*N |

[Description]

- The value in (SR1047+100*N, SR1046+100*N) indicates the number of pulses generated by the manual pulse generator for the axis specified. If the pulses generated by the manual pulse generator for the axis specified are clockwise pulses, the value in (SR1047+100*N, SR1046+100*N) will increase. If the pulses generated by the manual pulse generator for the axis specified are counterclockwise pulses, the value in (SR1047+100*N, SR1046+100*N) will decrease.
- The value in (SR1047+100*N, SR1046+100*N) does not vary with the values in SR1042+100*N and SR1043+100*N.
- 

4. Response speed of the manual pulse generator for the axis specified

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1048+100*N |

[Description]

- If the response speed set is high, the pulses output happen almost at the same time as the pulse input by the manual pulse generator used.
- If the response speed set is low, the pulses output follows the pulses input by the manual pulse generator used.

| Setting value | Response speed |
|---|---|
| ≧5 | 4 ms (Initial value) |
| 4 | 32 ms |
| 3 | 108 ms |
| 2 | 256 ms |
| 1 or 0 | 500 ms |

7

5. Operation command

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1030+100*N |

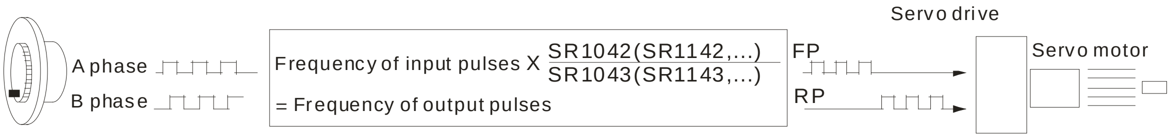
[Description]

| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1030+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | The execution of the Ox motion subroutine set starts. | A mode of inserting two-speed motion is activated. | A mode of two-speed motion is activated. | A mode of inserting single-speed motion is activated. | A mode of single-speed motion is activated. | - | A mode of triggering the return to home is activated. | A manual pulse generator is operated. | A mode of variable motion is activated. | The axis specified operates in JOG- mode. | The axis specified operates in JOG+ mode. | - | The motion of the axis specified is stopped by software. |

- If users want to activate a manual pulse generator mode, they have to set bit 5 in SR1030+100*N to 1.

### 7.5.2 Operation

- If bit 5 in SR1030+100*N is set to 1, a manual pulse generator mode will be activated.
- The value in (SR1047+100*N, SR1046+100*N) indicates the number of pulses generated by the manual pulse generator for the axis specified.




## 7.6 Introduction of a Mode of Triggering the Return to Home

### 7.6.1 Related Special Data Registers

1. Speed ($V_{RT}$) at which the axis specified returns home

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1013+100*N | SR1012+100*N |

[Description]

- Users can set the speed at which the axis specified returns home. The value in (SR1013+100*N, SR1012+100*N) is in the range of 1 to 2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- The frequency of pulses generated by motion is in the range of 10 PPS to 1000K PPS. If the value in (SR1013+100*N, SR1012+100*N) is greater than 1000K, the frequency of pulses generated will be 1000K PPS. If the value in (SR1013+100*N, SR1012+100*N) is less than 10,

the frequency of pulses generated will be 10 PPS.

- $V_{MAX}>V_{RT}>V_{BIAS}$
- When an axis returns home, the speed at which the axis returns home can not be changed.

2. Speed ($V_{CR}$) to which the speed of the axis specified decreases when the axis returns home

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1015+100*N | SR1014+100*N |

[Description]

- The value in (SR1015+100*N, SR1014+100*N) is in the range of 1 to 2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- The frequency of pulses generated by motion is in the range of 10 PPS to 1000K PPS. If the value in (SR1015+100*N, SR1014+100*N) is greater than 1000K, the frequency of pulses generated will be 1000K PPS. If the value in (SR1015+100*N, SR1014+100*N) is less than 10, the frequency of pulses generated will be 10 PPS.
- When motion of returning home is executed, the speed of the motor used is the $V_{RT}$ set. When there is a transition in DOG's signal from low to high or from high to low, the speed of the motor used decreases to the $V_{CR}$ set.
- In order for the axis specified to returns home precisely, it is suggested that the $V_{CR}$ set should be a low speed.
- When the motion of returning home is executed, the $\mathbf{V_{RT}}$ set can not be changed.

3. Number of PG0 pulses for the axis specified

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1016+100*N |

[Description]

- The value in SR1016+100*N is in the range of 0 to 65,535. It is a positive value.
- Please refer to the descriptions of bit 9 and bit10 in SR1000+100*N for more information about decelerating and stopping the motor used.

4. Supplementary pulses for the axis specified

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1017+100*N |

[Description]

- The value in SR1017+100*N is in the range of -32,768 to 32,767. If the value in SR1017+100*N is a positive value, the axis specified will move in the direction in which it returns home. If the value in SR1017+100*N is a negative value, the axis specified will move in the direction which is opposite to the direction in which it returns home.
- Please refer to the descriptions of bit 9 and bit10 in SR1000+100*N for more information about decelerating and stopping the motor used.

5. Home position of the axis specified

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1019+100*N | SR1018+100*N |

[Description]

- The value in (SR1019+100*N, SR1018+100*N) is in the range of 0 to ±999,999. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- After the axis specified returns home, the value in (SR1019+100*N, SR1018+100*N) will be

written into (SR1033+100*N, SR1032+100*N).

6. Mode of operation

| $(1+N)^{th}$ axis | |
|---|---|
| **HW** | **LW** |
| - | SR1031+100*N |

[Description]

- Bit 2 in SR1031+100*N: Mode of sending a CLR signal
  - Bit 2=0: After the axis returns home, the CLR output will send a 130 millisecond signal to the servo drive, and the present position of the servo drive which is stored in a register in the servo drive will be cleared.
  - Bit 2=1: The CLR output functions as a general output.

7. Setting the parameters of the axis specified

| $(1+N)^{th}$ axis | |
|---|---|
| **HW** | **LW** |
| - | SR1000+100*N |

[Description]

Motion of retuning home:

- Bit 8 in SR1000+100*N: Direction in which the axis specified returns home
  - Bit 8=0: The value indicating the present command position of an axis specified decreases, and the axis returns home in the negative direction.
  - Bit 8=1: The value indicating the present command position of an axis specified increases, and the axis returns home in the positive direction.
- Bit 9 in SR1000+100*N: Mode of returning home
  - Bit 9=0: Normal mode
    After DOG's signal is generated, the motor used will rotate for a specific number of PG0 pulses, then rotate for a specific number of supplementary pulses, and finally stop.
  - Bit 9=1: Overwrite mode
    After DOG's signal is generated, the motor used will rotate for a number of PG0 pulses or rotate for a number of supplementary pulses, and then stop.
- Bit 10 in SR1000+100*N: Mode of triggering the return to home
  - Bit 10=0: The return to home is triggered by a transition in DOG's signal from high to low.
  - Bit 10=1: The return to home is triggered by a transition in DOG's signal from low to high.

8. Operation command

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1030+100*N |

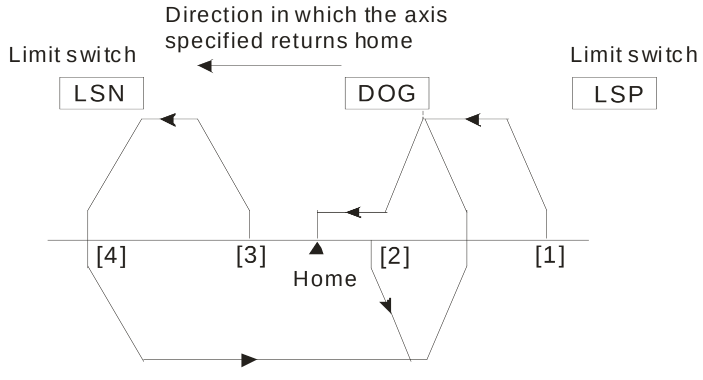
[Description]

| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1030+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | The execution of the Ox motion subroutine set starts. | A mode of inserting two-speed motion is activated. | A mode of two-speed motion is activated. | A mode of inserting single-speed motion is activated. | A mode of single-speed motion is activated. | - | A mode of triggering the return to home is activated. | A manual pulse generator is operated. | A mode of variable motion is activated. | The axis specified operates in JOG- mode. | The axis specified operates in JOG+ mode. | - | The motion of the axis specified is stopped by software. |

- If users want to activate a mode of triggering the return to home, they have to set bit 6 in SR1030+100*N to 1.

## 7.6.2 Operation

- When bit 6 in SR1030+100*N is turned from OFF to ON, a mode of triggering the return to home is activated. The mode of triggering the return to home varies with the present command position of the axis specified. There are two situations.




Position (1): Position [1] is at the right side of the home and DOG, and DOG is OFF.
Position (2): Position [2] is at the right side of the home, and DOG is ON.
Position (3): Not supported
Position (4): Not supported

- Setting parameters
  There are four modes of stopping the rotation of a motor.
  - Bit [9:10] in SR1000+100*N is 00.⇨The mode of returning home is a normal mode, and the return to home is triggered by a transition in DOG's signal from high to low.

Steps:

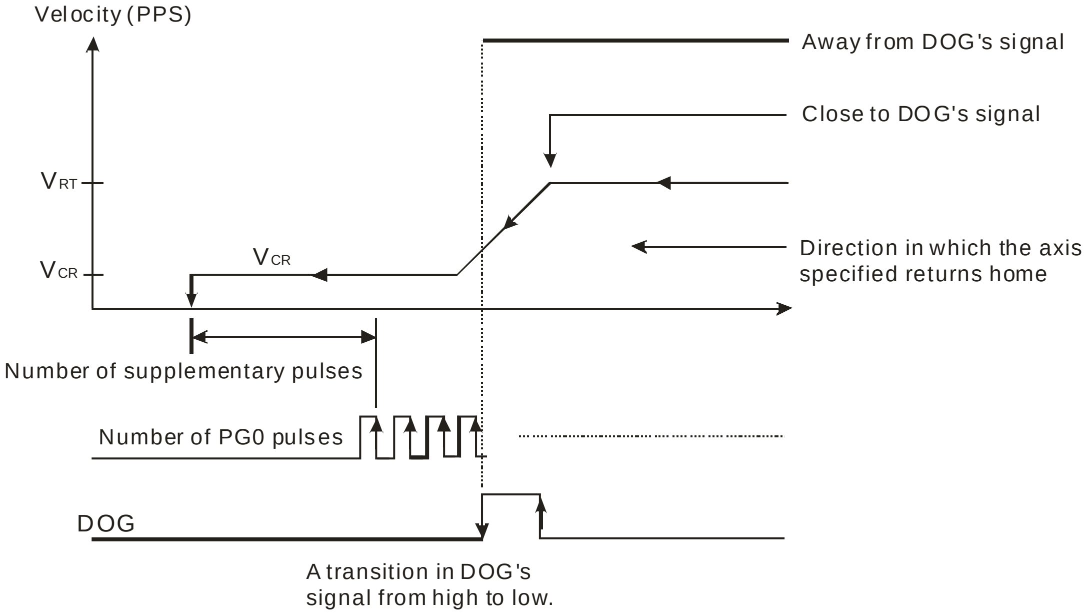
(1) The motor used rotates at the speed $\mathbf{V_{RT}}$.

(2) When DOG's signal is generated, the speed of the motor begins to decrease to the speed $V_{CR}$.

(3) After DOG's signal goes from high to low, the motor will rotate for a specific number of PG0 pulses, and then rotate for a specific number of supplementary pulses, and finally stop.

If the number of PG0 pulses or the number of supplementary pulses is not large, the speed of the motor used will decrease to the speed $V_{CR}$ after DOG's signal is generated. After DOG's signal goes from high to low, the motor will rotate for a specific number of PG0 pulses, and then rotate for a specific number of supplementary pulses, and finally stop whether the its speed is $V_{CR}$.

If the number of PG0 pulses is 0, and the number of supplementary pulses is 0, the motor used will stop after DOG's signal is generated and there is a transition in DOG's signal from high to low.




■ Bit [9:10] in SR1000+100*N is 01.⇨The mode of returning home is a normal mode, and the return to home is not triggered by a transition in DOG's signal from high to low.

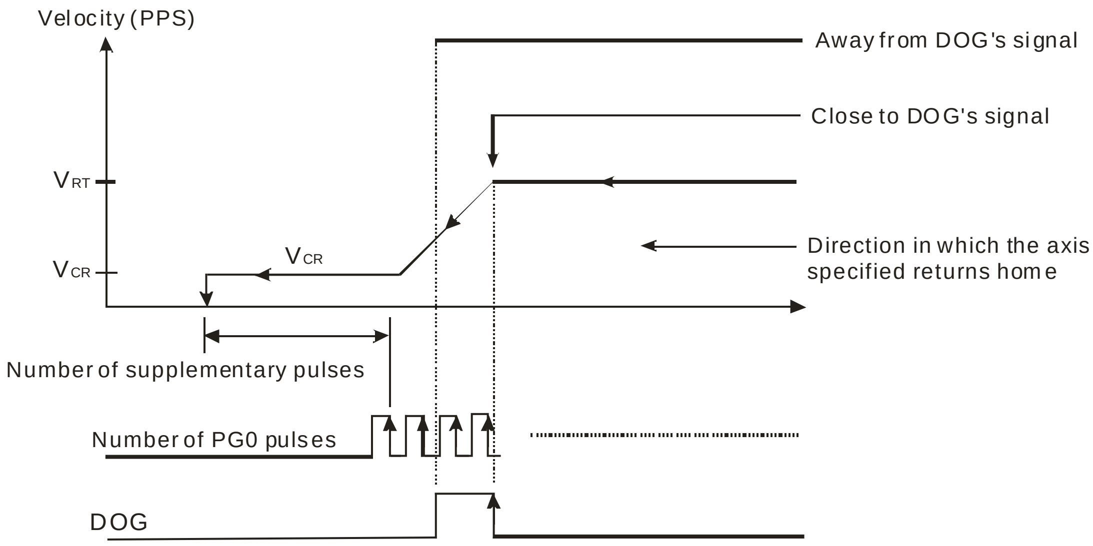
Steps:

(1) The motor used rotates at the speed $V_{RT}$.

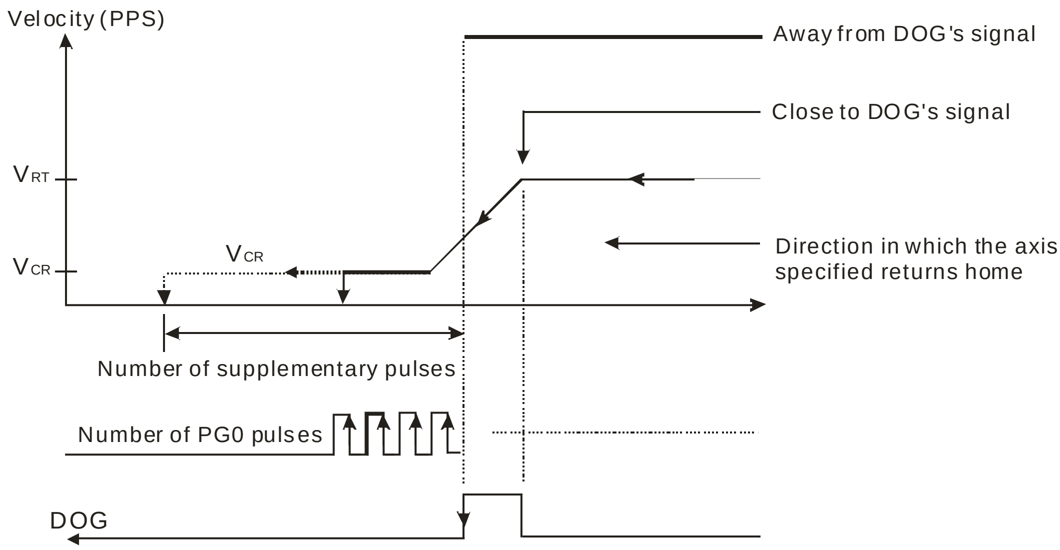
(2) When DOG's signal is generated, the speed of the motor begins to decrease to the speed $V_{CR}$. After the motor rotates for a specific number of PG0 pulses, and rotate for a specific number of supplementary pulses, it will stop.

If the number of PG0 pulses or the number of supplementary pulses is not large, the speed of the motor used will decrease to the speed $\mathbf{V_{CR}}$ after DOG's signal is generated. After the motor rotates for a specific number of PG0 pulses, and rotates for a specific number of supplementary pulses, it will stop whether its speed is $\mathbf{V_{CR}}$.

If the number of PG0 pulses is 0, and the number of supplementary pulses is 0, the motor used will stop after DOG's signal is generated.




- Bit [9:10] in SR1000+100*N is 10.⇨ The mode of returning home is an overwrite mode, and the return to home is triggered by a transition in DOG's signal from high to low.
  Steps:
  (1) The motor used rotates at the speed $V_{RT}$.
  (2) When DOG's signal is generated, the speed of the motor begins to decrease to the speed $V_{CR}$.
  (3) After DOG's signal goes from high to low, the motor will rotate for a specific number of PG0 pulses, or rotate for a specific number of supplementary pulses, and then stop.
  If the number of PG0 pulses or the number of supplementary pulses is not large, the speed of the motor used will decrease to the speed $V_{CR}$ after DOG's signal is generated. After DOG's signal goes from high to low, the motor will rotate for a specific number of PG0 pulses, or rotate for a specific number of supplementary pulses, and then stop whether the its speed is $V_{CR}$.
  If the number of PG0 pulses is 0, and the number of supplementary pulses is 0, the motor used will stop after DOG's signal is generated and there is a transition in DOG's signal from high to low.




- Bit [9:10] in SR1000+100*N is 11.⇨ The mode of returning home is an overwrite mode, and the return to home is not triggered by a transition in DOG's signal from high to low.

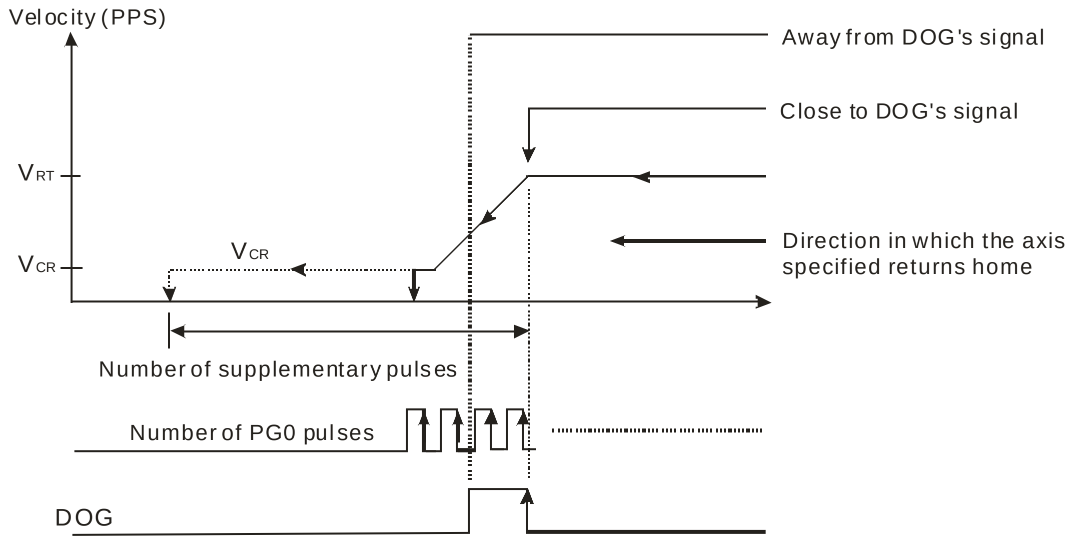
Steps:

(1) The motor used rotates at the speed $V_{RT}$.

(2) When DOG's signal is generated, the speed of the motor begins to decrease to the speed $V_{CR}$. After the motor rotates for a specific number of PG0 pulses, or rotate for a specific number of supplementary pulses, it will stop.

If the number of PG0 pulses or the number of supplementary pulses is not large, the speed of the motor used will decrease to the speed $V_{CR}$ after DOG's signal is generated. After the motor rotates for a specific number of PG0 pulses, or rotates for a specific number of supplementary pulses, it will stop whether its speed is $V_{CR}$.

If the number of PG0 pulses is 0, and the number of supplementary pulses is 0, the motor used will stop after DOG's signal is generated.




# 7.7 Introduction of Single-speed motion

## 7.7.1 Related Special Data Registers

1. Target position of the axis specified (P (I))

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1023+100*N | SR1022+100*N |

[Description]

- The value in (SR1023+100*N, SR1022+100*N) is in the range of -2,147,483,648 to +2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- Target position (P (I))
  - Absolute coordinates: Bit 12 in SR1000+100*N is 0.

    The target position of the axis specified indicates a distance from 0. If the target position of an axis is greater than its present command position, the motor used will rotate clockwise. If the target position of an axis is less than its present command position, the motor used will rotate counterclockwise.
  - Relative coordinates: Bit 12 in SR1000+100*N is 1

    The target position of an axis indicates a distance from its present command position. If the target position specified is a positive value, the motor used will rotate clockwise. If the target position specified is a negative value, the motor used will rotate counterclockwise.

2. Speed at which the axis specified rotates (V (I))

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1025+100*N | SR1024+100*N |

[Description]

- The value in (SR1025+100*N, SR1024+100*N) is in the range of 0 to +2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- The frequency of pulses generated by motion is in the range of 10 PPS to 1000K PPS. If the value in (SR1025+100*N, SR1024+100*N) is greater than 1000K, the frequency of pulses generated will be 1000K PPS. If the value in (SR1025+100*N, SR1024+100*N) is less than 10, the frequency of pulses generated will be 10 PPS.
- $V_{MAX} > V\ (I) > V_{BIAS}$
- When an axis operates, users can modify its V (I). If the value in (SR1025+100*N, SR1024+100*N) is 0, the motion of the axis specified will be stopped, and will need to be started again.

3. Operation command

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1030+100*N |

[Description]

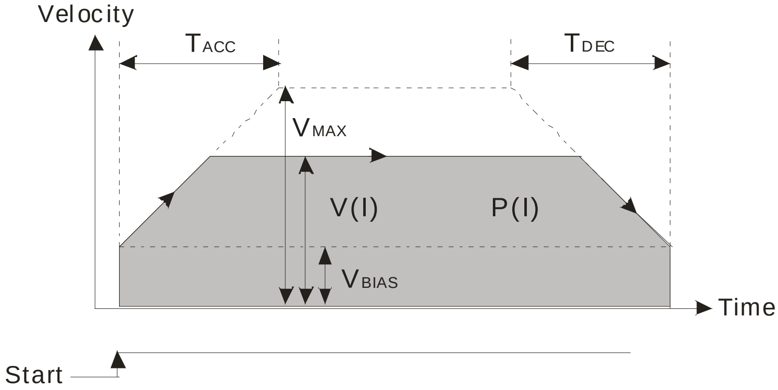
| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1030+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | The execution of the Ox motion subroutine set starts. | A mode of inserting two-speed motion is activated. | A mode of two-speed motion is activated. | A mode of inserting single-speed motion is activated. | A mode of single-speed motion is activated. | - | A mode of triggering the return to home is activated. | A manual pulse generator is operated. | A mode of variable motion is activated. | The axis specified operates in JOG- mode. | The axis specified operates in JOG+ mode. | - | The motion of the axis specified is stopped by software. |

- If users want to activate a mode of single-speed motion, they have to set bit 8 in SR1030+100*N to 1.

### 7.7.2 Operation

- After bit 8 in SR1030+100*N is set to 1, a mode of single-speed motion will be activated. The target position of the single-speed motion and the speed of the single-speed motion depend on the P (I) and the V (I) which are set by users.
- If relative single-speed motion is activated, the sign bit of the P (I) set by users will determine the direction of the relative single-speed motion. After relative single-speed motion is activated, the speed of the relative single-speed motion will increase from the $V_{BIAS}$ set to the V (I) set. The speed of the relative single-speed motion will not decrease from the V (I) set to the $V_{BIAS}$ set until the number of pulses output is near the P (I) set.

- Absolute single-speed motion: If the target position of the axis specified is greater than its present command position, the motor used will rotate clockwise. If the target position of the axis specified is less than its present command position, the motor used will rotate counterclockwise. After absolute single-speed motion is activated, the speed of the absolute single-speed motion will increase from the $V_{BIAS}$ set to the V (I) set. The speed of the absolute single-speed motion will not decrease from the V (I) set to the $V_{BIAS}$ set until the present command position of the axis specified is near the target position (P (I)) set.




# 7.8 Introduction of Inserting Single-speed Motion

## 7.8.1 Related Special Data Registers

1. Target position of the axis specified (P (I))

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1023+100*N | SR1022+100*N |

[Description]

- The value in (SR1023+100*N, SR1022+100*N) is in the range of -2,147,483,648 to +2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- Target position (P (I))
  After motion is triggered by a transition in DOG's signal from low to high or from high to low, the axis specified will move from its present command position to the target position (P (I)) set. If the relative target position specified is a positive value, the motor used will rotate clockwise. If the relative target position specified is a negative value, the motor used will rotate counterclockwise.

2. Speed at which the axis specified rotates (V (I))

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1025+100*N | SR1024+100*N |

[Description]

- When an axis operates, users can not change its V (I) at will.
- Please refer to section 7.7 for more information.

3. Setting parameters

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1000+100*N |

[Description]

Please refer to the descriptions of common special data registers in section 7.1 for more information.

- Mode of triggering the calculation of the target position
  - Bit 13=0: The calculation of the target position of the axis specified is triggered by a transition in DOG's signal from low to high.
  - Bit 13=1: The calculation of the target position of the axis specified is triggered by a transition in DOG's signal from high to low.

4. Operation command

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1030+100*N |

[Description]

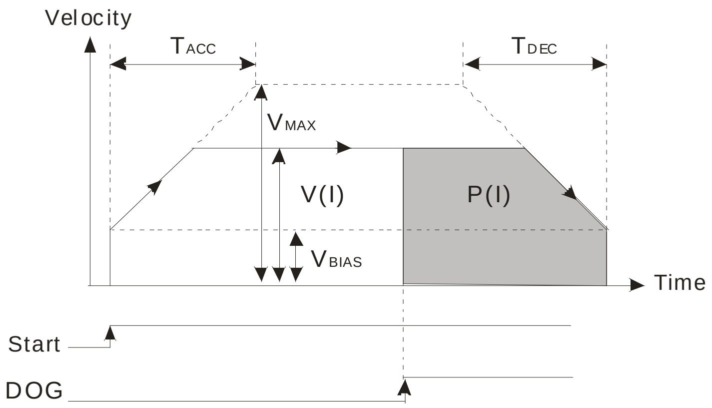
| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1030+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | The execution of the Ox motion subroutine set starts. | A mode of inserting two-speed motion is activated. | A mode of two-speed motion is activated. | A mode of inserting single-speed motion is activated. | A mode of single-speed motion is activated. | - | A mode of triggering the return to home is activated. | A manual pulse generator is operated. | A mode of variable motion is activated. | The axis specified operates in JOG- mode. | The axis specified operates in JOG+ mode. | - | The motion of the axis specified is stopped by software. |

- If users want to activate a mode of inserting single-speed motion, they have to set bit 9 in SR1030+100*N to 1.

## 7.8.2 Operation

- After bit 9 in SR1030+100*N is set to 1, a mode of inserting single-speed motion will be activated. The AH500 series motion control modules used sends pulses by pulse generators. After DOG's signal goes from low to high or from high to low, the axis specified will move to the relative target position indicated by the P (I) set. If the value in (SR1023+100*N, SR1022+100*N) is 0, an error will occur.
- Direction: If the target position (P (I)) specified is a positive value, the motor used will rotate clockwise. If the target position (P (I)) specified is a negative value, the motor used will rotate counterclockwise.
- Steps:
  - The speed of motion will increase from the $V_{BIAS}$ set to the V (I) set.
  - After DOG's signal goes from low to high or from high to low, the AH500 series motion control modules used will continue sending pulses. The speed of the motion will not

decrease from the V (I) set to the $V_{BIAS}$ set until the number of pulses output is near the P (I) set.




# 7.9 Introduction of Two-speed Motion

## 7.9.1 Related Special Data Registers

1. Target position of the axis specified (P (I))

| $(1+N)^{th}$ axis | |
|---|---|
| **HW** | **LW** |
| SR1023+100*N | SR1022+100*N |

[Description]

Please refer to section 7.7 for more information.

2. Speed at which the axis specified rotates (V (I))

| $(1+N)^{th}$ axis | |
|---|---|
| **HW** | **LW** |
| SR1025+100*N | SR1024+100*N |

[Description]

- When an axis operates, users can not change its V (I) at will.
- Please refer to section 7.7 for more information.

3. Target position of the axis specified (P (II))

| $(1+N)^{th}$ axis | |
|---|---|
| **HW** | **LW** |
| SR1027+100*N | SR1026+100*N |

[Description]

- The value in (SR1027+100*N, SR1026+100*N) is in the range of -2,147,483,648 to +2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- Target position (P (II))
  - Absolute coordinates: Bit 12 in SR1000+100*N is 0.
    The direction in which the axis specified moves from its P (I) to its P (II) must be the same as the direction in which it moves from its present command position to it P (I).
  - Relative coordinates: Bit 12 in SR1000+100*N is 1
    If the P (I) specified is a positive value, the P (II) is also a positive value. If the P (I) specified is a negative value, the P (II) is also a negative value. The direction in which the axis specified moves from its P (I) to its P (II) must be the same as the direction in which it moves from its present command position to it P (I).

4. Speed at which the axis specified rotates(V (II))

| $(1+N)^{th}$ axis | |
|---|---|
| **HW** | **LW** |
| SR1029+100*N | SR1028+100*N |

[Description]

- The value in (SR1029+100*N, SR1028+100*N) is in the range of 0 to 2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- The frequency of pulses generated by motion is in the range of 10 PPS to 1000K PPS. If the value in (SR1029+100*N, SR1028+100*N) is greater than 1000K, the frequency of pulses generated will be 1000K PPS. If the value in (SR1029+100*N, SR1028+100*N) is less than 10, the frequency of pulses generated will be 10 PPS.
- $V_{MAX}$ > V (II) > $V_{BIAS}$
- When an axis operates, users can not modify its V (II).

5. Operation command

| $(1+N)^{th}$ axis | |
|---|---|
| **HW** | **LW** |
| - | SR1030+100*N |

[Description]

| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1030+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | The execution of the Ox motion subroutine set starts. | A mode of inserting two-speed motion is activated. | A mode of two-speed motion is activated. | A mode of inserting single-speed motion is activated. | A mode of single-speed motion is activated. | - | A mode of triggering the return to home is activated. | A manual pulse generator is operated. | A mode of variable motion is activated. | The axis specified operates in JOG- mode. | The axis specified operates in JOG+ mode. | - | The motion of the axis specified is stopped by software. |

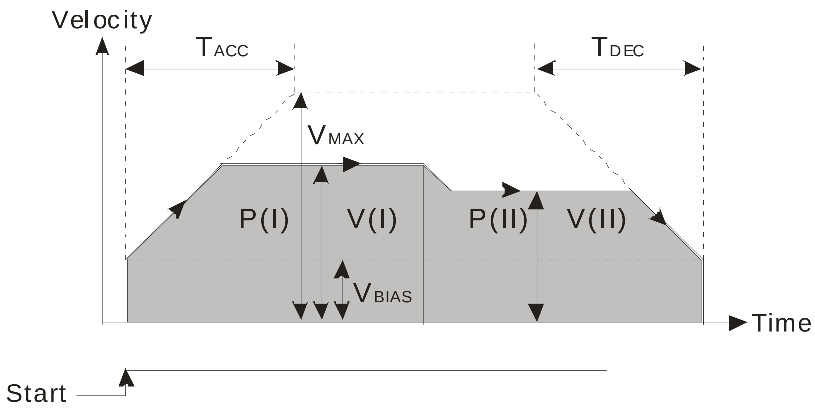
- If users want to activate a mode of two-speed motion, they have to set bit 10 in SR1030+100*N to 1.

### 7.9.2 Operation

- After bit 10 in SR1030+100*N is set to 1, a mode of two-speed motion will be activated. The axis specified moves at the V (I) set. After it moves to the P (I) set, it will move to the P (II) set at the V (II) set.
- Relative coordinates: The sign bit of the P (I) set by users determines the direction of motion. If the P (I) specified is a positive value, the motor used will rotate clockwise. If the P (I) specified is a negative value, the motor used will rotate counterclockwise. If the P (I) specified is a positive value, the P (II) is also a positive value. If the P (I) specified is a negative value, the P (II) is also a negative value. After motion is started, the speed of the motion will increase from the $V_{BIAS}$ set to the V (I) set. The speed of the motion will not increase/decrease from the V (I) set to the V (II) set until the number of pulses output is near the P (I) set. The speed of the motion will not

decrease from the V (II) to the $V_{BIAS}$ set until the number of pulses output is near the P (II) set.

- Absolute coordinates: If the target position (P (I)) of an axis is greater than its present command position, the motor used will rotate clockwise. If the target position (P (I)) of an axis is less than its present command position, the motor used will rotate counterclockwise. The P (I) set must be between the present command position of the axis specified and the P (II) set. After motion is started, the speed of the motion will increase from the $V_{BIAS}$ set to the V (I) set. The speed of the motion will not increase/decrease from the V (I) set to the V (II) set until the present command position of the axis specified is near the P (I) set. The speed of the motion will not decrease from the V (II) to the $V_{BIAS}$ set until the present command position of the axis specified is near the P (II) set.




# 7.10 Introduction of Inserting Two-speed Motion

## 7.10.1 Related Special Data Registers

1. Speed at which the axis specified rotates (V (I))

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1025+100*N | SR1024+100*N |

[Description]

- When an axis operates, users can not change its V (I) at will.
- Please refer to section 7.7 for more information.

7

2. Speed at which the axis specified rotates(V (II))

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1029+100*N | SR1028+100*N |

[Description]

Please refer to section 7.9 for more information.

3. Target position of the axis specified (P (II))

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| SR1027+100*N | SR1026+100*N |

[Description]

- The value in (SR1027+100*N, SR1026+100*N) is in the range of -2,147,483,648 to +2,147,483,647. (The unit used is determined by bit 0 and bit 1 in SR1000+100*N.)
- Target position (P (II))
  After motion is triggered by a transition in DOG's signal from low to high or from high to low, the

axis specified will move from its present command position to the target position (P (II)) set. If the relative target position specified is a positive value, the motor used will rotate clockwise. If the relative target position specified is a negative value, the motor used will rotate counterclockwise.

4. Operation command

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1030+100*N |

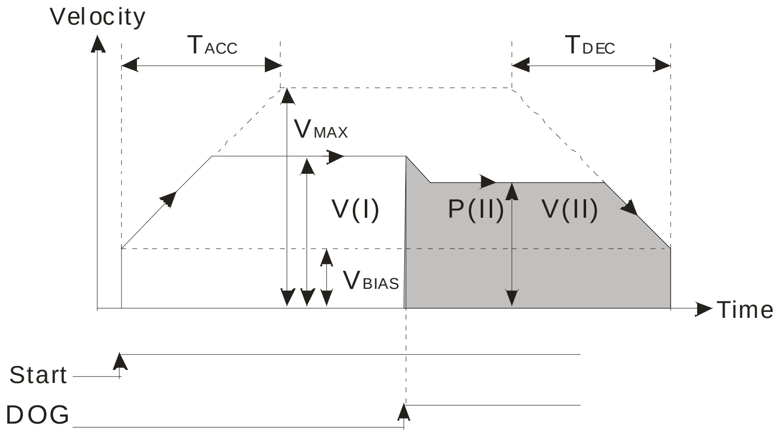
[Description]

| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1030+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | The execution of the Ox motion subroutine set starts. | A mode of inserting two-speed motion is activated. | A mode of two-speed motion is activated. | A mode of inserting single-speed motion is activated. | A mode of single-speed motion is activated. | - | A mode of triggering the return to home is activated. | A manual pulse generator is operated. | A mode of variable motion is activated. | The axis specified operates in JOG- mode. | The axis specified operates in JOG+ mode. | - | The motion of the axis specified is stopped by software. |

- If users want to activate a mode of inserting two-speed motion, they have to set bit 11 in SR1030+100*N to 1.

## 7.10.2 Operation

- After bit 11 in SR1030+100*N is set to 1, a mode of inserting two-speed motion will be activated. The axis specified moves at the V (I) set. After DOG's signal goes from low to high or from high to low, the axis will move to the relative target position indicated by the P (II) set at the V (II) set.
- If the target position (P (II)) specified is a positive value, the motor used will rotate clockwise. If the target position (P (II)) specified is a negative value, the motor used will rotate counterclockwise.
- After motion is started, the speed of the motion will increase from the $V_{BIAS}$ set to the V (I) set. After DOG's signal goes from low to high or from high to low, the speed of the motion will increase/decrease from the V (I) set to the V (II) set. The motion will not stop until the number of pulses output is near the P (II) set.




## 7.11 Status Flags and Status Registers

1. Ready flag

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SM1048+100*N |

[Description]

- There are ready flags for axes. For example, SM1048 is the ready flag for the first axis, SM1148 is the ready axis for the second axis, and SM1248 is the ready flag for the third axis. Users can judge whether an axis still operates by means of the flag for the axis.
- Description of SM1048: SM1048 is ON before the first axis operates. After the first axis begins to operate, SM1048 will be turned from ON to OFF. After the operation of the first axis is complete, SM1048 will b e turned from OFF to ON.

2. Motion error flag

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SM1049+100*N |

[Description]

- If an error occurs in an axis, an error message will be stored in the register for the axis.
- If users want to eliminate the error occuring in an axis, they have to clear the error message stored in the register for the axis, and reset the motion error flag for the axis.

3. State of the axis specified

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1040+100*N |

[Description]

| Special data register | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1040+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | - | - | - | - | - | - | - | - | - | - | The axis pauses. | An error occurs. | The axis specified is being operating. | Negative pulses are being output. | Positive pulses are being output. |

4. Axis error code

| (1+N)th axis | |
|---|---|
| **HW** | **LW** |
| - | SR1041+100*N |

[Description]

Please refer to appendix A for more information.

**MEMO**

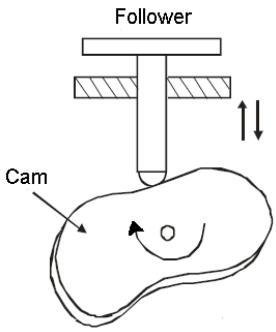
# Chapter 8 Electronic Cam

## Table of Contents

# 8.1 Introduction of Electronic Cams

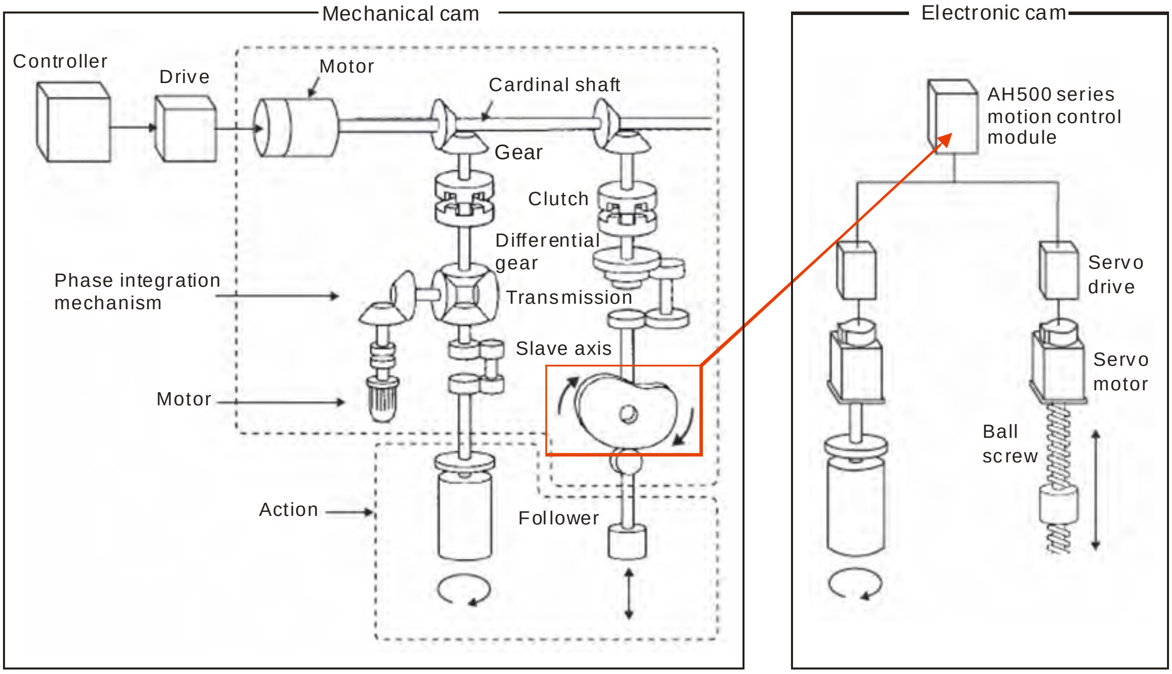
A traditional mechanical cam is composed of a cam, a follower, and a support.

1. A mechanical cam is a rotating sliding piece with irregular shape. In general, it is an input object which rotates at a uniform speed. It makes a follower move regularly by coming into contact with the follower.
2. A follower is a part driven by a mechanical cam. In general, it is an output object which generates motion which is not uniform, sequential, and regular motion.
3. A support is a piece that which is used to support a mechanical cam and a follower.

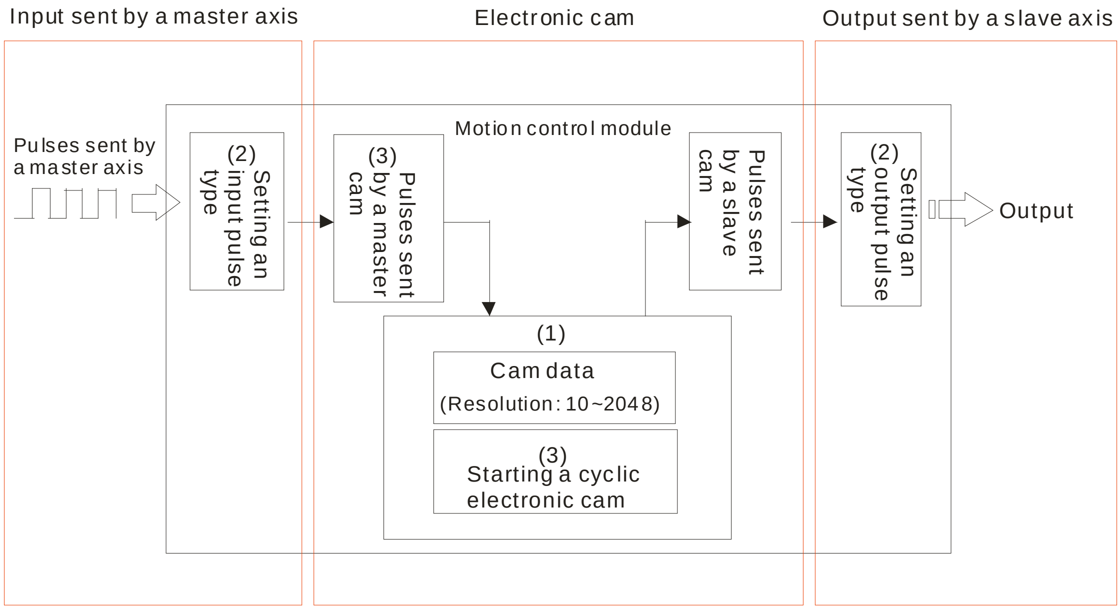



Compared with a traditional cam, an electronic cam has the following advantages.

1. Friendlier user interface
2. Different products require different cam curves. Users can modify the electronic cam data in an electronic cam in software. They do not need to modify a mechanism.
3. High acceleration
4. Smoother operation



# 8.2 Operation of an Electronic Cam




| Step 1 | Step 2 | Step 3 |
|---|---|---|
| Initial setting | Setting a master axis and a slave axis | Starting/Stopping an electronic cam |
| Creating electronic cam data (1)<br>Setting an input /a output pulse type (2) | Setting a master axis (3)<br>Setting the starting angle of the master axis specified (3)<br>Setting a slave axis (3) | Starting/Stopping a cam which operates cyclically (3) |

## 8.2.1 Initial Setting

### 8.2.1.1 Creating Electronic Cam Data

There are two methods of creating electronic cam data.

Method 1: Function that relates the positions of a master axis to the positions of a slave axis

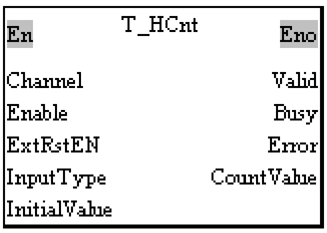
Method 2: Measuring the relation between the positions of a master axis and the positions of a slave axis at work

Please refer to section 8.3 for more information.

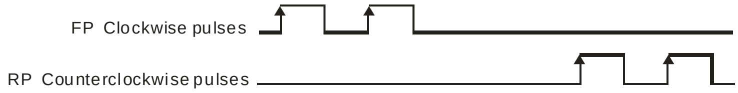
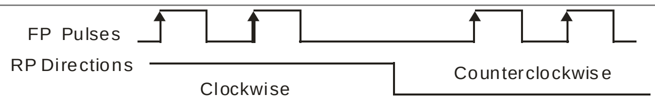
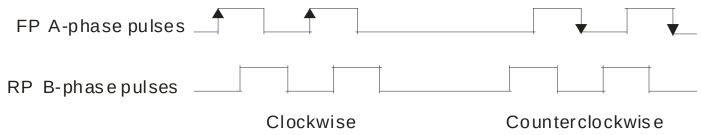
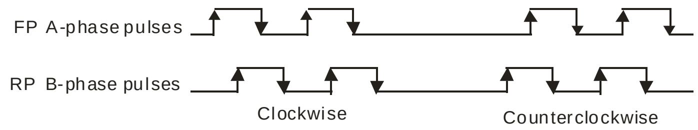
### 8.2.1.2 Setting an Input / a Output Pulse Type

1. Setting an input pulse type

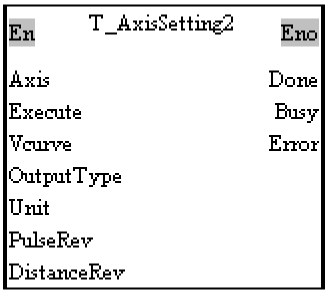
   The master axis specified can be a manual pulse generator, a motion axis, C200, C204, C208, C212, C216, or C220. If users use a counter as a master axis, they have to set an input pulse type. They can set an input pulse type for the counter used by means of the motion control function block T_HCnt.



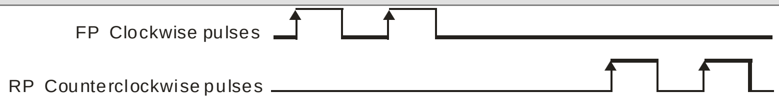
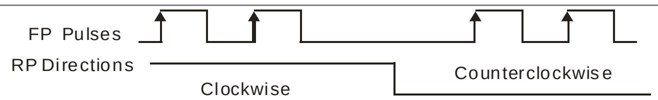
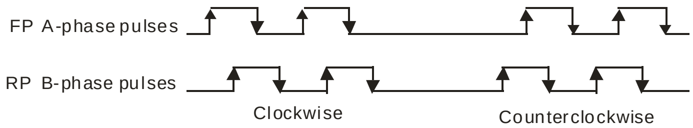
2. Value of the InputType input pin

| Input value | Input type (positive logic) | Description |
|---|---|---|
| mcUD (0) |   | Counting up/down |
| mcPD (1) |   | Pulses+Directions |
| mcAB (2) |   | A/B-phase pulses |
| mc4AB (3) |   | Four times the frequency of A/B-phase pulses |

3. Setting an output pulse type
   If a pulse-type motion controller (an AH10PM series motion control module) is used to execute cam motion, an output pulse type has to be set. If a communication-type motion controller is used to execute cam motion, no output pulse type needs to be set. User can set an output pulse type by means of the motion control function block T_AxisSetting2

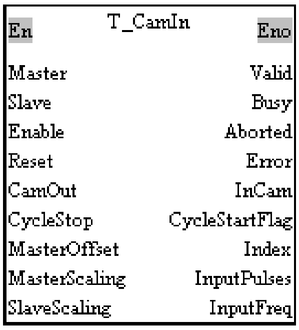



4. Value of the OutputType input pin

| Input value | Output type (positive logic) | Description |
|---|---|---|
| mcUD (0) |   | Counting up/down |
| mcPD (1) |   | Pulses+Directions |
| mcAB (2) |   | A/B-phase pulses |
| mc4AB (3) | | |

## 8.2.2 Setting a Master / Slave Axis and Operating an Electronic Cam

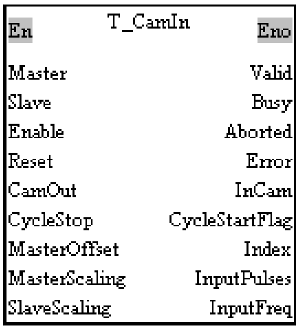
User can set a master axis and a slave axis, and operate an electronic cam by means of the motion control function block T_CamIn.




After the setting of the input pins in the motion control function block T_CamIn is complete, the function set will be enabled if the Enable input pin is set to True. The output pins in the motion control function block T_CamIn can be used to monitor the electronic cam motion set.

### 8.2.2.1 Setting a Master Axis

Users can set a master axis by means of the Master input pin in the motion control function block T_CamIn.

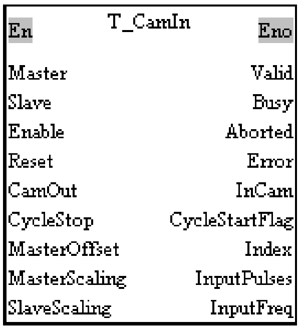



1. Value of the Master input pin

| Input value | Definition | Description |
|---|---|---|
| 0 | Manual pulse generator | An external manual pulse generator is used as a master axis. |
| 1~16 | Motion axis | A motion axis is used as a master axis. |
| 200 | Counter | The counter C200 is used as a master axis. |
| 204 | Counter | The counter C204 is used as a master axis. |
| 208 | Counter | The counter C208 is used as a master axis. |
| 212 | Counter | The counter C212 is used as a master axis. |
| 216 | Counter | The counter C216 is used as a master axis. |
| 220 | Counter | The counter C220 is used as a master axis. |

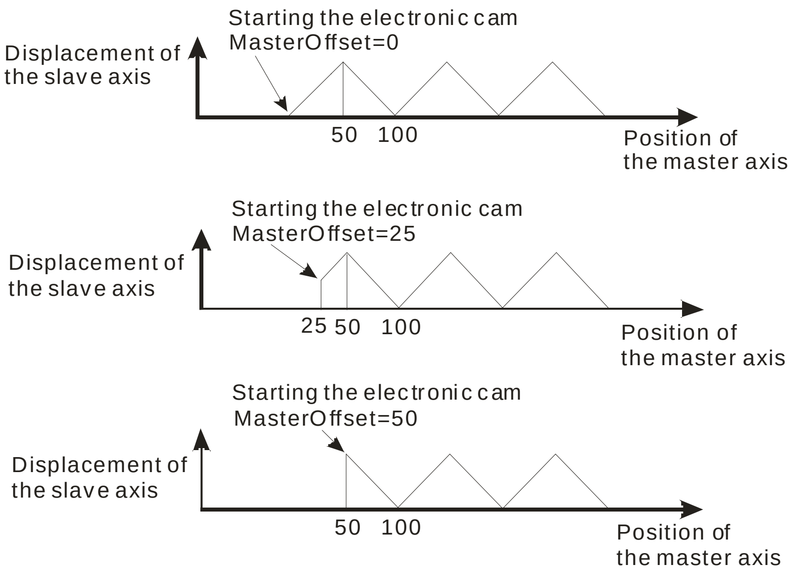
#### 8.2.2.2 Setting the Starting Angle of a Master Axis

Users can set the starting angle of the master axis specified by means of the MasterOffset input pin in the motion control function block T_CamIn.



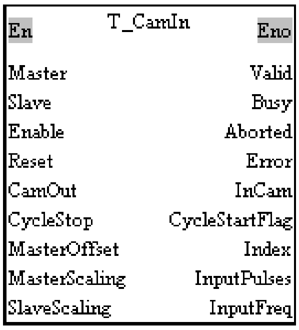

1. Value of the Masteroffset input pin

| Input value | Definition | Description |
|---|---|---|
| K0~K2,147,483,647 | Starting angle of the master axis specified | A pulse is a unit. The value of the Masteroffset input pin indicates the starting point of a cam curve. |



### 8.2.2.3 Setting a Slave Axis

Users can set a slave axis by means of the Slave input pin in the motion control function block T_CamIn.



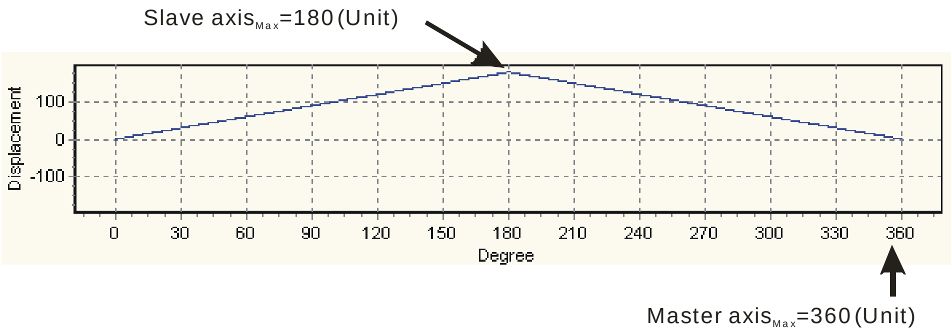

1. Value of the Slave input pin

| Input value | Definition | Description |
|---|---|---|
| 1~16 | Motion axis | A motion axis is used as a slave axis. |

## 8.2.3 Starting / Stopping an Electronic Cam Operating Cyclically

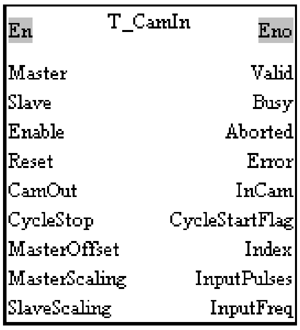
If an electronic cam operates cyclically, the slave axis of the electronic cam moves in accordance with electronic cam data when the master axis of the electronic cam moves. Electronic cam data defines only one cycle. The relation between the positions of a master axis and the positions of a slave axis is the repeated extension of electronic cam data.

Electronic cam data



### 8.2.3.1 Starting an Electronic Cam Operating Cyclically

After users set a master axis, a slave axis, and a starting angle by means of input pins in the motion control function block T_CamIn, electronic cam motion will be started if the Enable input pin is set to True.

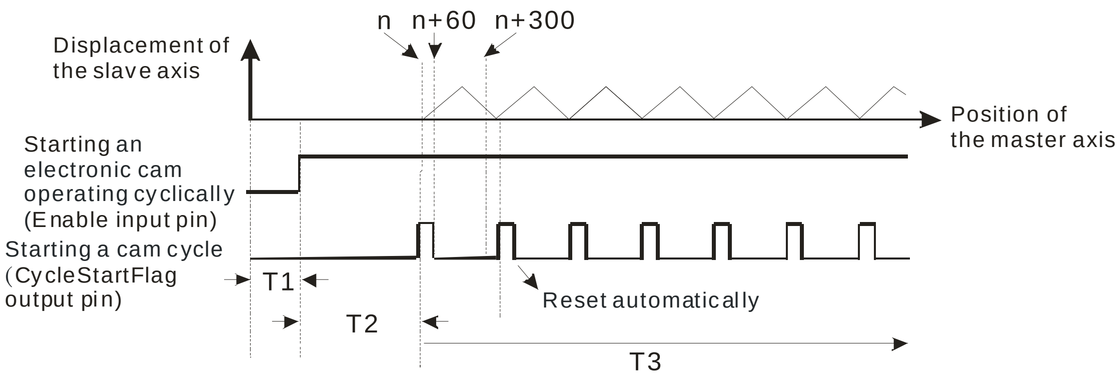



1. Value of the CycleStartFlag output pin

| Output value | Definition | Description |
|---|---|---|
| True/False | Starting a cam cycle | When the output pin is set to True, a cam cycle begins. The value of the CycleStartFlag output pin remains unchanged for one scan cycle. |

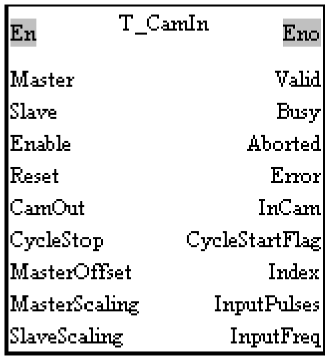
The steps of starting an electronic cam which operates cyclically are as follows.

- At the time T1, the Enable input pin is set to True (an electronic cam which operates cyclically is started).
- After the time T2 elapses, the CycleStartFlag output pin will be set to True. The value of the CycleStartFlag output pin will be cleared after one scan cycle.
- During the time T3, the initialization of the electronic cam is complete, the electronic cam operates cyclically, and the motion of the slave axis specified follows the motion of the master axis specified in accordance with the electronic cam data created.



### 8.2.3.2 Stopping an Electronic Cam Operating Cyclically

Users can stop an electronic cam by means of the Enable input pin and the CamOut input pin in the motion control function block T_CamIn.




1. Value of the CamOut input pin

| Input value | Definition | Description |
|---|---|---|
| True/False | Not meshing with the master axis specified | If the CamOut input pin is set to True, the slave axis specified will not mesh with the master axis specified. |

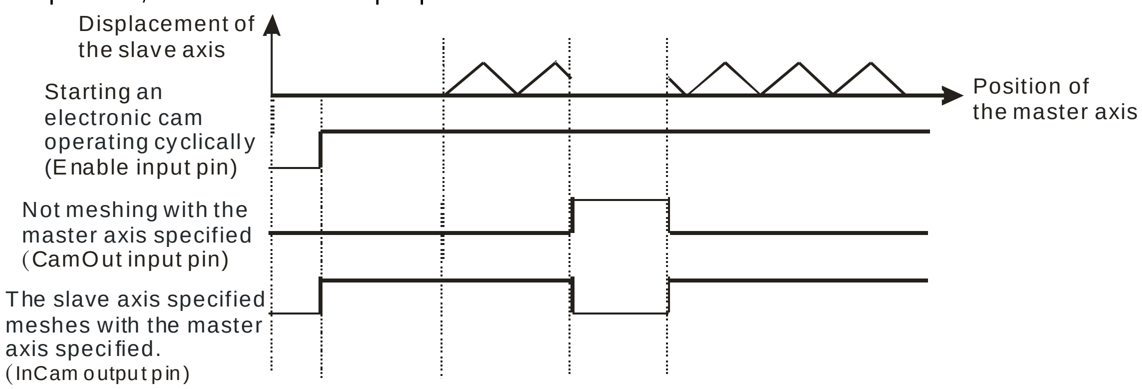
2. Value of the CycleStop input pin

| Input value | Definition | Description |
|---|---|---|
| True/False | Stopping a whole cycle | If the CycleStop input pin is set to True when the Enable input pin is reset, cam motion will not stop until a cycle is complete. |

3. Value of the InCam output pin

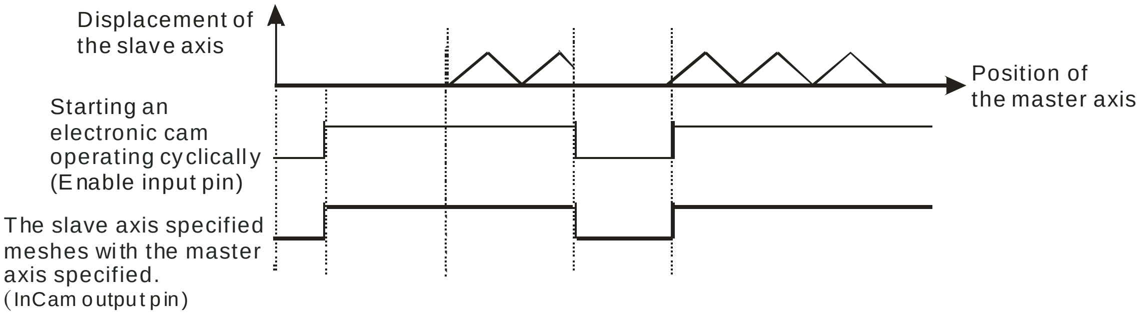
| Output value | Definition | Description |
|---|---|---|
| True/False | The slave axis specified meshes with the master axis specified. | If the slave axis specified meshes with the master axis specified, the InCam output pin will be set to True |

4. Stopping electronic cam motion by means of the CamOut input pin
   If the CamOut input pin is set to True when the Enable input pin is True, the slave axis specified will not mesh with the master axis specified, and the InCam output pin will be set to False. If the CamOut input pin is reset to False, the slave axis specified will mesh with the master axis specified, and the InCam output pin will be set to True.

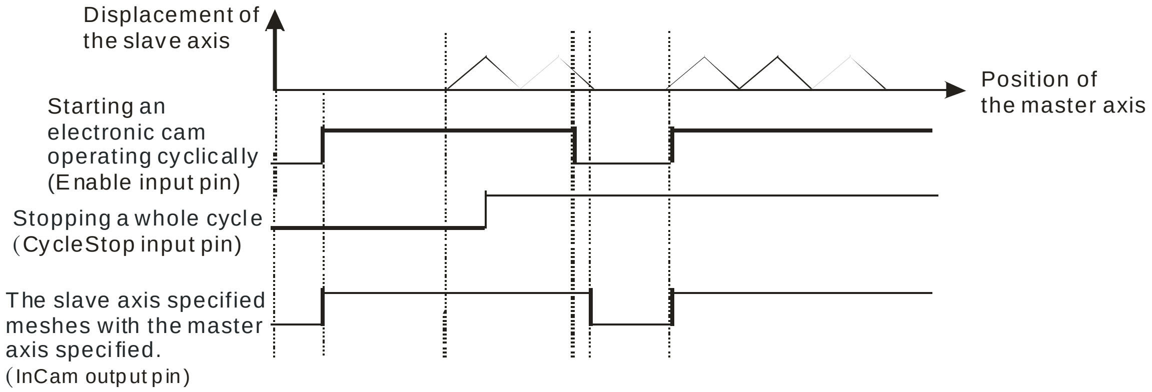


5. Stopping electronic cam motion by means of the Enable input pin

If the Enable input pin is set to True, the slave axis specified will mesh with the master axis specified. If the Enable input pin is reset to OFF, the slave axis specified will not mesh with the master axis specified, and the InCam output pin will be set to False. If the Enable input pin is set to True again, the slave axis specified will mesh with the master axis specified.




If the Enable input pin is set to True, the slave axis specified will mesh with the master axis specified. If the Enable input pin is set to False after the CycleStop input pin is set to True, cam motion stops when a cycle is complete, and the InCam output pin is False when the cam motion stops. If the Enable input pin is set to True again, the slave axis specified will mesh with the master axis specified.



## 8.3 Creating Electronic Cam Data

Electronic cam data defines the relation between the positions of a master axis and the positions of a slave axis.

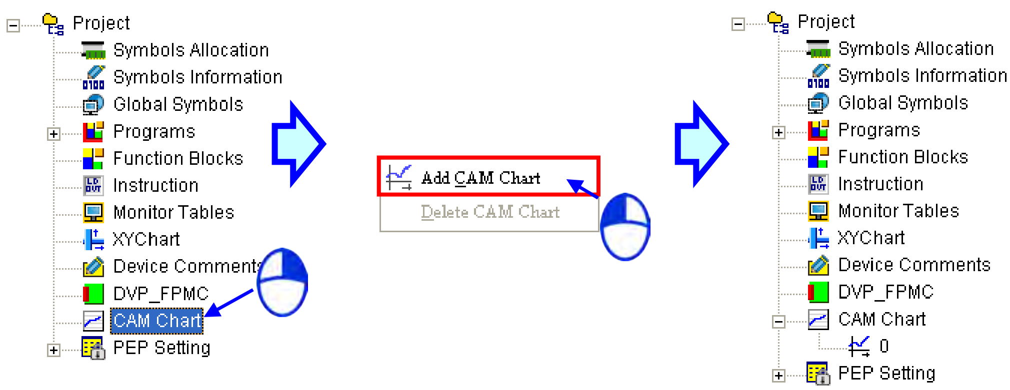
1. Before users create a cam chart in PMSoft, they have to know the relation between the positions of a master axis and the position a slave axis. There are two methods of getting the relation between the positions of a master axis and the positions of a slave axis.
   - Method 1: Function that relates the positions of a master axis to the positions of a slave axis
   - Method 2: Measuring the relation between the positions of a master axis and the positions of a slave axis at work

After electronic cam data defines the relation between the positions of a master axis and the positions of a slave axis, users can get the positions of the slave axis by means of the positions of the master axis.

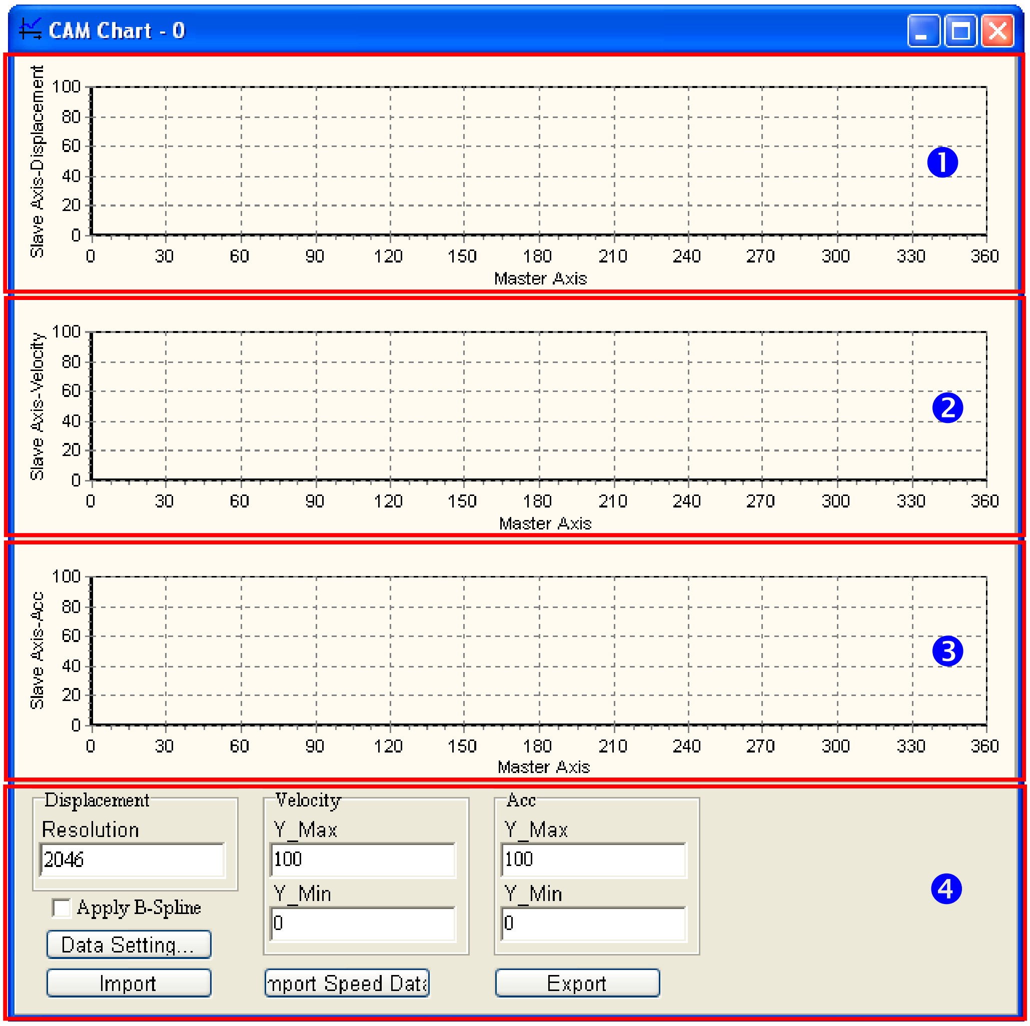
### 8.3.1 Creating a Cam Chart in PMSoft

#### 8.3.1.1 Function Relates the Positions of a Master Axis to the Positions of a Slave Axis

After users create a project in PMSoft, right-click **CAM Chart** in the system information area, and click **Add CAM Chart** on the context menu, the **CAM Chart-0** window will appear.



The **CAM Chart-0** window is shown below.




❶ Displacement: The relation between the master axis and the slave axis is described in terms of displacement.

❷ Velocity: The relation between the master axis and the slave axis is described in terms of speed.

❸ Acceleration: The relation between the master axis and the slave axis is described in terms of acceleration.

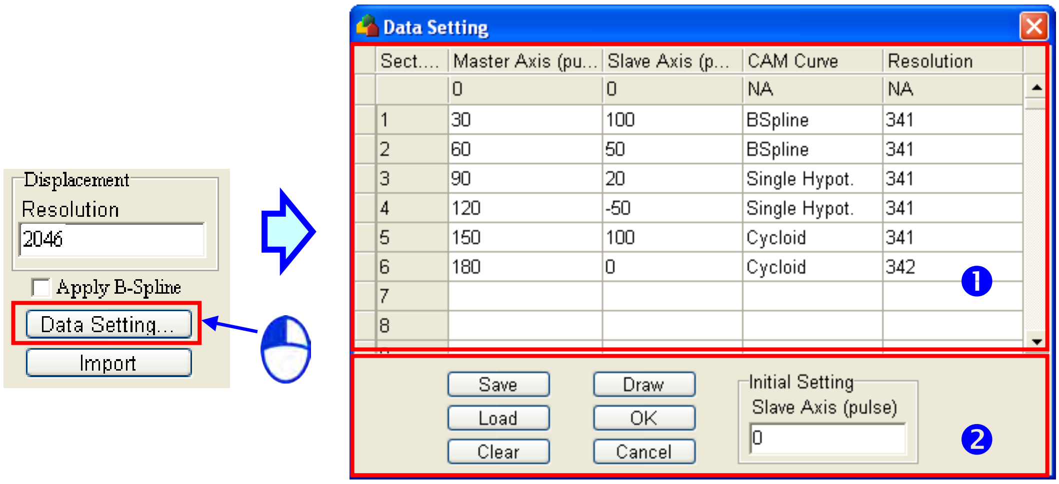
❹ Data setting area:

- **Resolution**: Users can set the number of data points required in the electronic cam chart. The number of data points must be in the range of 10 to 2046.
- **Velocity**: The maximum speed of the slave axis and the minimum speed of the slave axis are shown in this section. They are calculated by the system according to the data related to displacement. Users can change the maximum speed of the slave axis and the minimum speed of the slave axis by themselves.
- **Acc**: The maximum acceleration of the slave axis and the minimum acceleration of the slave axis are shown in this section. They are calculated by the system according to the data related to displacement. Users can change the maximum acceleration of the slave axis and the minimum acceleration of the slave axis by themselves.
- **Data Setting...**: The description of the relation between the master axis and the slave axis in terms of displacement is shown in the **Data Setting** window. The displacement resolution set in the data setting area will be brought into the

**Data Setting** window after the **Data Setting** window is opened. If users click the **Apply B-spline** checkbox in the data setting area, **B-spline** will be automatically selected in the **Data Setting** window.

- **Import**: Importing the description of the relation between the master axis and the slave axis in terms of displacement
- **Export**: Exporting the description of the relation between the master axis and the slave axis in terms of displacement
- **Import Speed Data**: Importing the description of the relation between the master axis and the slave axis in terms of speed

After the users click **Data Setting…** in the **CAM Chart-0** window, the **Data Setting** window will appear. The **Data Setting** window is composed of sections. The users can set a section of a cam curve in every section. A complete cam curve is composed of several sections. The users can set 360 sections at most. An electronic cam cycle is composed of the sections created by the users.




❶ Users can define the relation between the master axis and the slave axis in every section.

- **Master Axis**: Users can set the displacement of the master axis. A pulse is a unit of the measurement for displacement. The values that the users type in the **Master Axis (pulse)** column must be greater than 0, and must be in numerical order.
- **Slave Axis**: Users can set the displacement of the master axis. A pulse is a unit of the measurement for displacement. The values that the users type in the **Slave Axis (pulse)** column can be positive values or negative values.
- **CAM Curve**: The functions which can be selected are **Const Speed**, **Const Acc.**, **Single Hypot.**, **Cycloid**, and **B-Spline**. If users click the **Apply B-spline** checkbox in the **CAM Chart-0** window, **B-spline** will be automatically selected in the **Data Setting** window.
- **Resolution**: Users can set the number of data points used in a section. The number of data points must be in the range of 10 to 2047. If the users do not set resolutions for sections, the number of data points left will be equally distributed to the sections. The users have to set resolutions according to equipment's requirements. The higher the resolutions set are, the more smoothly the equipment used operates. Besides, the size of the electronic cam data gotten is big if the resolutions set are high.

❷ After sections of a cam curve are created, users can click **Save**, **Load**, **Clear**, **Draw**, **OK**, **Cancel**, or set the initial position of the slave axis.

- **Save**: Saving the data set in sections
- **Load**: Loading the data which was saved
- **Clear**: Clearing all the data in sections
- **Draw**: Compiling the data set in sections, and drawing the electronic cam data gotten on the electronic cam chart created
- **OK**: Compiling the data set in sections, drawing the electronic cam data gotten on the

electronic cam chart created, and closing the **Data Setting** window

- **Cancel**: Closing the **Data Setting** window.
- **Initial Setting**: Setting the initial position of the slave axis

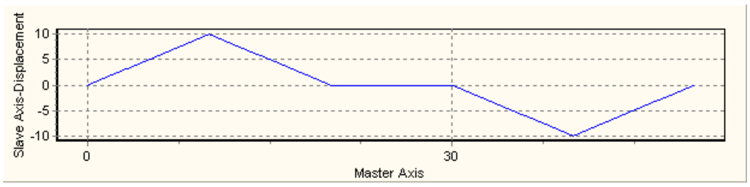
After the users click “Export” in the **CAM Chart-0** window, the displacement chart, the velocity chart, and the acceleration chart in the **CAM Chart-0** window will be saved in the CAMData folder in the folder in which PMSoft is installed. There are three files in the CAMData folder.

- <Folder in which PMSoft is installed>\CAMData\Data_S.txt: Displacement
- <Folder in which PMSoft is installed>\CAMData\Data_V.txt: Velocity
- <Folder in which PMSoft is installed>\CAMData\Data_A.txt: Acceleration

Data_S.txt, Data_V.txt, and Data_A.txt are shown below.

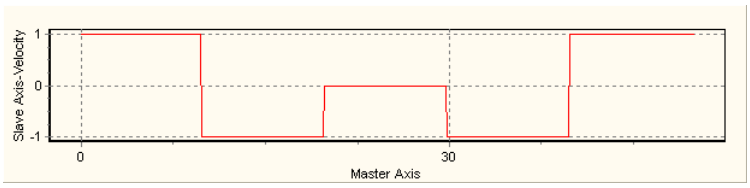
Displacement chart

Path: <Folder in which PMSoft is installed>\CAMData\Data_S.txt



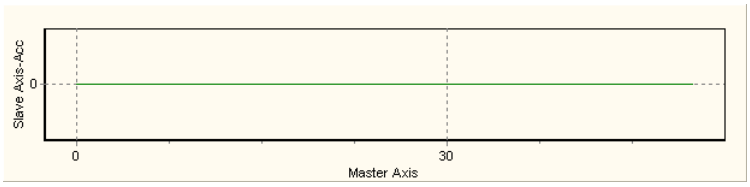

Velocity chart

Path: <Folder in which PMSoft is installed>\CAMData\Data_V.txt




Acceleration chart

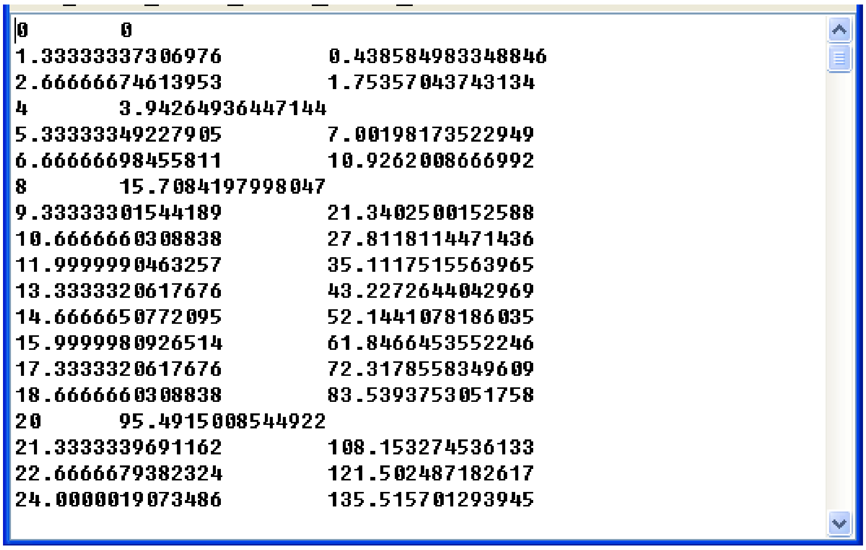
Path: <Folder in which PMSoft is installed>\CAMData\Data_A.txt



### 8.3.1.2 Measuring the Relation between the Positions of a Master Axis and the Positions of a Slave Axis at Work

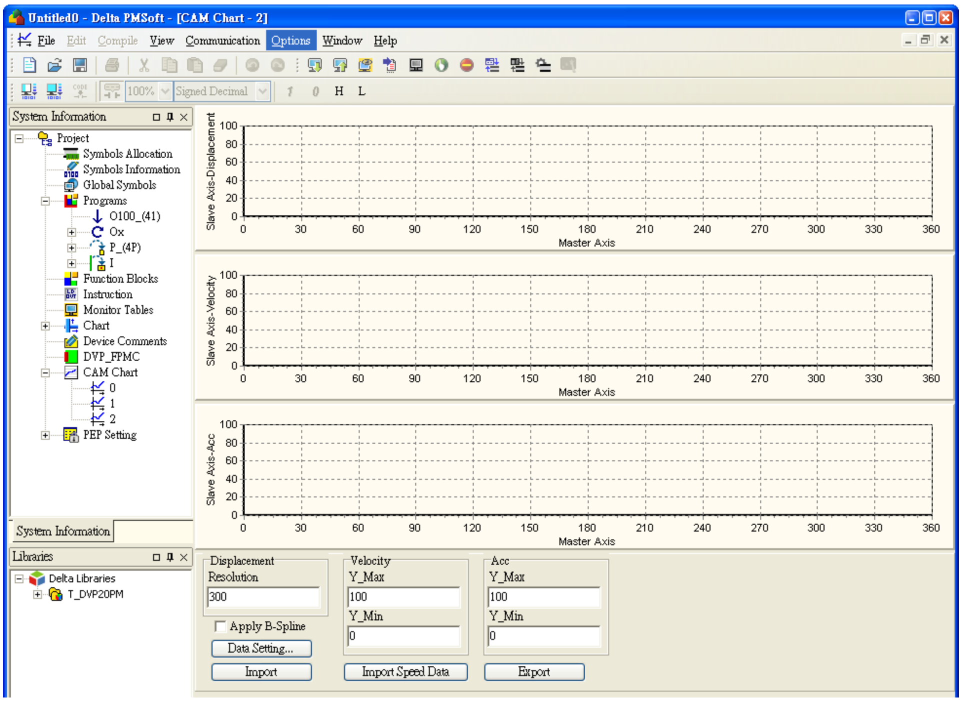
Users can store the relation between the positions of the master axis specified and the positions of the slave axis specified in a file in the CAMData folder, and then import the relation into a cam chart in PMSoft. The steps of importing the relation between the positions of the master axis specified and the positions of the slave axis specified into a can chart in PMSoft are as follows.

1. Store data about displacement in Data_S.txt in the folder in the CAMData folder.

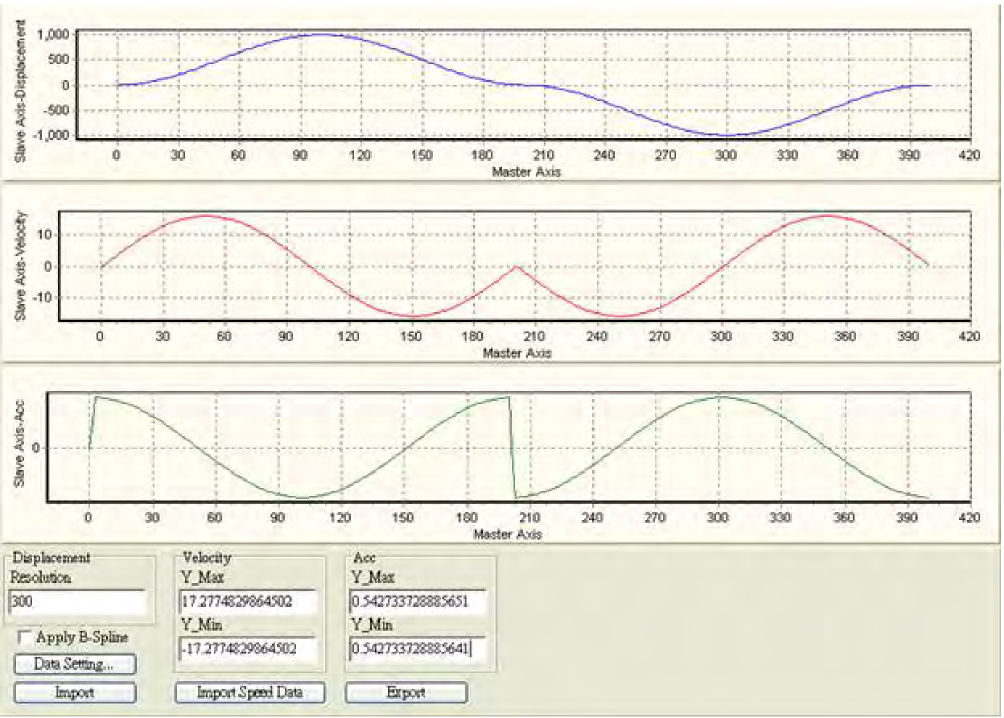


```
0	0
1.33333337306976	0.438584983348846
2.66666674613953	1.75357043743134
4	3.94264936447144
5.33333349227905	7.00198173522949
6.66666698455811	10.9262008666992
8	15.7084197998047
9.33333301544189	21.3402500152588
10.6666660308838	27.8118114471436
11.9999990463257	35.1117515563965
13.3333320617676	43.2272644042969
14.6666650772095	52.1441078186035
15.9999980926514	61.8466453552246
17.3333320617676	72.3178558349609
18.6666660308838	83.5393753051758
20	95.4915008544922
21.3333339691162	108.153274536133
22.6666679382324	121.502487182617
24.0000019073486	135.515701293945
```

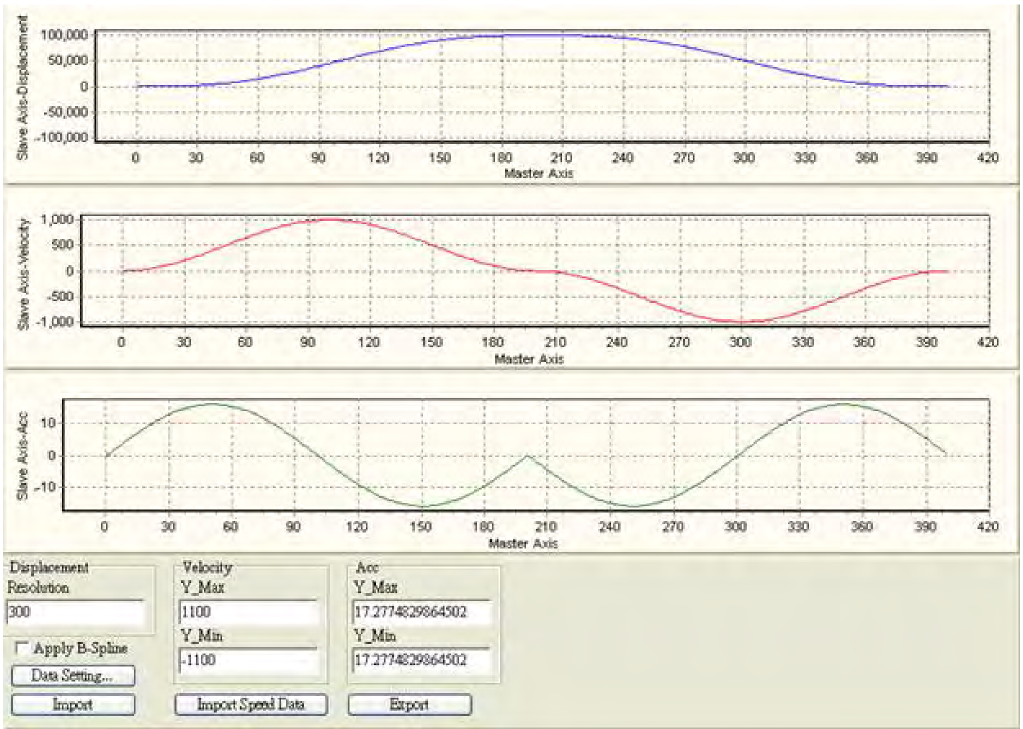
2. Open a **CAM Chart** window in PMSoft, and then type a value in the **Resolution** box.



3. After **Import** is clicked, PMSoft will read Data_S.txt and draw a displacement chart in the **CAM Chart** window, and a velocity chart and an acceleration chart will be drawn in accordance with the contents of Data_S.txt.




4. After **Import Speed Data** is clicked, PMSoft will read Data_S.txt and draw a velocity chart in the **CAM Chart** window, and a displacement chart and an acceleration chart will be drawn in accordance with the contents of Data_S.txt



### 8.3.1.3 Creating / Modifying Electronic Cam Data

After users create electronic cam data in a cam chart in PMSoft, the cam data will be downloaded to an AH500 series motion control module. If the users modify the electronic cam data in PMSoft, they have to download the new electronic cam data created to the AH500 series motion control module again after they modify the electronic cam data. If the users want to modify the electronic cam data in the program in the AH500 series motion control module, they can use motion control function blocks.

The motion control function block T_CamRead is used to read a particular point in a cam chart, and the motion control function block T_CamWrite is used to modify a particular point in a cam chart.

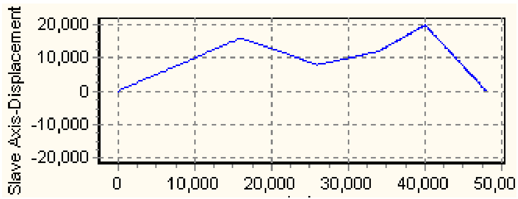
Note: If users want to modify all the points in a cam chart, the pair of coordinates (0, 0) will need to be written after the last point is modified.

| T_CamRead | T_CamWrite |
|---|---|
| T_CamRead<br>En Eno<br>Axis Valid<br>Enable Error<br>CamPointNo MasterPosition<br>SlavePosition | T_CamWrite<br>En Eno<br>Axis Done<br>Execute Busy<br>CamPointNo Error<br>MasterPosition<br>SlavePosition |

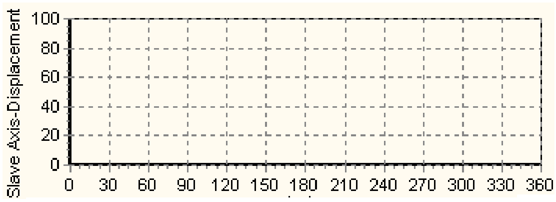
1. Example

【Function】

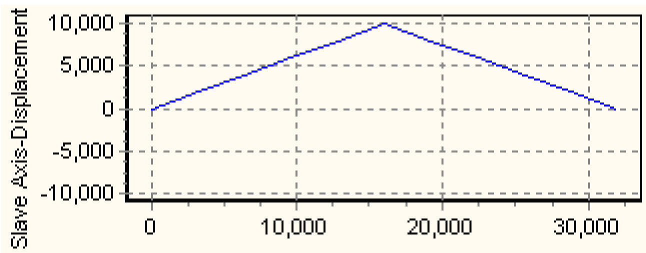
Users can modify a particular point in a cam chart. In figure (a), there is an original cam chart. There may be data or no data in the cam chart which will be modified. In figure (b), the three pairs of coordinates (16000, 10000), (32000, 0), and (0, 0) are written.




or



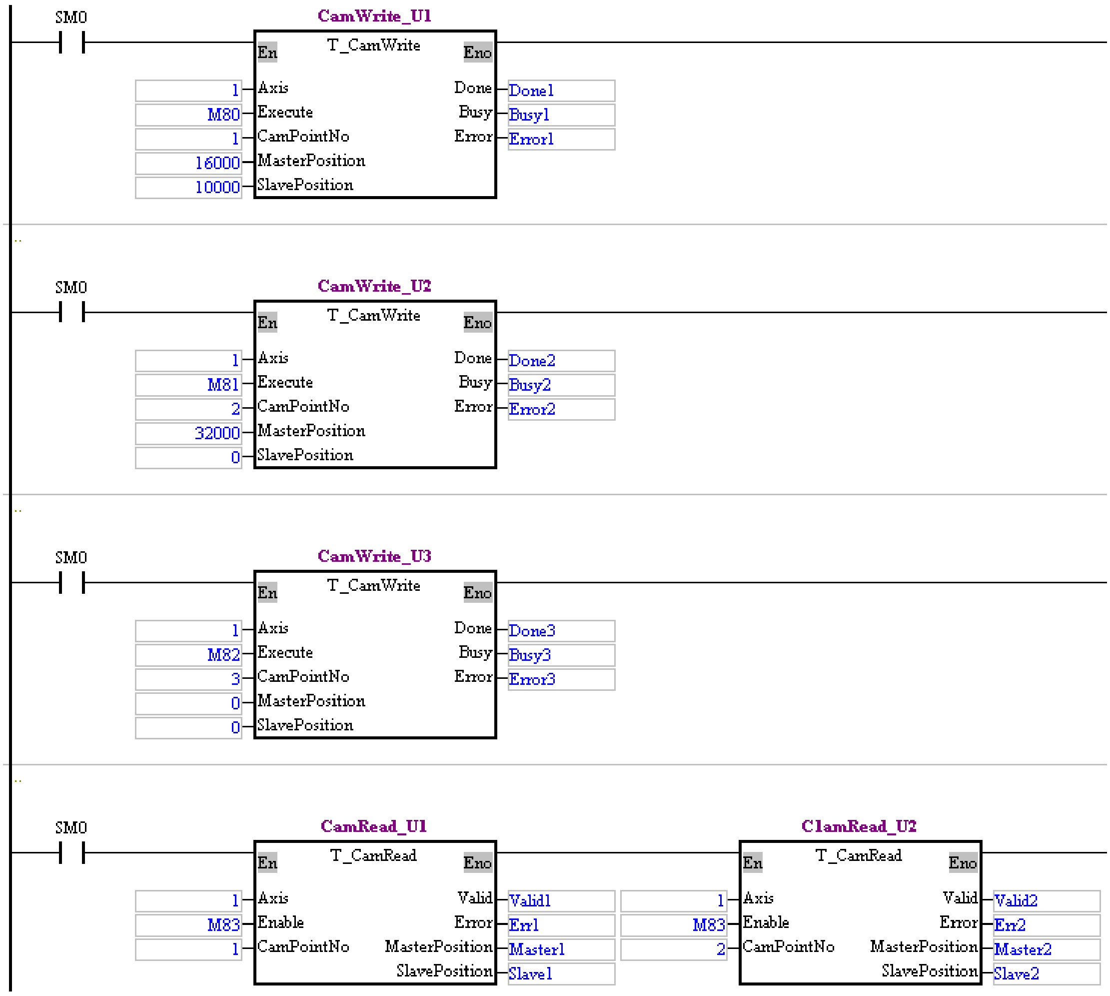

(a) Original cam chart




(b) Three pairs of coordinates in a cam chart

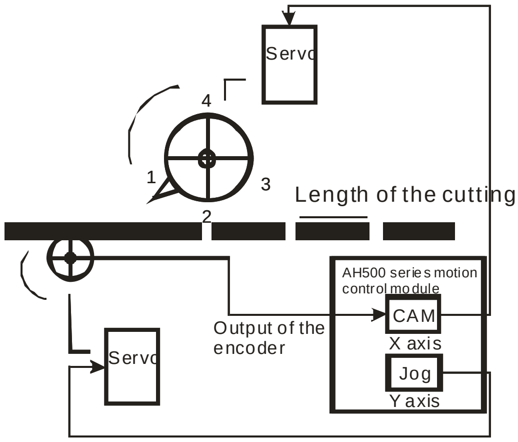
【Steps】

- Set M80 to True. The pair of coordinates (16000, 10000) is written into cam point number 1.
- Set M81 to True. The pair of coordinates (32000, 0) is written into cam point number 2.
- Set M82 to True. The pair of coordinates (0, 0) is written into cam point number 3.
- Set M83 to True. Cam point number 1 and cam point number 2 are read. Check whether the values read are the same as the values written into cam point number 1 and cam point number 2.


SM0
CamWrite_U1
T_CamWrite
En
Eno
1 Axis
M80 Execute
1 CamPointNo
16000 MasterPosition
10000 SlavePosition
Done Done1
Busy Busy1
Error Error1
SM0
CamWrite_U2
T_CamWrite
En
Eno
1 Axis
M81 Execute
2 CamPointNo
32000 MasterPosition
0 SlavePosition
Done Done2
Busy Busy2
Error Error2
SM0
CamWrite_U3
T_CamWrite
En
Eno
1 Axis
M82 Execute
3 CamPointNo
0 MasterPosition
0 SlavePosition
Done Done3
Busy Busy3
Error Error3
SM0
CamRead_U1
T_CamRead
En
Eno
1 Axis
M83 Enable
1 CamPointNo
Valid Valid1
Error Err1
MasterPosition Master1
SlavePosition Slave1
ClamRead_U2
T_CamRead
En
Eno
1 Axis
M83 Enable
2 CamPointNo
Valid Valid2
Error Err2
MasterPosition Master2
SlavePosition Slave2

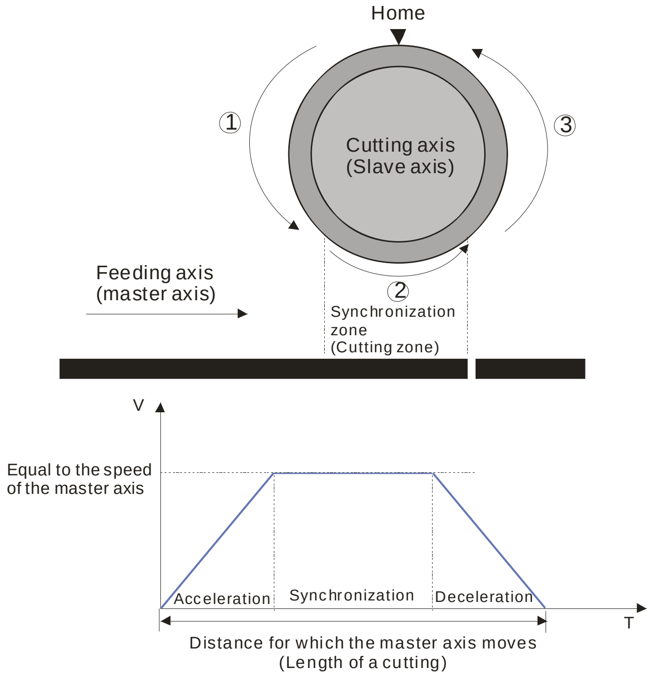
## 8.4 Application of an Electronic Cam—Using a Rotary Cutter

In the application of cutting materials on a feeding belt, a traditional approach is that a cutting roller will rotate after a feeding roller rotates for a certain length, and the alternation of feeding materials and cutting the materials is repeated. The disadvantage of this approach is that the acceleration/deceleration needed in order for a feeding roller to rotate/stop decreases production efficiency. As a result, a new approach is that materials are fed continuously. There are two ways of cutting materials on a feeding belt. They are rotary cut and flying shear. Flying shear is reciprocating motion, while rotary cut is unidirectional motion. The cam curve for rotary cut is different from the cam curve for flying shear. The application of rotary cut is described below.

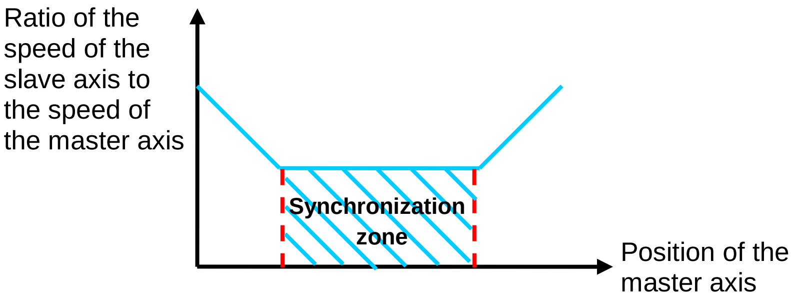



**【Concept】**

1. In the application of rotary cut, the cutting roller of a rotary cutter rotates in a direction. A material is cut when the blade of the rotary cutter comes into contact with the material. The feeding roller of the rotary cutter continuously feeds materials at a uniform speed. The relation between rotary cut and the output generated is shown below.
   - At first, the slave axis accelerates until it moves to the synchronization zone.
   - After the slave axis leaves the synchronization zone, it decelerates until it returns home. A cycle is complete when the slave axis is at home. Users can draw a speed relation chart.

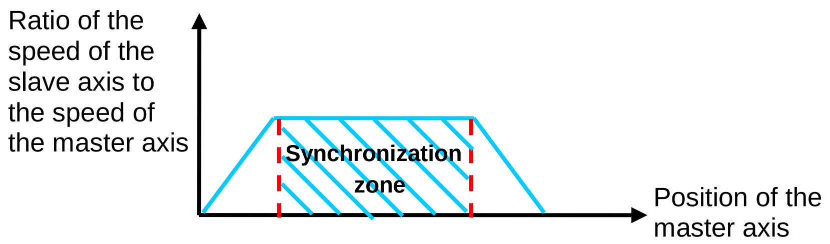



2. During the processing of cutting materials, synchronization is an important factor. When the bladed of a rotary cutter come into contact with a material, the speed of the blade must be the same as the speed of the material. If the speed of the blade of a rotary cutter is greater than the speed of a material when it comes into contact with the material, the force which pulls the material forward will appear, and the material will be cut smoothly. If the speed of the blade of a rotary cutter is less than the speed of a material when it comes into contact with the material, the material will be jammed.
3. The design of a synchronization zone affects the operation of equipment. The bigger the synchronization zone is in a cycle, the less time it takes for the slave axis specified to accelerate/decelerate. If equipment needs to accelerate/decelerate in a short time, there will be a great impact on the electric machinery used and the blade used, and there will be an overcurrent passing through the servo used.
4. Relation between the length of a cutting and the circumference of a blade
   - Length of a cutting<Length of the blade used: The speed of cutting roller used is the same as the speed of the feeding roller used in the synchronization zone designed. After the cutting roller used leaves the synchronization zone, the cutting roller will accelerate.

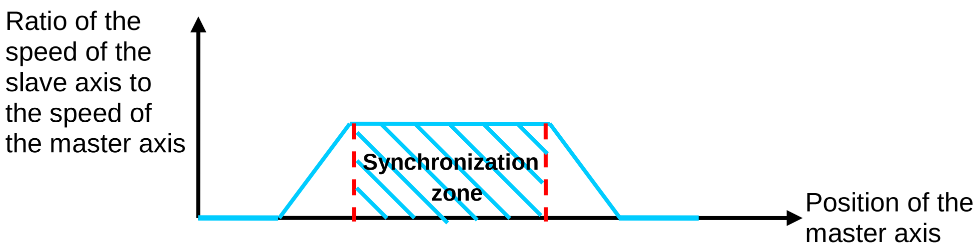


- Length of a cutting=Circumference of the blade used: The cutting roller used rotates at a uniform speed.


- One time the circumference of the blade used<Length of a cutting<Two times the circumference of the blade used: After the cutting roller used completes cutting in the synchronization zone designed, it will decelerate, and then accelerate until its speed is the same the speed of the feeding roller used.




- Length of a cutting>Two times the circumference of the blade used: The length of a cutting is greater than two times the circumference of the blade. (It is a common situation.) After the blade used completes cutting in a cycle, it will decelerate until it stops. After a material of a certain length is fed, the blade used will start cutting again.

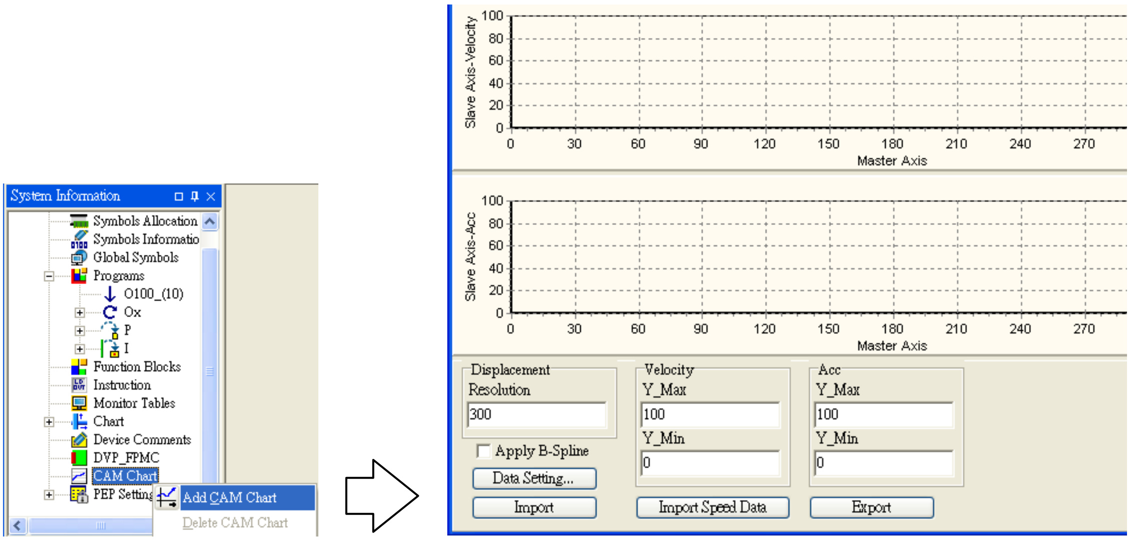



### 8.4.1 Creating Rotary Cut Data

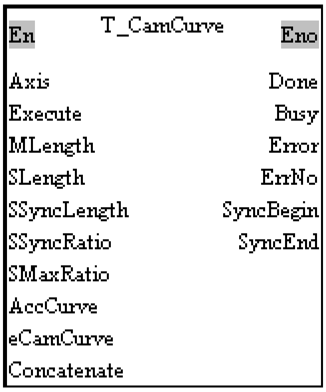
Users can create a rotary cut curve by means of creating electronic cam data in a way introduced in section 8.3. Besides, an AH500 series motion control module provides the motion control function block T_CamCurve. T_CamCurve can be used to automatically create cam data.

## 8.4.2 Function Block—T_CamCurve

1. Creating a cam chart: Users have to create a blank cam chart in PMSoft, and then set resolution in accordance with the number of rotary cut curves. In order to create a rotary cut curve, the users need to type 300 in the **Resolution** box. Download the cam chart to an AH500 series motion control module. When the AH500 series motion control module operates, electronic cam data is stored in the cam chart.




2. Setting the motion control function block T_CamCurve: Users have to set the parameters related to a rotary cutter, including the distance for which the master axis specified moves, the distance for which the slave axis specified is synchronized with the master axis specified, and the synchronization ratio of the speed of the slave axis specified to the speed of the master axis specified. After the motion control function block T_CamCurve is executed, a rotary cut curve will be created.



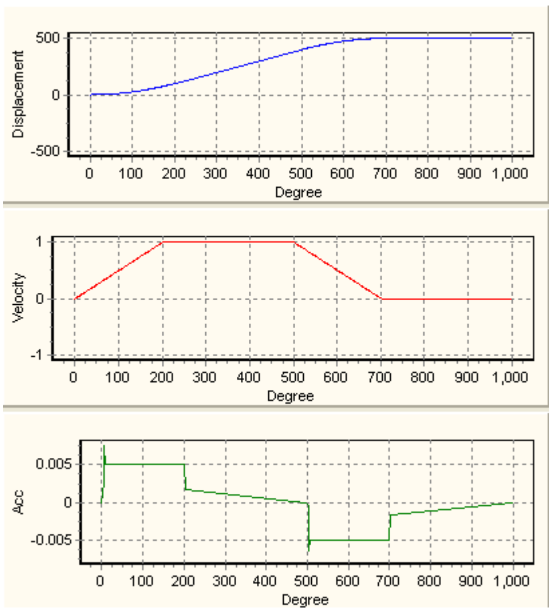

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Description** |
| Axis | Motion axis number | WORD | 1~16 | Slave axis number |

| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Description** |
| Execute | The creation of a rotary cut curve is enabled when there is a transition in the Execute input pin's signal from low to high. | BOOL | True/False | Starting the creation of a rotary cut curve |
| MLength | Distance for which the master axis specified moves | DWORD | K1~K2,147,483,647 | Distance for which the master axis specified moves (PPS) |
| SLength | Distance for which the slave axis specified moves | DWORD | K1~K2,147,483,647 | Distance for which the slave axis specified moves (PPS) |
| SSyncLength | Distance for which slave the slave axis specified is synchronized with the master axis specified | DWORD | K1~K2,147,483,647 | Distance for which the slave axis specified is synchronized with the master axis specified (PPS) |
| SSyncRatio | Synchronization ratio of the speed of the slave axis specified to the speed of the master axis specified | FLOAT | 1.1755x10-38~ 3.4028x10+38 | Synchronization ratio of the speed of the slave axis specified to the speed of the master axis specified (Speed of the slave axis specified/Speed of the master axis specified) |
| SMaxRatio | Maximum ratio of the speed of the slave axis specified to the speed of the master axis specified | FLOAT | $1.1755 \times 10^{-38}$~ $3.4028 \times 10^{+38}$ | Maximum ratio of the speed of the slave axis specified to the speed of the master axis specified |
| AccCurve | Acceleration curve | WORD | 0~3 (*1) | Acceleration curve |
| eCamCurve | Cam curve | WORD | 0~5 (*2) | Rotary cut curve |
| Concatenate | Concatenation | BOOL | True/False | Connecting to the preceding cam curve |
| **State output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Description** |
| Done | The execution of the motion control function block is complete. | BOOL | True/False | The creation of a curve is complete. |

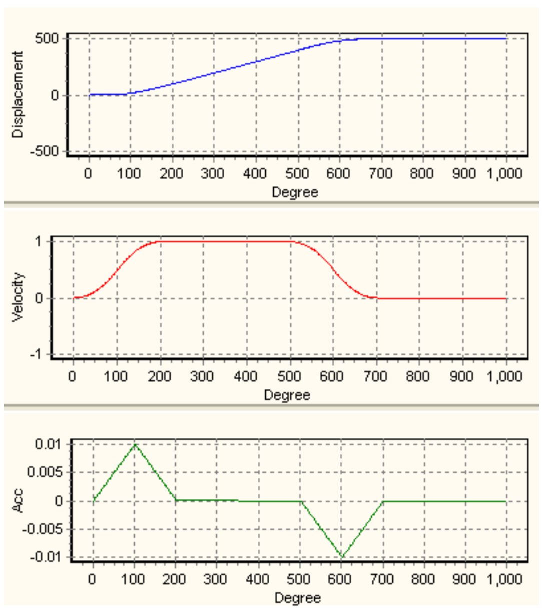
| State output pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Output range** | **Description** |
| Busy | The motion control function block is being executed. | BOOL | True/False | A curve is being created. |
| Error | An error occurs in the motion control function block. | BOOL | True/False | An error occurs when a curve is created. |
| **Value output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Description** |
| ErrNo | Error code | WORD | 0~2 | Error code |
| SyncBegin | Starting point of synchronization | DWORD | K0~K2,147,483,647 | Starting point of synchronization |
| SyncEnd | Terminal point of synchronization | DWORD | K0~K2,147,483,647 | Terminal point of synchronization |

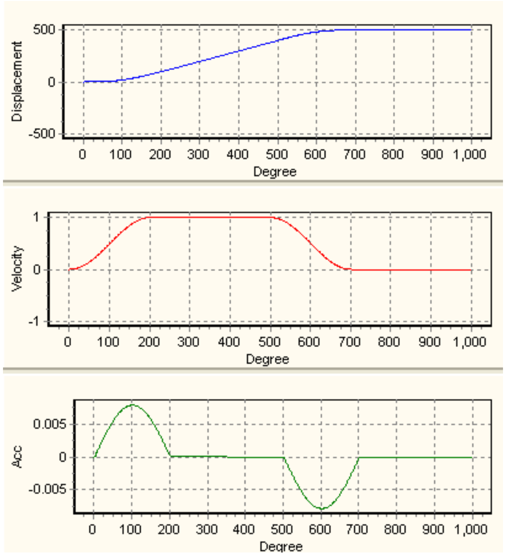
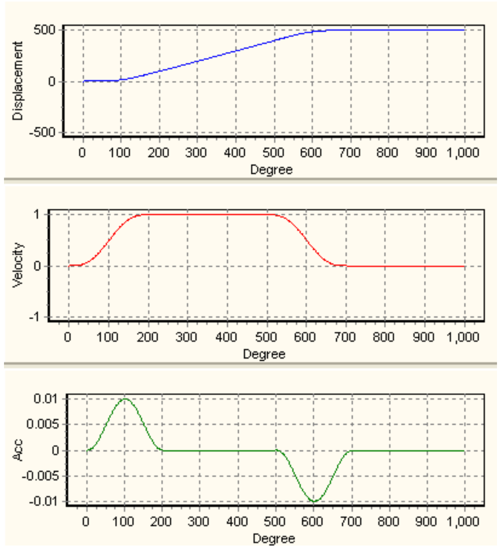
*1: Value of the AccCurve input pin

0: Uniform curve

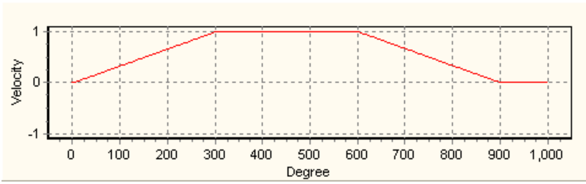
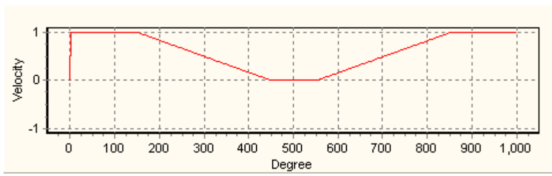
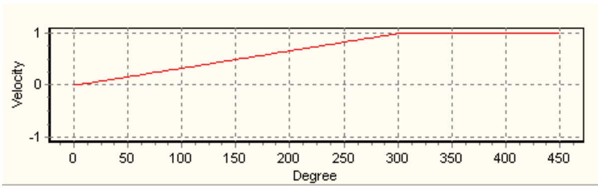
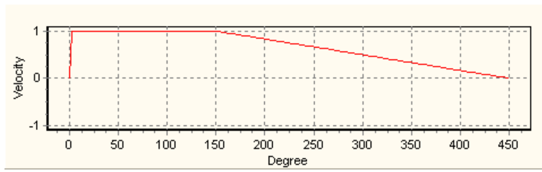
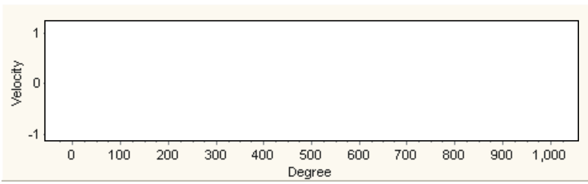



1: Uniform acceleration curve



<table>
<tr><th>2: SingleHypot curve</th><th>3: Cycloid</th></tr>
<tr><td>
</td><td>
</td></tr>
</table>

*2: Value of the eCamCurve input pin

<table>
<tr><td>0: leftCAM</td><td>1: midCAMall</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>2: midCAMbegin</td><td>3: midCAMend</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>5: rightCAM</td><td></td></tr>
<tr><td>
</td><td></td></tr>
</table>

● Example

【Function】

The steps of creating a rotary cut curve by means of the motion control function block T_CamCurve are described below.

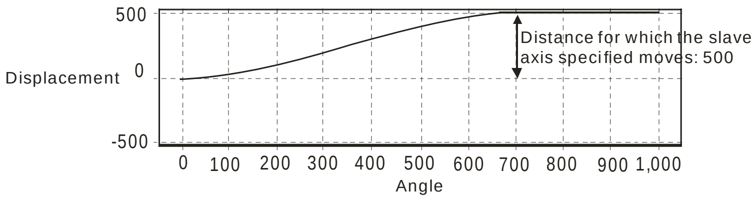
Distance for which the master axis specified moves=1000

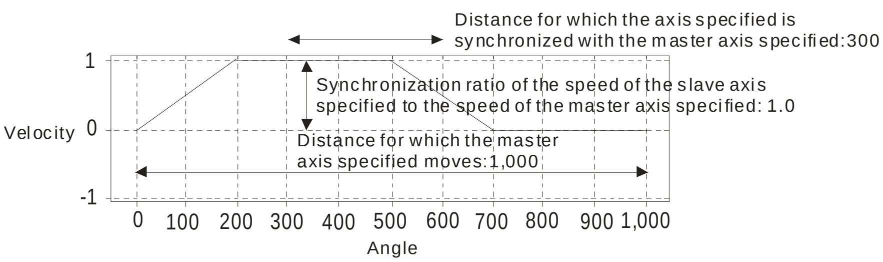
Distance for which the slave axis specified moves=500

Distance for which the slave axis specified is synchronized with the master axis specified=300

Synchronization ratio of the speed of the slave axis specified to the speed of the master axis specified=1.0

The rotary cut curve created and the values of parameters are shown below.



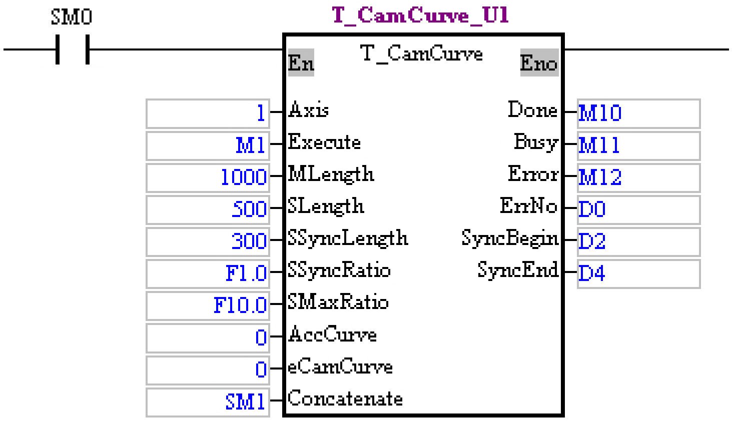



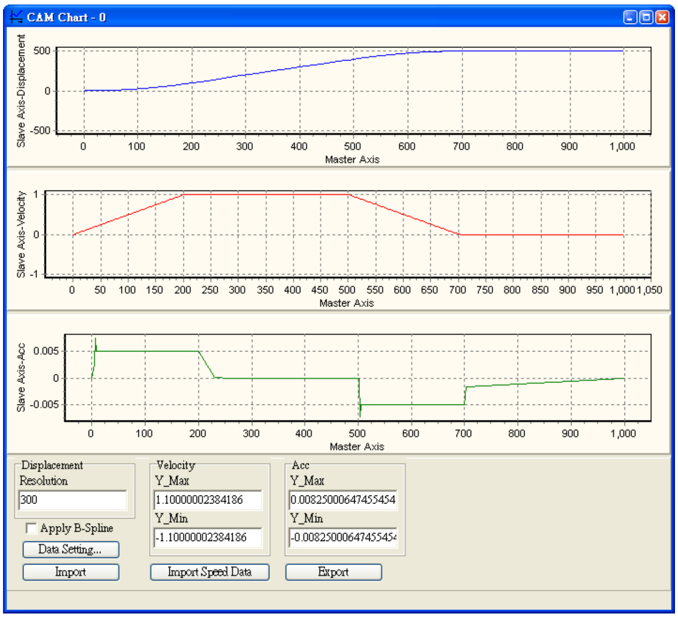

| Parameter | Setting value |
|---|---|
| Distance for which the master axis specified moves | 1000 |
| Distance for which the slave axis specified moves | 500 |
| Distance for which the slave axis specified is synchronized with the master axis specified | 300 |
| Synchronization ratio of the speed of the slave axis specified to the speed of the master axis specified | 1.0 |
| Maximum ratio of the speed of the slave axis specified to the speed of the master axis specified | 10.0 |
| Acceleration curve | 0 |
| Cam curve | 0 |
| Concatenation | 0 |

【Steps】

- Open a **CAM Chart** window in PMSoft, and then type 300 in the **Resolution** box.
- Download the program created to an AH500 series motion control module, and then execute the program.
- After M1 is set to True, a rotary cut curve will be created.
- Stop the AH500 series motion control module, and then upload the program in the AH500 series motion control module.
- View the first curve in the **CAM Chart-0** window. The curve is a rotary cut curve which is created automatically.

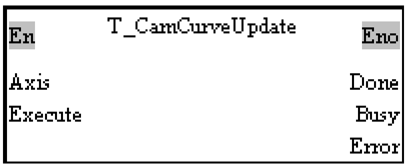
【Program in PMSoft】




## 8.4.3 Function Block—T_CamCurveUpdate

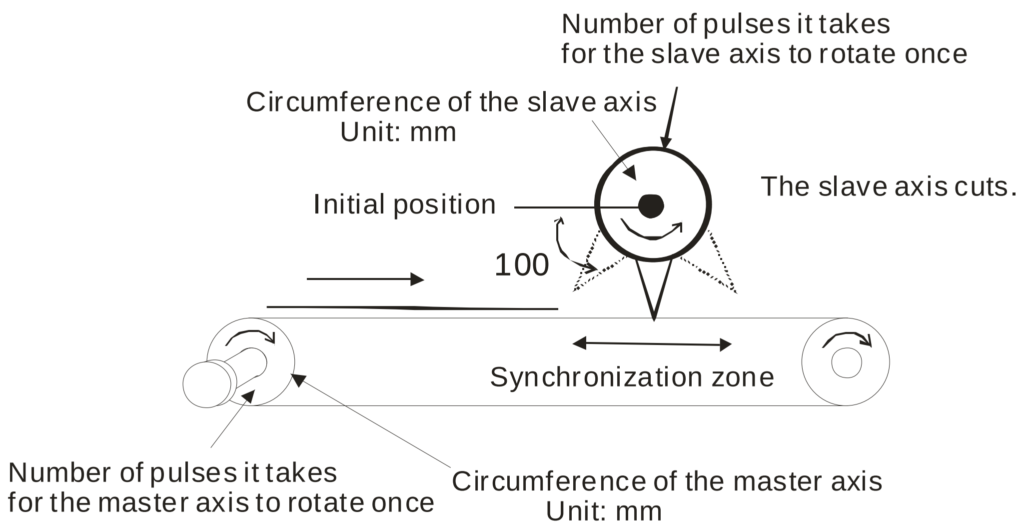
After the motion control function block T_CamCurve is executed, the motion control function block T_CamCurveUpdate can be used. If users want to modify a rotary cut curve, they can create a new rotary cut curve by means of the motion control function block T_CamCurve, and then update the rotary cut curve by means of the motion control function block T_CamCurveUpdate.



| Input pin | | | | |
|---|---|---|---|---|
| **Name** | **Function** | **Data type** | **Setting value** | **Description** |
| Axis | Motion axis number | WORD | 1~16 | Slave axis number |
| Execute | When there is a transition in the Execute input pin's signal from low to high, the update of a cam curve is enabled. | BOOL | True/False | Starting the update of a cam curve |
| **State output pin** | | | | |
| **Name** | **Function** | **Data type** | **Output range** | **Description** |
| Done | The update of a cam curve is complete. | BOOL | True/False | The update of a cam curve is complete. |
| Busy | The motion control function block is being executed. | BOOL | True/False | A cam curve is being updated. |
| Error | An error occurs in the motion control function block. | BOOL | True/False | An error occurs when a cam curve is updated. |

## 8.4.4 Example

**【Example】**



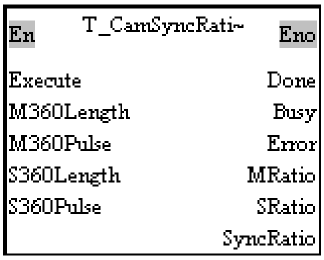
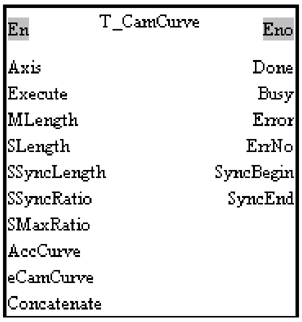
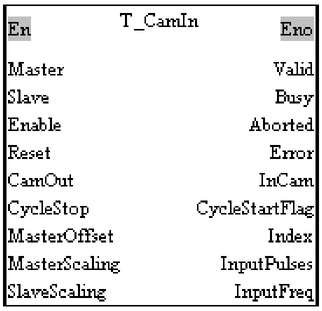

In a basic framework to which rotary cut is applied, the first axis is a slave axis, and the second axis is a master axis.

**【Control requirement】**

1. Using the motion control function block T_CamCurve to automatically create a cam curve
2. The electronic gear ratio for the cutting roller used is 10,000 pulses per revolution, and the electronic gear ratio for the feeding roller used is 10,000 pulses per revolution.
3. Related parameters

- The length of a cutting is 500 mm.
- The circumference of the cutting roller used is 60π mm.
- The circumference of the feeding roller used is 100π mm.
- The speed of the feeding roller used is 1,000 Hz.

4. Motion axes
   First axis: Slave axis
   Second axis: Master axis
5. Function blocks

| Name | Motion control function block | Function |
|---|---|---|
| Calculating a synchronization ratio |  | The motion control function block is used to calculate the value of the SyncRatio input pin when a cam curve is created. |
| Creating a cam curve |  | Automatically creating a cam curve |
| Electronic cam motion |  | Starting/Stopping electronic cam motion |

**【Elements】**

| Device in a PLC | | Description |
|---|---|---|
| **Contacts in PMSoft** | M0 | Calculating a synchronization ratio |
| | M10 | Starting the creation of a rotary cut curve |
| | M50 | The creation of a rotary cut curve is complete. |
| | M70 | Starting/Stopping electronic cam motion |
| | M72 | Not meshing with the master axis specified |
| | M200 | Electronic cam motion started |

**【Control program】**

1. Using the motion control function block T_CamCurve to automatically create an electronic cam curve
   - Users have to set the length of a cutting. The length of a cutting is 500 mm.

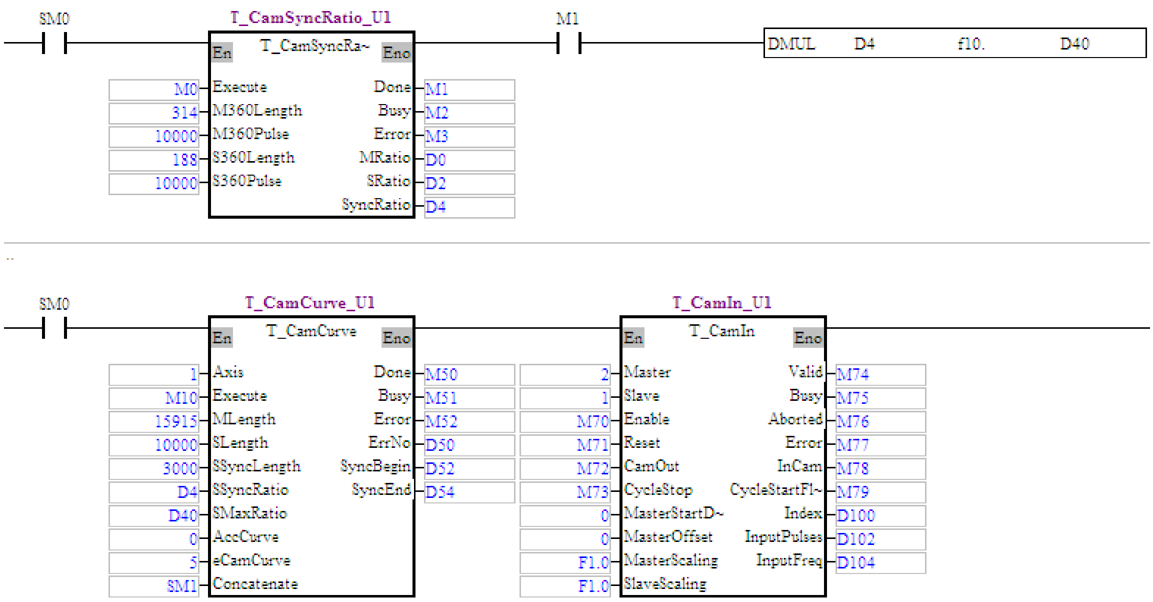
     $$500\text{ mm} \xrightarrow{\text{Conversion}} 500 \times \frac{1000}{100\pi} = 15{,}915$$

   - The circumference of a slave axis specified is the number of pulses it takes for the slave axis to rotate once. In this example, the circumference of the slave axis specified is 10,000 pulses.
   - The distance for which a slave axis is synchronized with a master axis is thirty percent of the circumference of the slave axis.

     $$10{,}000 \times 30\% = 3{,}000$$

   - Calculating a synchronization ratio
     - The circumference of the master axis specified is 314 mm.
     - The number of pulses it takes for the master axis specified to rotate once is 10,000.
     - The circumference of the slave axis specified is 188 mm.
     - The number of pulses it takes for the slave axis specified to rotate once is 10,000.
   - Maximum ratio of the speed of the slave axis specified to the speed of the master axis specified: Ten times the synchronization ratio of the speed of the slave axis specified to the speed of the master axis specified.
   - Acceleration curve: Uniform curve
   - Rotary cut curve: rightCAM
   - The Concatenate input pin is set to False.

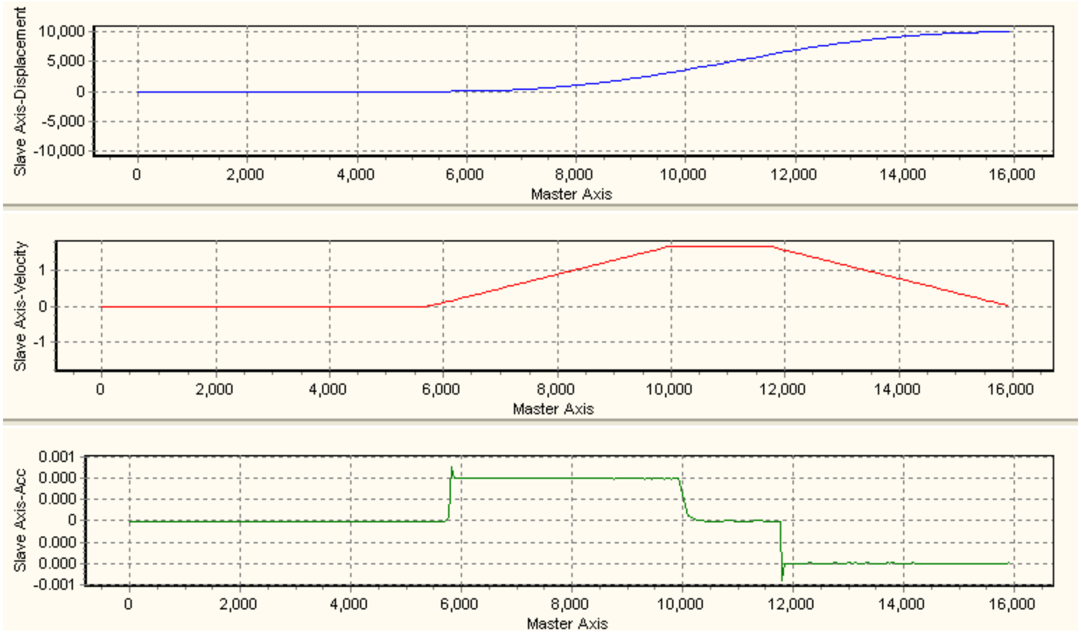
   After the input pins in the motion control function block T_CamCurve are set in accordance with the setting described above, a cam curve can be created.
2. Main program in PMSoft




**【Steps】**

1. Open a **CAM Chart** window in PMSoft, and then type 300 in the **Resolution** box.
2. Download the program created to an AH500 series motion control module, and then execute the program.

3. After M0 is set to True, a synchronization ratio will be calculated.
4. After M10 is set to True, a cam curve will be created. The synchronization zone gotten is in the range of 9,927 pulses to 11,723 pulses.
5. Update the cam data in the AH500 series motion control module, and then check whether the cam chart created is correct.
   - The distance for which the master axis specified moves is the same as the length set, that is, the distance for which the master axis specified moves is 15,915 pulses.
   - The synchronization ratio of the speed of the slave axis specified to the speed of the master axis specified is the same as the value of the SyncRatio output pin in the motion control function block T_CamSyncRatio.
   - The distance for which the slave axis specified is synchronized with the master axis specified is 3,000 pulses.
   - The synchronization zone gotten is in the range of 9,927 pulses to 11,723 pulses. The distance for which the master axis specified is synchronized with the slave axis specified is 1,796 pulses.
   - Distance for which the slave axis specified is synchronized with the master axis specified = Distance for which the master axis specified is synchronized with the slave axis specified×Synchronization ratio, that is, $1,796 \times 1.6702 = 3000$.




   - After M70 is set to True, electronic cam motion will be started.
   - After M72 is set to True, the slave axis specified will not mesh with the master axis specified.

# Chapter 9 Multiaxial Interpolation

## Table of Contents

## 9.1 Introduction of Multiaxial Interpolation

AH500 series motion control modules support multiaxial interpolation. There are two types of interpolation.

- Users can write G-codes in the motion subroutines Ox0~Ox99 to execute simple CNC.
- Users can execute interpolation by means of the instruction TO.

## 9.2 Table of O Pointers / M-codes and Table of G-codes

| Instruction code | Function | Description | Applicable model | | | |
|---|---|---|---|---|---|---|
| | | | 20MC | 10PM | 15PM | 05PM |
| O | Program pointer | Main program: O100<br>Motion subroutines: Ox0~Ox99 | ✓ | ✓ | ✓ | ✓ |
| M | M-code | M0~M65535<br>M102: End of the main program O100<br>M2: End of a motion subroutine | ✓ | ✓ | ✓ | ✓ |

📖 G-codes

| G-code | Function | Applicable model | | | |
|---|---|---|---|---|---|
| | | 20MC | 10PM | 15PM | 05PM |
| 00 | Rapid positioning (three axes) | ✓ | ✓ | ✓ | ✓ |
| 01 | Linear interpolation (two axes) | ✓ | ✓ | ✓ | ✓ |
| 01 | Linear interpolation (three axes) | ✓ | ✓ | ✓ | ✓ |
| 02 | Circular interpolation, clockwise (arc center) | ✓ | ✓ | ✓ | ✓ |
| 02 | Helical interpolation, clockwise (arc center) | ✓ | ✓ | ✓ | ✓ |
| 02 | Circular interpolation, clockwise (radius) | ✓ | ✓ | ✓ | ✓ |
| 02 | Helical interpolation, clockwise (radius) | ✓ | ✓ | ✓ | ✓ |
| 03 | Circular interpolation, counterclockwise (arc center) | ✓ | ✓ | ✓ | ✓ |
| 03 | Helical interpolation, counterclockwise (arc center) | ✓ | ✓ | ✓ | ✓ |
| 03 | Circular interpolation, counterclockwise (radius) | ✓ | ✓ | ✓ | ✓ |
| 03 | Helical interpolation, counterclockwise (radius) | ✓ | ✓ | ✓ | ✓ |
| 04 | Dwell | ✓ | ✓ | ✓ | ✓ |
| 17 | XY plane selection | ✓ | ✓ | ✓ | ✓ |
| 18 | ZX plane selection | ✓ | ✓ | ✓ | ✓ |
| 19 | YZ plane selection | ✓ | ✓ | ✓ | ✓ |
| 90 | Absolute programming | ✓ | ✓ | ✓ | ✓ |
| 91 | Incremental programming | ✓ | ✓ | ✓ | ✓ |

Additional remark: 05PM=AH05PM-5A; 10PM=AH10PM-5A; 15PM=AH15PM-5A; 20MC=AH20MC-5A

## 9.3 Composition of a G-code

- A G-code instruction is composed of an instruction name and operands.

| Instruction name | | Function which is executed |
|---|---|---|
| Operand | Indication of a function | Parameter mark (X, Y, Z, I, J, K, R, F) |
| | Setting of a parameter | Value of a parameter |

- Users must type parameter marks.

- If the value of a parameter is a contant, it must be a 32-bit arabic integer.
  Example 1: G00 X**100** Y**100**
  Example 2: G00 X**100.0** Y**100.0**
- If the value of a parameter is a floating-point constant, it will be converted into an integer after it is multiplied by 1000. Only three decimal places are supported.
  Example: G00 X**100.123** Y**100.45678** is converted into G00 X**100123** Y**100456**.
- The value of a parameter can be a 16-bit D/W register, or a 32-bit DD/WW register.
  Examples:
  G0 XD11 YDD20 ZWW25;
  G01 XDD30 YD40 ZW10 F400;
  G1 X100.0 Y25.0 FD50;
  G02 XD60 Y50.0 ID100 JDD80;
  G03 YDD90 RD70 F300.0

◆ Size of a G-code

- G00, G01, G02, and G03 individually occupy two steps in a program. The other G-codes individaully occupy one step in a program.
- If the value of a paramter is an arabic integer, it will occupy three steps in a program. If the value of a parameter is a D/W register, it will occupy 1 step in a program. If the value of a parameter is a DD/WW register, it will occupy two steps in a program.

◆ Format of a G-code instruction

| G-code | Instruction code | Operand | Function |
|---|---|---|---|
| 0000 | G00 | X $P_1$ Y $P_2$ Z $P_3$ A $P_4$ B $P_5$ C $P_6$ | Rapid positioning |

| Device | K | 16# | F | D | DD | W | WW |
|---|---|---|---|---|---|---|---|
| $P_1$ | ● | ● | | ● | ● | ● | ● |
| $P_2$ | ● | ● | | ● | ● | ● | ● |
| $P_3$ | ● | ● | | ● | ● | ● | ● |
| $P_4$ | ● | ● | | ● | ● | ● | ● |
| $P_5$ | ● | ● | | ● | ● | ● | ● |
| $P_6$ | ● | ● | | ● | ● | ● | ● |

① G-code
② Parameter mark
③ Value of a parameter
④ Devices which can be used

◆ Typing a G-code instruction

Some G-code instructions are composed of instruction names, e.g. G90 and G91. Most G-code instructions are composed of instruction names and operands. No conditional contact precedes a G-code.

◆ Usage of a G-code

- Users can put several functions in a line.
  Example: G91 G01 X100.0 Y300.0 F500.0 M8 G04 X4.5;
- If G00, G01, G02, and G03 are in the same line, the last G-code will be executed.
  Examples:
  G02 G00 G03 G01 X100.0 Y300.0 F500.0;

=>G01 X100.0 Y300.0 F500.0;
G02 G00 X100.0 G03 G01 Y300.0 F500.0;
=>G01 Y300.0 F500.0;

- If G00 is used, users do not have to set a velocity.
  Example: G00 X100.2 Y500.0;
  The speeds at which the axes move are the maximum speeds set in the AH500 series motion control module used.
- G00 and G01 can be extended to the next line.

| | |
|---|---|
| N0000 G00 X500.0 Y125.0; | |
| N0001 X-400.0 Y-500.0; | =>G00 X-400.0 Y-500.0; |
| N0002 G01 X100.0 Y25.0 F200.0; | |
| N0003 X-200.0 Y50.0; | =>G01 X-200.0 Y50.0 F200.0; |

- The speed parameter F for G01/G02/G03 can be extended to the next line. (Users must specify the value of the speed parameter F in the first line.)

| | |
|---|---|
| N0000 G01 X500.0 Y125.0 F200.0; | |
| N0001 G03 X-40.0 Y-50.0 R100.0; | =>G03 X-40.0 Y-50.0 R100.0 F200.0; |
| N0002 G02 X100.0 Y25.0 I400.5 F200.0; | |
| N0003 G01 X-200.0 Y50.0; | =>G01 X-200.0 Y50.0 F200.0; |

- G90 and G91 have high priority over the other G-codes.

| | |
|---|---|
| G90 G01 X100.0 Y300.0 F500.0; | =>G90 G01 X100.0 Y300.0 F500.0; |
| G01 G90 X100.0 Y300.0 F500.0; | =>G90 G01 X100.0 Y300.0 F500.0; |

- Whether there are spaces in a program code, the program code can be identified.

| | |
|---|---|
| G01G91X500.0  Y125.0F200.0; | =>G01 G91 X500.0 Y125.0 F200.0; |

- Coordinates and speeds are converted into 32-bit values.

| | |
|---|---|
| G01 X-125.5 F200.0; | =>G01 X-125500 F200000; |

- If the value of a parameter is a floating-point constant, it will be converted into an integer after it is multiplied by 1000.

| | |
|---|---|
| G01 X100 Y-125.5 F200.0; | =>G01 X100 Y-125500 F200000; |

- Parameter X for G04: A second is a unit of measurement for dwell duration.
  Parameter P for G04: A millisecond is a unit of measurement for dwell duration. In the example below, the system used automatically ignores 9 in P2509.
  G04 X4.5  (Dwell duration: 4.5 seconds)
  G04 X5  (Dwell duration: 5 seconds)
  G04 P4500  (Dwell duration: 4.5 seconds)
  G04 P2509  (Dwell duration: 2.5 seconds)
- The G-codes not supported are ignored and not read.

| | |
|---|---|
| G21G54G01 X-125.5  F200.0; | =>G01 X-125500 F200000; |
| G43G87G96 X250.5 F200.0; | =>G01 X250500 F200000; |

- The writing of instructions conforms to the writing of general G-codes. Users can arrange the parameter marks used in any order.

| | |
|---|---|
| G0  X4.5 Z40.0 Y30.5 F200.2; | =>G00 X4.500 Y30.500 Z40.000 F200.200; |
| Z100.5Y400.0X300.0; | =>G00 X300.000 Y400.000 Z100.500; |
| G1xd100zd300y200.45 fd400; | =>G01 XD100 Y200.450 ZD300 FD400; |
| G3 ZD100 I200.0F50.60XD300 m80; | =>G03 XD300 ZD100 I200.000 F50.600 M80; |
| G03 yD100 x9999.9Z200.0r777.7  Fd800; | =>G03 X9999.900 YD100 Z200.000 R777.700 FD800; |

9

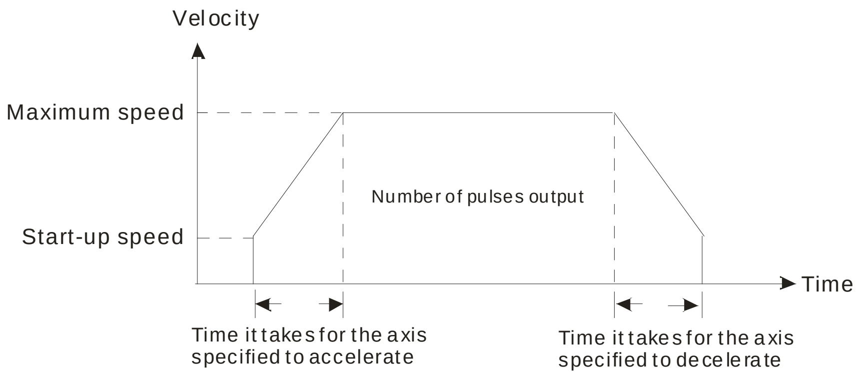
## 9.4 Descriptions of G-code Instructions

| G-code | Instruction code | Operand | Function |
|---|---|---|---|
| 0000 | G00 | X P1 Y P2 Z P3 A P4 B P5 C P6 | Rapid positioning |

| Device | K | 16# | F | D | DD | W | WW |
|---|---|---|---|---|---|---|---|
| $P_1$ | ● | ● | | ● | ● | ● | ● |
| $P_2$ | ● | ● | | ● | ● | ● | ● |
| $P_3$ | ● | ● | | ● | ● | ● | ● |
| $P_4$ | ● | ● | | ● | ● | ● | ● |
| $P_5$ | ● | ● | | ● | ● | ● | ● |
| $P_6$ | ● | ● | | ● | ● | ● | ● |

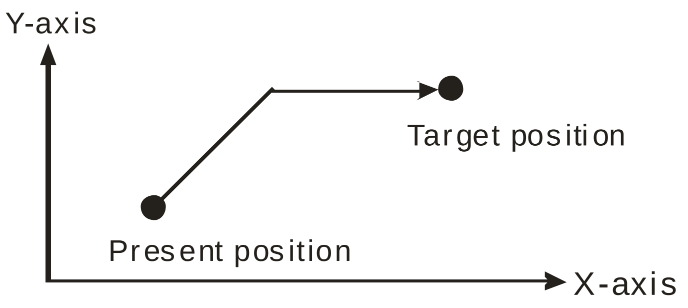
**Description:**

- **$P_1$**: Target position of an x-axis; **$P_2$**: Target position of a y-axis; **$P_3$**: Target position of a z-axis; **$P_4$**: Target position of an A-axis; **$P_5$**: Target position of a B-axis; **$P_6$**: Target position of a C-axis
- If the value of a parameter is not a floating-point value, it must be in the range of -2,147,483,648 to 2,147,483,647. If the value of a parameter is a floating-point value, it must be in the range of-2,147,483.648 to 2,147,483.647.
- Users do not need to set the speeds at which the axes used moves. The speed at which an axis moves is its maximum speed.
- Users only need to specify one target position or more than one target position.
- Operation




  - Users can set the time it takes for an axis specified to accelerate/decelerate and the start-up speed of the axis by means of special data registers.
  - The time it takes for an axis specified to accelerate/decelerate is proportional to the maximum speed of the axis.

- Example: G00 X250.0 Y150
  The instruction moves two axes from the present position (50.0, 50.0) to the target position (250.0, 150.0). If G90 precedes the instruction, the target position is an absolute position. If G91 precedes the instruction, the target position is a relative position. The speeds at which the x-axis and the y-axis move are their maximum speeds.
  Path:

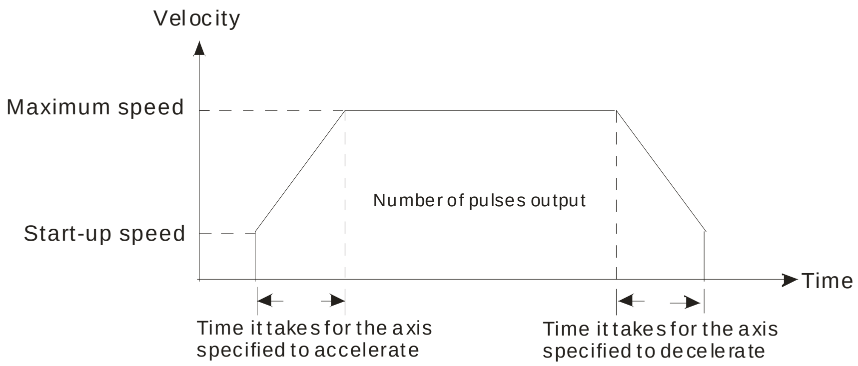


| G-code | Instruction code | Operand | Function |
|---|---|---|---|
| 0001 | G01 | X P1 Y P2 Z P3 A P4<br>B P5 C P6 F V | Linear interpolation<br>(The distance remaining can be considered.) |

| Device | K | 16# | F | D | DD | W | WW |
|---|---|---|---|---|---|---|---|
| $P_1$ | ● | ● | | ● | ● | ● | ● |
| $P_2$ | ● | ● | | ● | ● | ● | ● |
| $P_3$ | ● | ● | | ● | ● | ● | ● |
| $P_4$ | ● | ● | | ● | ● | ● | ● |
| $P_5$ | ● | ● | | ● | ● | ● | ● |
| $P_6$ | ● | ● | | ● | ● | ● | ● |
| V | ● | ● | | ● | ● | ● | ● |

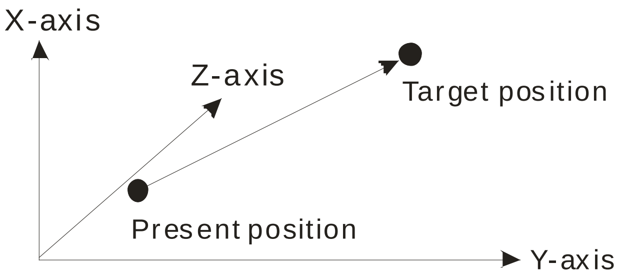
**Description:**

- **$P_1$**: Target position of an x-axis; **$P_2$**: Target position of a y-axis; **$P_3$**: Target position of a z-axis; **$P_4$**: Target position of an A-axis; **$P_5$**: Target position of a B-axis; **$P_6$**: Target position of a C axis; **V**: Speed of linear interpolation
- If the value of **$P_1$/$P_2$/$P_3$** is not a floating-point value, it must be in the range of -2,147,483,648 to 2,147,483,647, and the value of **V** must be in the range of 0 to 500,000. If the value of **$P_1$/$P_2$/$P_3$** is a floating-point value, it must be in the range of-2,147,483.648 to 2,147,483.647, and the value of **V** must be in the range of 0 to 500.0.
- If users specify a speed for G01, the speed of interpolation will be the speed specified. If no speed is specified for the instruction, the speed of interpolation will be the speed specified for G01/G02/G03 which precedes G01.
  - **V**: Maximum speed of interpolation
  - Users only need to specify one target position or more than one target position.
  - Operation




  - Users can set the time it takes for an axis specified to accelerate/decelerate and the start-up speed of the axis by means of special data registers.
  - The time it takes for an axis specified to accelerate/decelerate is proportional to the maximum speed of the axis.

- Example: G01 X200.0 Y400.0 Z250.0 F400.0
  The instruction moves three axes from the present position (0, 10.0, 100.0) to the target position (200.0, 400.0, 250.0). If G90 precedes the instruction, the target position is an absolute position. If G91 precedes the instruction, the target position is a relative position. The speed at which the x-axis, the y-axis, and the z-axis move is 400 kHz.
  Path:

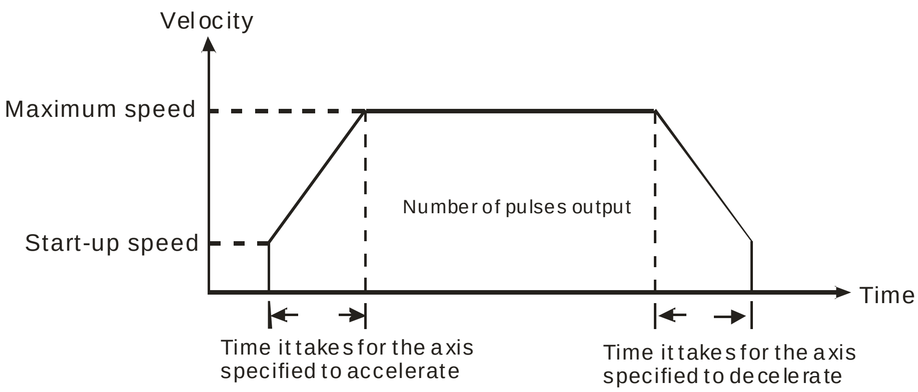


| G-code | Instruction code | Operand | Function |
|---|---|---|---|
| 0002<br>0003 | G02<br>G03 | X $P_1$ Y $P_2$ Z $P_3$ I $P_4$<br>J $P_5$ K $P_6$ F V | Clockwise circular/helical interpolation<br>Counterclockwise circular/helical interpolation<br>(arc center) |

| Device | K | 16# | F | D | DD | W | WW |
|---|---|---|---|---|---|---|---|
| $P_1$ | ● | ● | | ● | ● | ● | ● |
| $P_2$ | ● | ● | | ● | ● | ● | ● |
| $P_3$ | ● | ● | | ● | ● | ● | ● |
| $P_4$ | ● | ● | | ● | ● | ● | ● |
| $P_5$ | ● | ● | | ● | ● | ● | ● |
| $P_6$ | ● | ● | | ● | ● | ● | ● |
| V | ● | ● | | ● | ● | ● | ● |

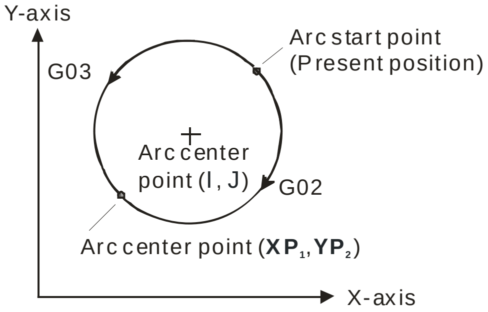
**Description:**

- **$P_1$**: Target position of an x-axis; **$P_2$**: Target position of a y-axis; **$P_3$**: Target position of a z-axis; **$P_4$**: Vector from the present position of an x-axis to an arc center; **$P_5$**: Vector from the present position of an y-axis to an arc center; **$P_6$**: Vector from the present position of an z-axis to an arc center; **V**: Speed of circular/helical interpolation
- **$P_4$**, **$P_5$** and **$P_6$**: Vectors from the present positions of an x-axis, a y-axis, and a z-axis to an arc center
- If the value of **$P_1$/$P_2$/$P_3$/$P_4$/$P_5$/$P_6$** is not a floating-point value, it must be in the range of -2,147,483,648 to 2,147,483,647, and the value of **V** must be in the range of 0 to 500,000. If the value of **$P_1$/$P_2$/$P_3$/$P_4$/$P_5$/$P_6$** is a floating-point value, it must be in the range of-2,147,483.648 to 2,147,483.647, and the value of **V** must be in the range of 0 to 500.0.
- If users specify a speed for G02/G03, the speed of circular/helical interpolation will be the speed specified. If no speed is specified for the instruction, the speed of circular/helical interpolation will be the speed specified for G01/G02/G03 which precedes G02/G03.
  - **V**: Maximum speed of circular/helical interpolation
  - Operation

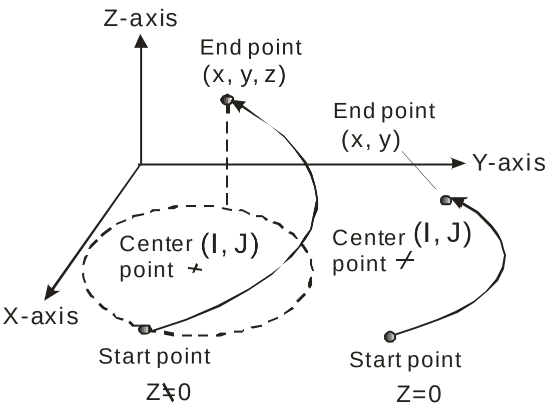



  - Users can set the time it takes for an axis specified to accelerate/decelerate and the start-up speed of the axis by means of special data registers.
  - The time it takes for an axis specified to accelerate/decelerate is proportional to the maximum speed of the axis.

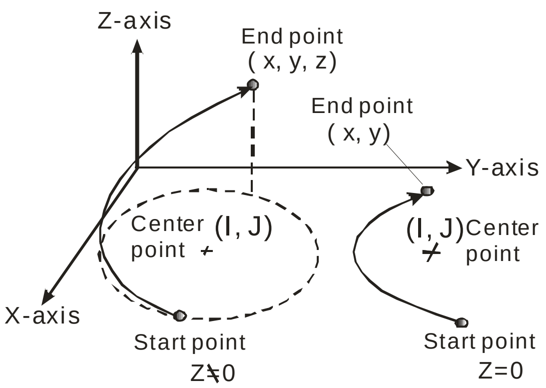
- Circular interpolation: Two axes which are perpendicular to each other are used. G17, G18, or G19 is used to control circular interpolation.




- Helical interpolation: Three axes which are perpendicular to one another are used. They move synchronously. Helical interpolation is the extension of circular interpolation. If a helical interpolation instruction is used, and the change of height is zero, circular interpolation will be executed.
  - If G17 is used, and the position of a z-axis does not change, helical interpolation will be equivalent to circular interpolation.

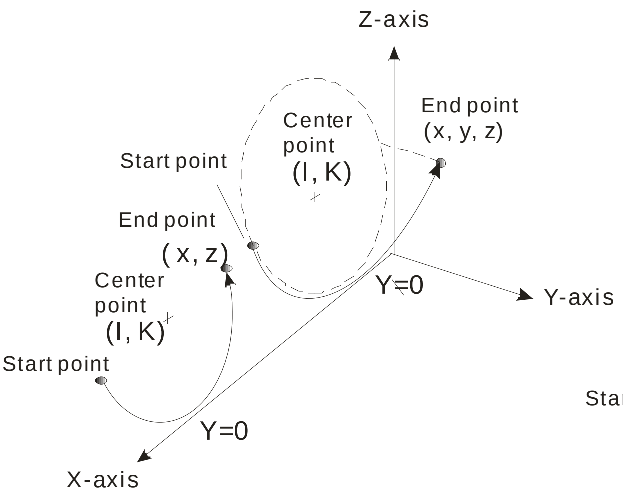



G03 (Counterclockwise interpolation)

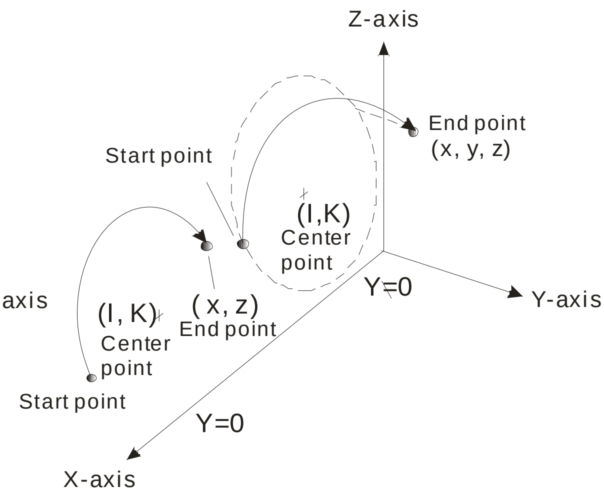



G02 (Clockwise interpolation)

  - If G18 is used, and the postion of a y-axis does not change, helical interpolation will be equivalent to circular interpolation.

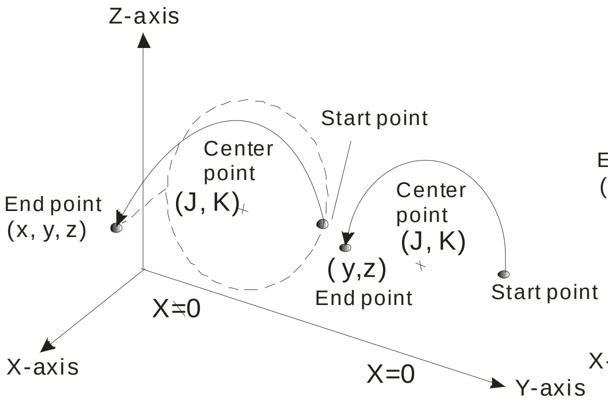



G03 (Counterclockwise interpolation)

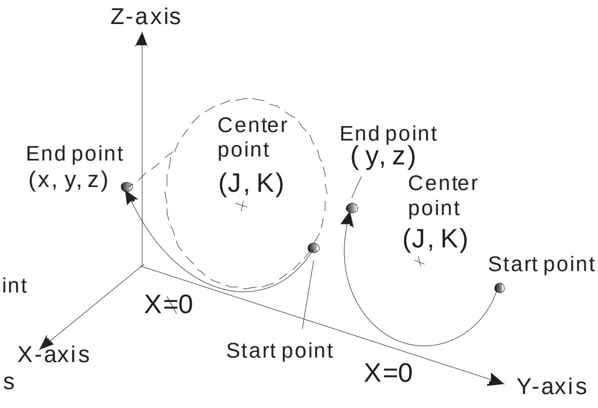



G02 (Clockwise interpolation)

- If G19 is used, and the postion of an x-axis does not change, helical interpolation will be equivalent to circular interpolation.




G03 (Counterclockwise interpolation)




G02 (Clockwise interpolation)

- Principles of writing an instruction: (1) Users have to specify a target position, and an arc center. They do not have to specify the speed of interpolation. (2) If there is no vector from the present position of an axis to its target position, users do not need to specify the target position of the axis. (3) If there is no vector from the present position of an axis to an arc center, users do not need to specify a vector. The circular/helical instrctions listed below are available.

| NO. | G-code | Combination of operands | G17 | G18 | G19 |
|---|---|---|---|---|---|
| 1 | G02/G03 | X $P_1$ I $P_3$ | ✓ | ✓ | |
| 2 | | X $P_1$ I $P_3$ F V | ✓ | ✓ | |
| 3 | | X $P_1$ J $P_4$ | ✓ | | |
| 4 | | X $P_1$ J $P_4$ F V | ✓ | | |
| 5 | | X $P_1$ I $P_3$ J $P_4$ | ✓ | | |
| 6 | | X $P_1$ I $P_3$ J $P_4$ F V | ✓ | | |
| 7 | | Y $P_2$ I $P_3$ | ✓ | | |
| 8 | | Y $P_2$ I $P_3$ F V | ✓ | | |
| 9 | | Y $P_2$ J $P_4$ | ✓ | | ✓ |
| 10 | | Y $P_2$ J $P_4$ F V | ✓ | | ✓ |
| 11 | | Y $P_2$ I $P_3$ J $P_4$ | ✓ | | |
| 12 | | Y $P_2$ I $P_3$ J $P_4$ F V | ✓ | | |
| 13 | | X $P_1$ Y $P_2$ I $P_3$ | ✓ | ✓ | |
| 14 | | X $P_1$ Y $P_2$ I $P_3$ F V | ✓ | ✓ | |
| 15 | | X $P_1$ Y $P_2$ J $P_4$ | ✓ | | ✓ |
| 16 | | X $P_1$ Y $P_2$ J $P_4$ F V | ✓ | | ✓ |
| 17 | | X $P_1$ Y $P_2$ I $P_3$ J $P_4$ | ✓ | | |
| 18 | | X $P_1$ Y $P_2$ I $P_3$ J $P_4$ F V | ✓ | | |
| 19 | | X $P_1$ Z $P_3$ I $P_3$ | ✓ | ✓ | |
| 20 | | X $P_1$ Z $P_3$ I $P_3$ F V | ✓ | ✓ | |

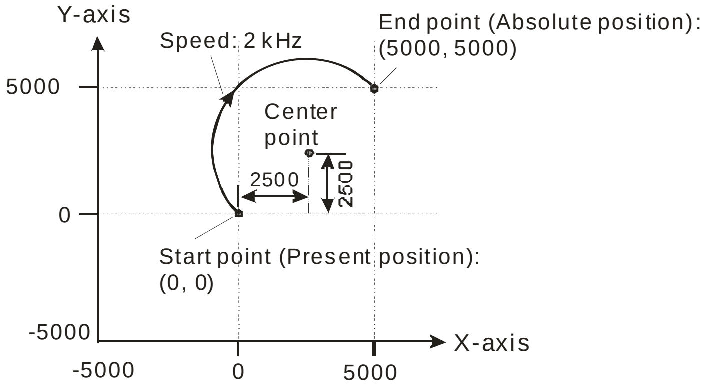
9

| NO. | G-code | Combination of operands | G17 | G18 | G19 |
|---|---|---|---|---|---|
| 21 | G02/G03 | X $P_1$ Z $P_3$ J $P_4$ | ✓ | | ✓ |
| 22 | | X $P_1$ Z $P_3$ J $P_4$ F $V$ | ✓ | | ✓ |
| 23 | | X $P_1$ Z $P_3$ I $P_3$ J $P_4$ | ✓ | | |
| 24 | | X $P_1$ Z $P_3$ I $P_3$ J $P_4$ F $V$ | ✓ | | |
| 25 | | Y $P_2$ Z $P_3$ I $P_3$ | ✓ | ✓ | |
| 26 | | Y $P_2$ Z $P_3$ I $P_3$ F $V$ | ✓ | ✓ | |
| 27 | | Y $P_2$ Z $P_3$ J $P_4$ | ✓ | | ✓ |
| 28 | | Y $P_2$ Z $P_3$ J $P_4$ F $V$ | ✓ | | ✓ |
| 29 | | Y $P_2$ Z $P_3$ I $P_3$ J $P_4$ | ✓ | | |
| 30 | | Y $P_2$ Z $P_3$ I $P_3$ J $P_4$ F $V$ | ✓ | | |
| 31 | | X $P_1$ Y $P_2$ Z $P_3$ I $P_3$ | ✓ | ✓ | |
| 32 | | X $P_1$ Y $P_2$ Z $P_3$ I $P_3$ F $V$ | ✓ | ✓ | |
| 33 | | X $P_1$ Y $P_2$ Z $P_3$ J $P_4$ | ✓ | | ✓ |
| 34 | | X $P_1$ Y $P_2$ Z $P_3$ J $P_4$ F $V$ | ✓ | | ✓ |
| 35 | | X $P_1$ Y $P_2$ Z $P_3$ I $P_3$ J $P_4$ | ✓ | | |
| 36 | | X $P_1$ Y $P_2$ Z $P_3$ I $P_3$ J $P_4$ F $V$ | ✓ | | |
| 37 | | X $P_1$ K $P_6$ | | ✓ | |
| 38 | | X $P_1$ K $P_6$ F $V$ | | ✓ | |
| 39 | | X $P_1$ I $P_3$ K $P_6$ | | ✓ | |
| 40 | | X $P_1$ I $P_3$ K $P_6$ F $V$ | | ✓ | |
| 41 | | Z $P_3$ I $P_3$ | | ✓ | |
| 42 | | Z $P_3$ I $P_3$ F $V$ | | ✓ | |
| 43 | | Z $P_3$ K $P_6$ | | ✓ | ✓ |
| 44 | | Z $P_3$ K $P_6$ F $V$ | | ✓ | ✓ |
| 45 | | Z $P_3$ I $P_3$ K $P_6$ | | ✓ | |
| 46 | | Z $P_3$ I $P_3$ K $P_6$ F $V$ | | ✓ | |
| 47 | | X $P_1$ Z $P_3$ K $P_6$ | | ✓ | ✓ |
| 48 | | X $P_1$ Z $P_3$ K $P_6$ F $V$ | | ✓ | ✓ |
| 49 | | X $P_1$ Z $P_3$ I $P_3$ K $P_6$ | | ✓ | |
| 50 | | X $P_1$ Z $P_3$ I $P_3$ K $P_6$ F $V$ | | ✓ | |
| 51 | | X $P_1$ Y $P_2$ K $P_6$ | | ✓ | ✓ |
| 52 | | X $P_1$ Y $P_2$ K $P_6$ F $V$ | | ✓ | ✓ |
| 53 | | X $P_1$ Y $P_2$ I $P_3$ K $P_6$ | | ✓ | |
| 54 | | X $P_1$ Y $P_2$ I $P_3$ K $P_6$ F $V$ | | ✓ | |
| 55 | | Y $P_2$ Z $P_3$ K $P_6$ | | ✓ | ✓ |
| 56 | | Y $P_2$ Z $P_3$ K $P_6$ F $V$ | | ✓ | ✓ |
| 57 | | Y $P_2$ Z $P_3$ I $P_3$ K $P_6$ | | ✓ | |
| 58 | | Y $P_2$ Z $P_3$ I $P_3$ K $P_6$ F $V$ | | ✓ | |

| NO. | G-code | Combination of operands | G17 | G18 | G19 |
|---|---|---|---|---|---|
| 59 | G02/G03 | X $P_1$ Y $P_2$ Z $P_3$ K $P_6$ | | ✓ | ✓ |
| 60 | | X $P_1$ Y $P_2$ Z $P_3$ K $P_6$ F $V$ | | ✓ | ✓ |
| 61 | | X $P_1$ Y $P_2$ Z $P_3$ I $P_3$ K $P_6$ | | ✓ | |
| 62 | | X $P_1$ Y $P_2$ Z $P_3$ I $P_3$ K $P_6$ F $V$ | | ✓ | |
| 63 | | Y $P_2$ K $P_6$ | | | ✓ |
| 64 | | Y $P_2$ K $P_6$ F $V$ | | | ✓ |
| 65 | | Y $P_2$ J $P_4$ K $P_6$ | | | ✓ |
| 66 | | Y $P_2$ J $P_4$ K $P_6$ F $V$ | | | ✓ |
| 67 | | Z $P_3$ J $P_4$ | | | ✓ |
| 68 | | Z $P_3$ J $P_4$ F $V$ | | | ✓ |
| 69 | | Z $P_3$ J $P_4$ K $P_6$ | | | ✓ |
| 70 | | Z $P_3$ J $P_4$ K $P_6$ F $V$ | | | ✓ |
| 71 | | Y $P_2$ Z $P_3$ J $P_4$ K $P_6$ | | | ✓ |
| 72 | | Y $P_2$ Z $P_3$ J $P_4$ K $P_6$ F $V$ | | | ✓ |
| 73 | | X $P_1$ Y $P_2$ J $P_4$ K $P_6$ | | | ✓ |
| 74 | | X $P_1$ Y $P_2$ J $P_4$ K $P_6$ F $V$ | | | ✓ |
| 75 | | X $P_1$ Z $P_3$ J $P_4$ K $P_6$ | | | ✓ |
| 76 | | X $P_1$ Z $P_3$ J $P_4$ K $P_6$ F $V$ | | | ✓ |
| 77 | | X $P_1$ Y $P_2$ Z $P_3$ J $P_4$ K $P_6$ | | | ✓ |
| 78 | | X $P_1$ Y $P_2$ Z $P_3$ J $P_4$ K $P_6$ F $V$ | | | ✓ |

- The path of circular interpolation can be a 360° arc. The path of helical interpolation which is viewed from the top can be a full circle.
- Example 1

  Absolute coordinates are set, and a clockwise circular interpolation instruction is used. The arc start point set is (0, 0), the arc end point set is (5000, 5000), and the vector from the arc start point to the arc center point set is (2500, 2500). The output speed set is 2000 Hz.



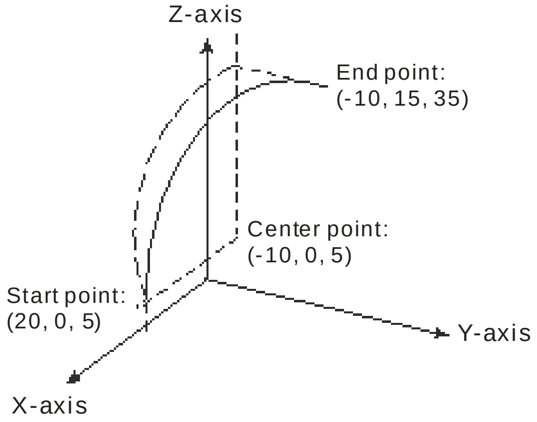

Program:

G90;

G02 X50.0 Y50.00 I2500 J2.5 F2000;

- Example 2

  Absolute coordinates are set. G18 and G02 are used. The arc end point set is (-10, 15, 35) and the arc cent point set is (-10, 0, 5). The output speed set is 2000 Hz.



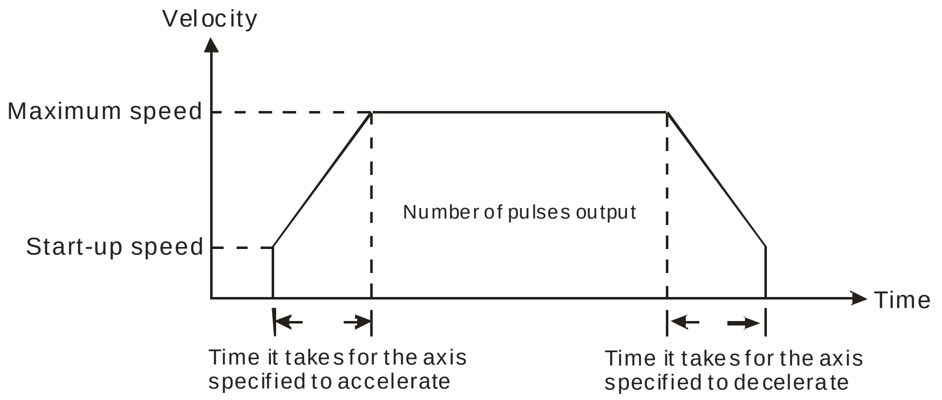

Program:
G90;
G18;
G02 X-10 Y15 Z35 I-30 J0 (omissible) K0 (omissible) F2000;

| G-code | Instruction code | Operand | Function |
|---|---|---|---|
| 0002<br>0003 | G02<br>G03 | X $P_1$ Y $P_2$ Z $P_3$ R L<br>F V | Clockwise circular/helical interpolation<br>Counterclockwise circular/helical interpolation<br>(radius) |

| Device | K | 16# | F | D | DD | W | WW |
|---|---|---|---|---|---|---|---|
| $P_1$ | ● | ● | | ● | ● | ● | ● |
| $P_2$ | ● | ● | | ● | ● | ● | ● |
| $P_3$ | ● | ● | | ● | ● | ● | ● |
| L | ● | ● | | ● | ● | ● | ● |
| V | ● | ● | | ● | ● | ● | ● |

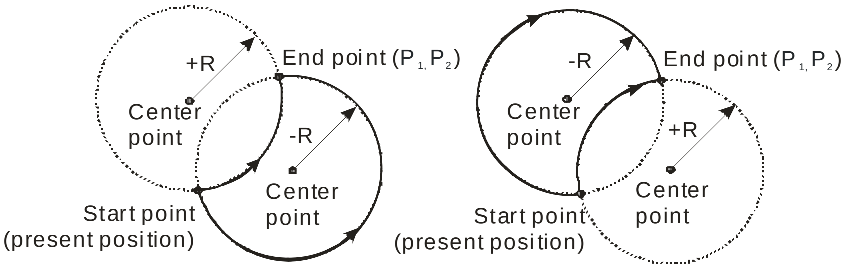
**Description:**

- **$P_1$**: Target position of an x-axis; **$P_2$**: Target position of a y-axis; **$P_3$**: Target position of a z-axis; **L**: Arc radius (If the angle subtended by an arc is less than 180°, the value of R is a positive value. If the angle subtended by an arc is greater than 180°, the value of R is a negative value.); **V**: Speed of circular/helical interpolation
- If the value of **$P_1$/$P_2$/$P_3$/L** is not a floating-point value, it must be in the range of -2,147,483,648 to 2,147,483,647, and the value of **V** must be in the range of 0 to 500,000. If the value of **$P_1$/$P_2$/$P_3$/L** is a floating-point value, it must be in the range of-2,147,483.648 to 2,147,483.647, and the value of **V** must be in the range of 0 to 500.0.
- **L**: If the angle subtended by an arc is less than 180°, the value of R is a positive value. If the angle subtended by an arc is greater than 180°, the value of R is a negative value.
- If users specify a speed for G02/G03, the speed of circular/helical interpolation will be the speed specified. If no speed is specified for G02/G03, the speed of circular/helical interpolation will be the speed specified for G01/G02/G03 which precedes G02/G03.
  - **V**: Maximum speed of circular/helical interpolation
  - Operation



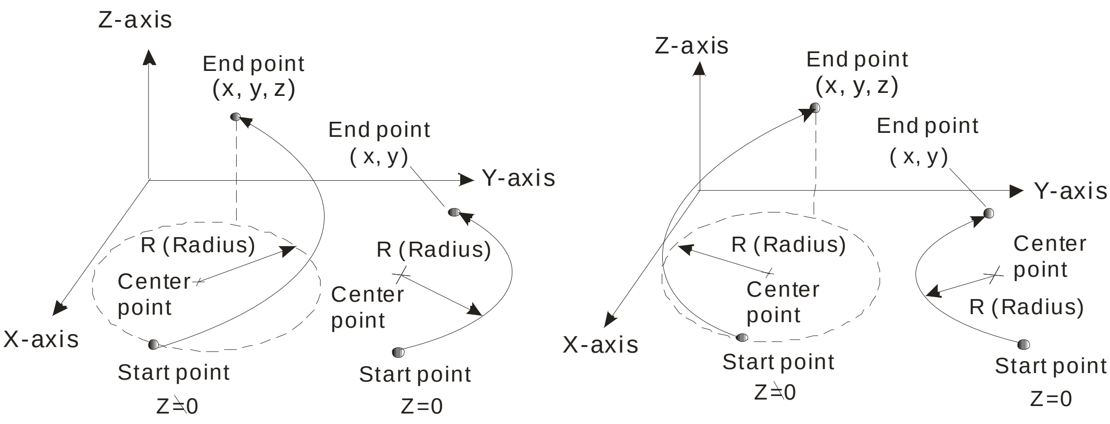

  - Users can set the time it takes for an axis specified to accelerate/decelerate and the start-up speed of the axis by means of special data registers.
  - The time it takes for an axis specified to accelerate/decelerate is proportional to the maximum speed of the axis.

- Circular interpolation: Two axes which are perpendicular to each other are used. G17, G18, or G19 is used to control circular interpolation.



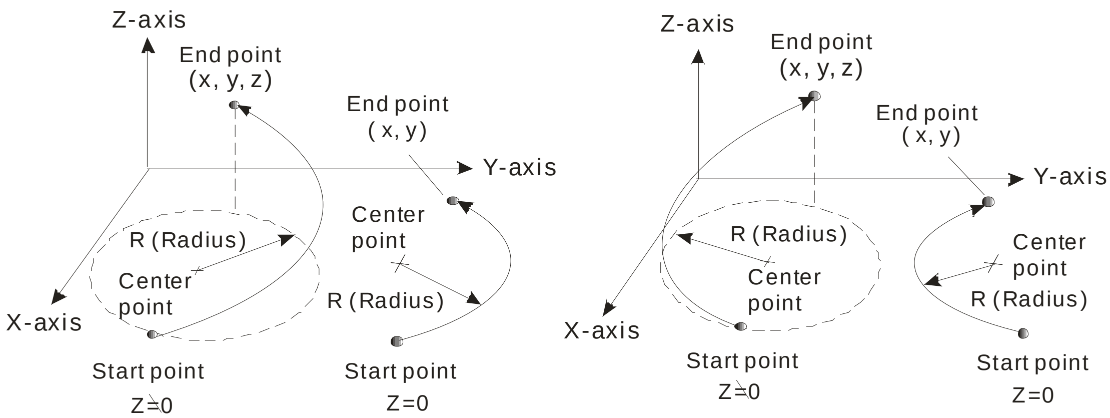

G03 (Counterclockwise interpolation) G02 (Clockwise interpolation)

- Helical interpolation: Three axes which are perpendicular to one another are used. They move synchronously. Helical interpolation is the extension of circular interpolation. If a helical interpolation instruction is used, and the change of height is zero, circular interpolation will be executed.
  - If G17 is used, and the position of a z-axis does not change, helical interpolation will be equivalent to circular interpolation.



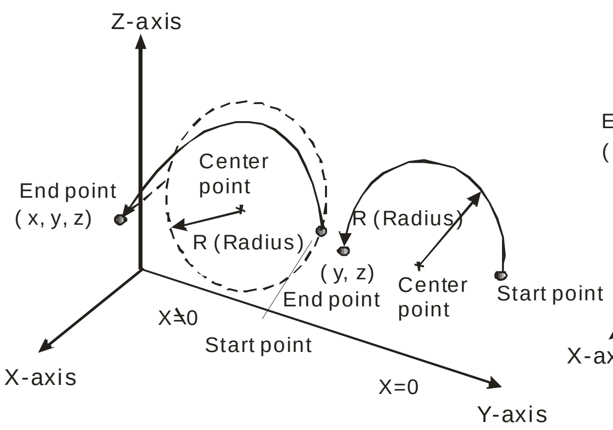

G03 (Counterclockwise interpolation) G02 (Clockwise interpolation)

  - If G18 is used, and the postion of a y-axis does not change, helical interpolation will be equivalent to circular interpolation.

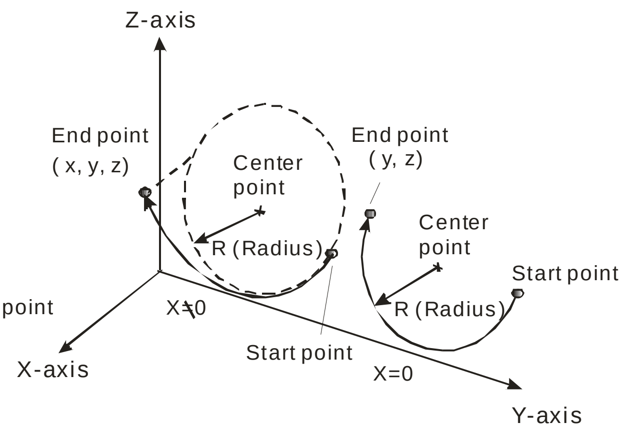



G03 (Counterclockwise interpolation) G02 (Clockwise interpolation)

- If G19 is used, and the postion of an x-axis does not change, helical interpolation will be equivalent to circular interpolation.




G03 (Counterclockwise interpolation)



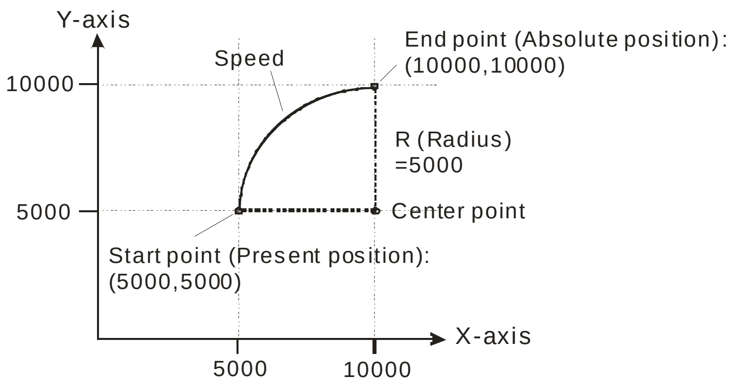

G02 (Clockwise interpolation)

- Principles of writing an instruction: (1) Users have to specify a target position, and an arc center. They do not have to specify the speed of interpolation. (2) If there is no vector from the present position of an axis to its target position, users do not need to specify the target position of the axis. The circular/helical instrctions listed below are available.

| NO. | G-code | Combination of operands | G17 | G18 | G19 |
|---|---|---|---|---|---|
| 1 | G02/G03 | X $P_1$ R L | ✓ | ✓ | |
| 2 | | X $P_1$ R L F V | ✓ | ✓ | |
| 3 | | Y $P_2$ R L | ✓ | | ✓ |
| 4 | | Y $P_2$ R L F V | ✓ | | ✓ |
| 5 | | X $P_1$ Y $P_2$ R L | ✓ | ✓ | ✓ |
| 6 | | X $P_1$ Y $P_2$ R L F V | ✓ | ✓ | ✓ |
| 7 | | X $P_1$ Z $P_3$ R L | ✓ | ✓ | ✓ |
| 8 | | X $P_1$ Z $P_3$ R L F V | ✓ | ✓ | ✓ |
| 9 | | Y $P_2$ Z $P_3$ R L | ✓ | ✓ | ✓ |
| 10 | | Y $P_2$ Z $P_3$ R L F V | ✓ | ✓ | ✓ |
| 11 | | X $P_1$ Z $P_3$ Y $P_2$ R L | ✓ | ✓ | ✓ |
| 12 | | X $P_1$ Z $P_3$ Y $P_2$ R L F V | ✓ | ✓ | ✓ |
| 13 | | Z $P_3$ R L | | ✓ | ✓ |
| 14 | | Z $P_3$ R L F V | | ✓ | ✓ |

- The path of circular interpolation can not be a 360° arc. The path of helical interpolation which is viewed from the top can not be a full circle.
- Example 1

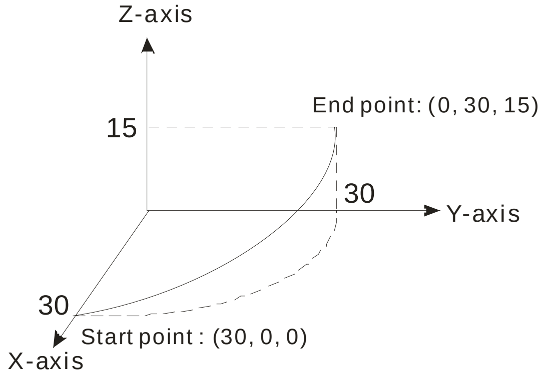
  Absolute coordinates are set, and G02 is used. The arc start point set is (5000, 5000), the arc end point set is (10000, 10000), and **L** is 5000. The angle subtended by the arc is less than 180°, and therefore the value of R is a positive value. The axes move at a speed of 1,000 per second.




Program:
G90;
G02 X10000 Y10000 R5000 F1000;

- Example 2

  Absolute coordinates are set, and G03 is used. The arc start point set is (30, 0, 0), the arc end point set is (0, 30, 15), and **L** is 30.0. The angle subtended by the arc is less than 180°, and therefore the value of R is a positive value. The axes move at a speed of 1,000 per second.



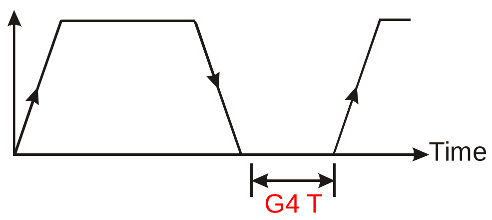

Program:
G90;
G3 X10000 Y10000 R5000 F1000;

| G-code | Instruction code | Operand | Function |
|---|---|---|---|
| 0004 | G04 | **X** T **/ P** T | Dwell |

| Device | K | 16# | F | D | DD | W | WW |
|---|---|---|---|---|---|---|---|
| **T** | ● | | ● | ● | ● | ● | ● |

**Description:**

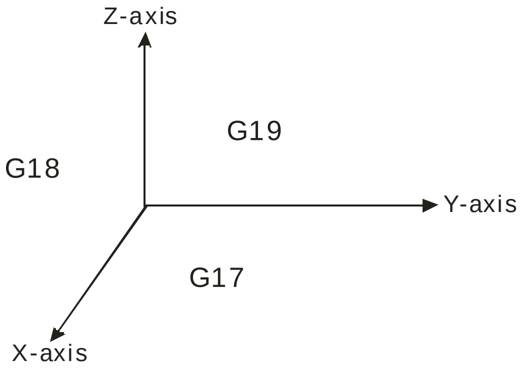
- **T**: Dwell time value
- If the operand X is used, a second is a unit of measurement for dwell duration. For example, the dwell period set is one second if G4 X1 is used, and the dwell period set is 2.5 seconds if G4 X2.5 is used.
- If the operand X is used, a millisecond is a unit of measurement for dwell duration. For example, the dwell period set 0.1 seconds if G4 P100 is used, and the dwell priod set is 4.5 seconds if G4 P4500 is used. The dwell period set must be a multiple of 10 millseconds. If the dwell period set is less than 10 milliseconds, the dwell period will become 0 milliseconds. If the dwell period set is 23 milliseconds, the dwell period will become 20 milliseconds.
- A dwell period is a time interval between two instructions.



| G-code | Instruction code | | | Operand | Function |
|---|---|---|---|---|---|
| 0017 | | G17 | | None | XY plane selection |
| 0018 | | G18 | | None | XZ plane selection |
| 0019 | | G19 | | None | YZ plane selection |

**Description:**

- Users can select a plane for circular/helical interpolation by means of G17, G18, or G19. The three G-codes do not have any effect on linear interpolation.
- When a program is executed, the three planes available can be switched. If users do not specify a plane, an XY plane will be selected (G17) by the system used.
- Three planes



| G-code | Instruction code | Operand | Function |
|---|---|---|---|
| 0090 | G90 | None | Absolute programming |
| 0091 | G91 | None | Incremental programming |

**Description:**

- G90: Positioning defined with reference to part zero
  If the target position of an axis is greater than its present position, the motor for the axis will rotate clockwise. If the target position of an axis is less than its present position, the motor for the axis will rotate counterclockwise.
- G91: Positioning defined with reference to the present position of an axis
  If the relative target position of an axis is a positive value, the motor for the axis will rotate clockwise. If the relative target position of an axis is a negative value, the motor for the axis will rotate counterclockwise.
- I, J, K, and R indicate incremental positions. They are not affected by G60 and G91.

## 9.5 O Pointers / M-codes

■ O pointers

All O pointers which can be used are shown below. O100 is a main program pointer. Ox0~Ox99 are motion subroutine pointers.

| Instruction code | Operand | Function |
|---|---|---|
| O100 | None | Main program pointer |
| Ox0~Ox99 | None | G-code motion subroutine pointers |

**Description:**

- O100 is a main program pointer. M102 indicates the end of O100.
- Ox0~Ox99 are G-code motion subroutine pointers. Users can use them to create different motion paths. A G-code subroutine pointer is stored in the low byte in SR1052. If users want to store a G-code subroutine pointer in the low byte in SR1052, they have to set bit 14 or bit 15 in SR1052 to 1. After bit 12 in SR1030 is set to ON, the execution of the Ox motion subroutine specified will start.

  Example: The steps of starting the execution of the motion subroutine Ox98 are as follows.

  (1) Setting a number: SR1052=16#8062 (or 16#4062/16#C062)

  (2) Starting the execution of Ox98: SR1030=16#1000

- M2: End of a motion subroutine

  **Example:**

  The main program O100 is composed of N0000~N0100, and the motion subroutine is composed of N0102~N0304.

| Line number | Program |
|---|---|
| **N0000** | **O100** |
| **N0001** | **LD M1000** |
| **N0002** | **MOV H8063 D1868** |
| **N0007** | **MOV H1000 D1846** |
| | **: :** |
| **N0099** | **OUT Y30** |
| **N0100** | **M102** |
| **N0101** | **NOP** |
| **N0102** | **OX50** |
| **N0103** | **G90 G00 X200.0 Y40.0** |
| **N0104** | **G01 X500.0 F25.0** |
| | **: :** |
| **N0304** | **M2** |

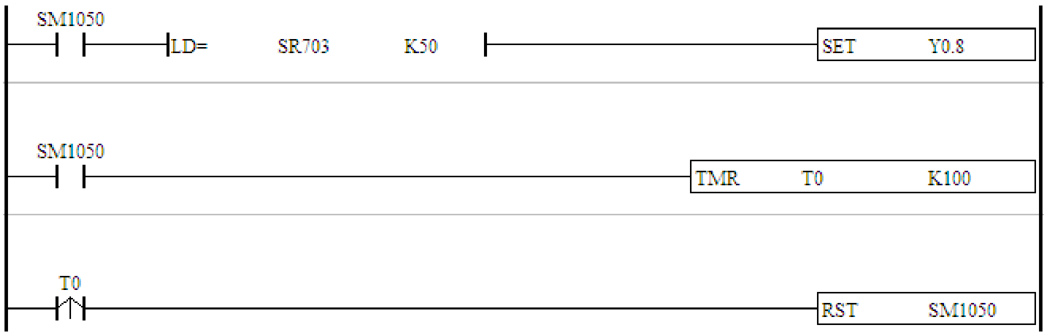
9

- M-codes

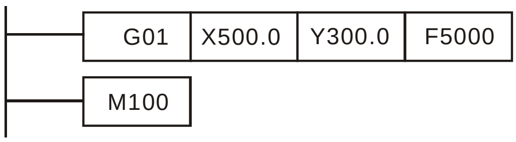
| Instruction code | Operand | Function |
|---|---|---|
| M0~M65535 | None | M-code instructions |

**Description:**

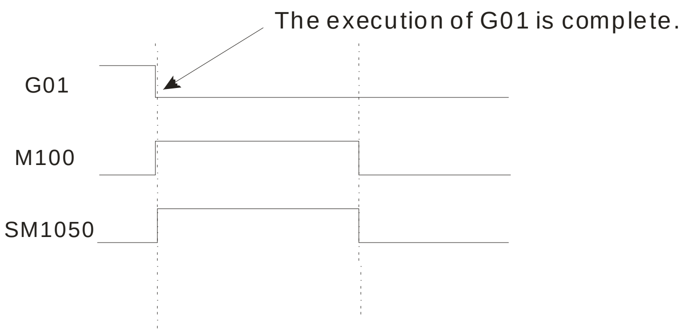
- M102 indicates the end of O100, and M02 indicates the end of a motion subroutine. Users should avoid using M102 and M02.
- M-codes are used in Ox motion subroutines, If an M-code is executed, the M-code will be stored in SR703, and SM1050 will be automatically set to ON. If SM1050 is OFF, the execution of an M-code is complete.
- If an M-code is executed, SM1050 will be ON, and the M-code will be stored in SR703. Users can set control conditions in O100 by means of this character.
  - When M50 is executed, Y0.8 is ON. The execution of M50 will be complete after one second. O100 is shown below.




- M codes can be used in two modes. If an M-code forms a line, it is used in after mode. If an M-code is in back of a motion instruction, it is used in with mode. The difference is described below.
  - After mode: An M-code forms a line.

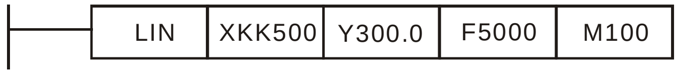


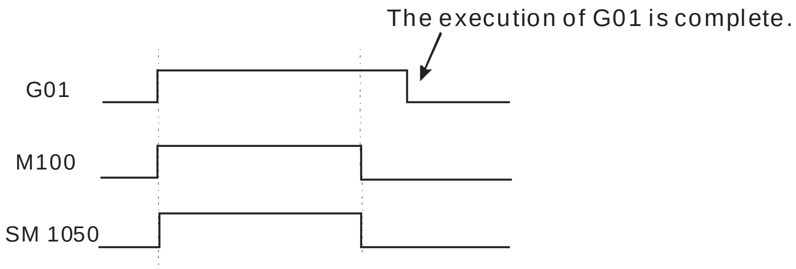

Timing diagram:




After G01 is executed, the M-code M100 will be started automatically, and SM1050 will be automatically turned ON. If users want to stop M100, they have to turn SM1050 OFF. If they want to start the M-code again, they can create a program that starts the M-code again.

- With mode: An M-code is in back of a motion instruction.



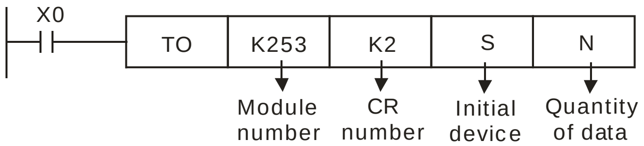




When the execution of G01 is triggered, the M-code M100 is started automatically, and SM1050 is automatically turned ON. If users want to stop M100, they have to turn SM1050 OFF. If they want o start the M-code again, they have to set parameters after the execution of G01 is complete, and create a program that starts the M-code again.

## 9.6 Description of TO

An AH500 series motion control module can start and stop linear interpolation by means of the instruction TO. The use of TO to set linear interpolation is described below.

- CR#2: Starting interpolation




- Data

| Device | Setting |
|---|---|
| $\mathbf{S}$, $\mathbf{S}_{+1}$ | Axes specified |
| $\mathbf{S}_{+2}$, $\mathbf{S}_{+3}$ | Speed of interpolation |
| $\mathbf{S}_{+4}$, $\mathbf{S}_{+5}$ | Poistion of the first axis |
| $\mathbf{S}_{+6}$, $\mathbf{S}_{+7}$ | Position of the second axis |
| $\mathbf{S}_{+8}$, $\mathbf{S}_{+9}$ | Poistion of the third axis |
| : | : |
| $\mathbf{S}_{+29}$, $\mathbf{S}_{+28}$ | Position of the fifteenth axis |
| $\mathbf{S}_{+31}$, $\mathbf{S}_{+30}$ | Position of the sixteenth axis |

- The device ($\mathbf{S}$, $\mathbf{S}_{+1}$) is described below.

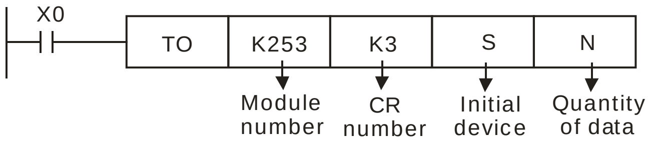
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Axis number | 16th axis | | 15th axis | | 14th axis | | 13th axis | | 12th axis | | 11th axis | | 10th axis | | 9th axis | |
| **Bit** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Axis number | 8th axis | | 7th axis | | 6th axis | | 5th axis | | 4th axis | | 3rd axis | | 2nd axis | | 1st axis | |

- Evey axis is controlled by two bits in ($\mathbf{S}$, $\mathbf{S}_{+1}$).

| Value | Definition |
|---|---|
| 0 | Not participating in interpolation |

| Value | Definition |
|---|---|
| 1 | Participating in interpolation |
| 2 | Not used |
| 3 | Not used |

■ CR#3: Stopping interpolation




● Data

| Device | Setting |
|---|---|
| **S**, **S**$_{+1}$ | Axes specified |

● The device (**S**, **S**$_{+1}$) is described below.

| **Bit** | **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Axis number | 16$^{th}$ axis | | 15$^{th}$ axis | | 14$^{th}$ axis | | 13$^{th}$ axis | | 12$^{th}$ axis | | 11$^{th}$ axis | | 10$^{th}$ axis | | 9$^{th}$ axis | |
| **Bit** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Axis number | 8$^{th}$ axis | | 7$^{th}$ axis | | 6$^{th}$ axis | | 5$^{th}$ axis | | 4$^{th}$ axis | | 3$^{rd}$ axis | | 2$^{nd}$ axis | | 1$^{st}$ axis | |

● Evey axis is controlled by two bits in (**S**, **S**$_{+1}$).

| Value | Definition |
|---|---|
| 0 | Not participating in interpolation |
| 1 | Stopping linear interpolation |
| 2 | Not used |
| 3 | Not used |

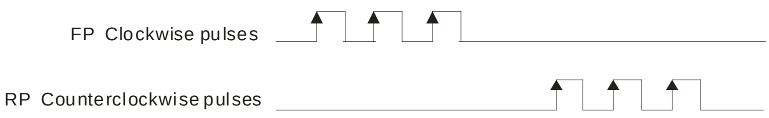
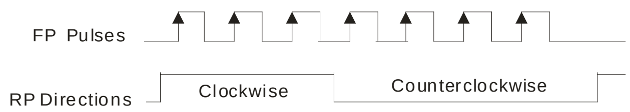
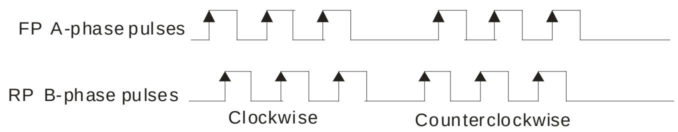
■ Users can set the parameters of the axes participating in linear interpolation by means of SR1000+100*N.

| **Special data register** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SR1000+100*N | | | | | | | | | | | | | | | |
| **b15** | **b14** | **b13** | **b12** | **b11** | **b10** | **b9** | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| - | Curve (Note 3) | Mode of triggering the calculation of the target position | Relative/Absolute coordinates | Direction in which the motor used rotates | Mode of triggering the return to home | Mode of returning home | Direction in which the axis specified returns home | - | - | Output type (Positive logic) (Note 2) | | - | | Unit (Note 1) | |

Note 1:

| | Motor unit | Compound unit | Mechanical unit |
|---|---|---|---|
| **Position** | pulse | µm | |
| | pulse | mdeg | |
| | pulse | $10^{-4}$ inches | |
| **Speed** | pulse/second | | centimeter/minute |
| | pulse/second | | 10 degrees/minute |
| | pulse/second | | inch/minute |

Note 2:

<table>
<tr><th>b5</th><th>b4</th><th>Output type (positive logic)</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td></td><td>Counting up/down</td></tr>
<tr><td>0</td><td>1</td><td></td><td>Pulses+Directions</td></tr>
<tr><td>1</td><td>0</td><td rowspan="2"></td><td>A/B-phase pulses</td></tr>
<tr><td>1</td><td>1</td><td>Four times the frequency of A/B-phase pulses</td></tr>
</table>

Note 3:

| bit# | Description |
|---|---|
| 12 | Bit 12=0: Absolute coordinates<br>Bit 12=1: Relative coordinates |
| 14 | Bit 14=0: Trapezoid curve<br>Bit 14=1: S curve |

Users can judge whether interpolation is complete by means of the motion flag SR1048+100*N.

# Chapter 10 High-speed Counters and High-speed Timers

## Table of Contents

An AH500 series motion control module is equipped with high-speed counter and virtual high-speed counter. These high-speed counters can be used as timers. The functions of high-speed counters and the functions of timers are described below.

## 10.1 High-speed Counters

1. Selecting a mode of counting

   The setting of high-speed counters is described below.

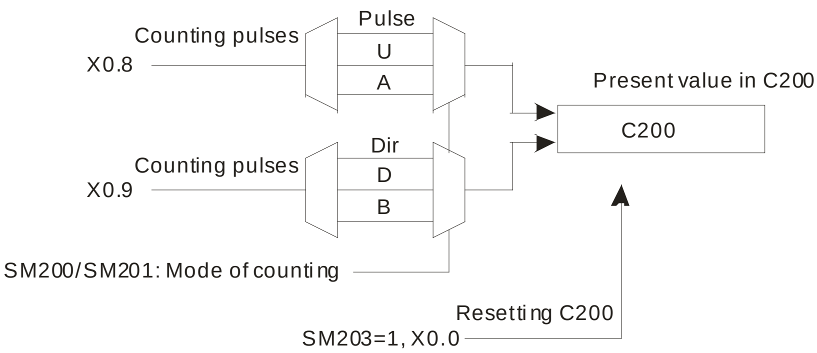
| Number | Counter | Mode of counting: Device | Mode of counting: Setting value*3 | Resetting of a counter | External resetting terminal | External input terminal*1 *2 |
|---|---|---|---|---|---|---|
| 0 | C200 | K1SM200 | 0: U/D<br>1: P/D<br>2: A/B (One time the frequency of A/B-phase inputs)<br>3: 4A/B (Four times the frequency of A/B-phase inputs) | SM203 | X0.0+ and X0.0- | X0.8, X0.9, and S/S |
| 1 | C204 | K1SM204 | | SM207 | X0.1+ and X0.1- | X0.10, X0.11, and S/S |
| 2 | C208 | K1SM208 | | SM211 | X0.2+ and X0.2- | X0.12，X0.13, and S/S |
| 3 | C212 | K1SM212 | | SM215 | X0.3+ and X0.3- | X0.14, X0.15, and S/S |
| 4 | C216 | K1SM216 | | SM219 | X0.2+ and X0.2- | X0.12，X0.13, and S/S |
| 5 | C220 | K1SM220 | | SM223 | X0.3+ and X0.3- | X0.14, X0.15, and S/S |

*1. The input terminals of AH20MC-5A are differential input terminals. X0.8 and X0.9 on AH15PM-5A are differential input terminals. The input terminals of AH05PM-5A/AH10PM-5A are transistors whose collectors are open collectors. X0.10~X0.15 on AH15PM-5A are transistors whose collectors are open collectors.

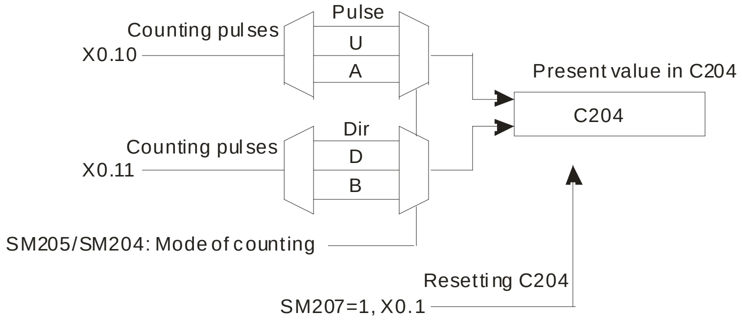
*2. The terminal S/S on AH05PM-5A/AH10PM-5A must be connected. X0.10~X0.15 on AH15PM-5A must be connected to the terminal S/S.

*3. U/D: Counting up/Counting down; P/D: Pulse/Direction; A/B: A phase/B phase

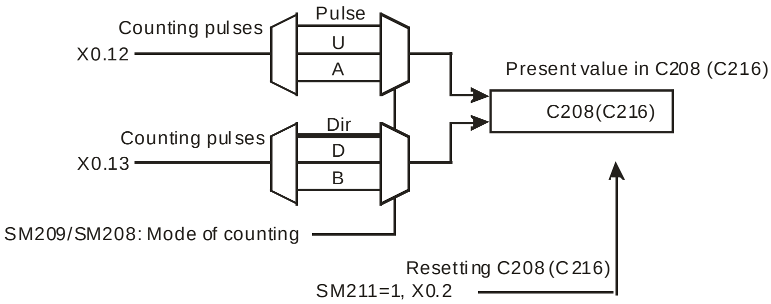
- Users can select a mode of counting by setting SM200 and SM201. Input signals are controlled by X0.8 and X0.9. If SM203 is ON, the function of resetting C200 will be enabled. Resetting signals are controlled by X0.0.



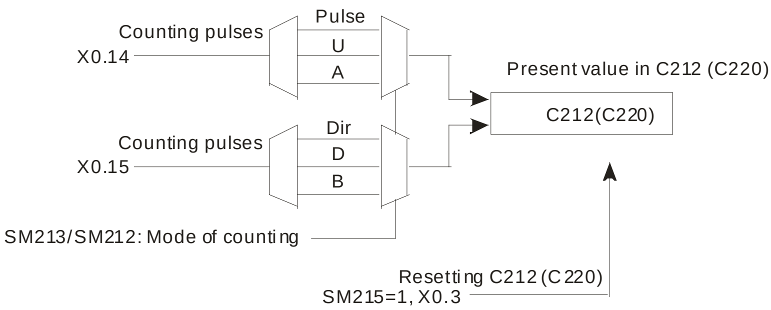
- C204: Users can select a mode of counting by setting SM204 and SM205. Input signals are controlled by X0.10 and X0.11. If SM207 is ON, the function of resetting C204 will be enabled. Resetting signals are controlled by X0.1.




- C208: Users can select a mode of counting by setting SM208 and SM209. Input signals are controlled by X0.12 and X0.13. If SM211 is ON, the function of resetting C208 will be enabled. Resetting signals are controlled by X0.2. C216 counts with C208. It is the first virtual counter.




- C212: Users can select a mode of counting by setting SM212 and SM213. Input signals are controlled by X0.14 and X0.15. If SM215 is ON, the function of resetting C212 will be enabled. Resetting signals are controlled by X0.3. C220 counts with C212. It is the second virtual counter.




2. If a power cut occurs when a general counter counts, the present value of the counter will be cleared.
3. If a counter counts up from the present value 2,147,483,647, the next value following 2,147,483,647 will be -2,147,483,648. If a counter counts down from the present value -2,147,483,648, the next value following -2,147,483,648 will be 2,147,483,647.

**Example:**

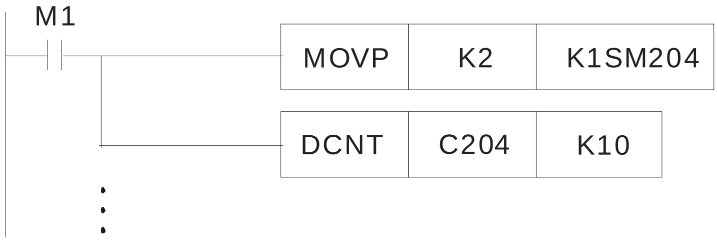
The steps of setting C204 are as follows.

(1) Write K2 into K1SM204.

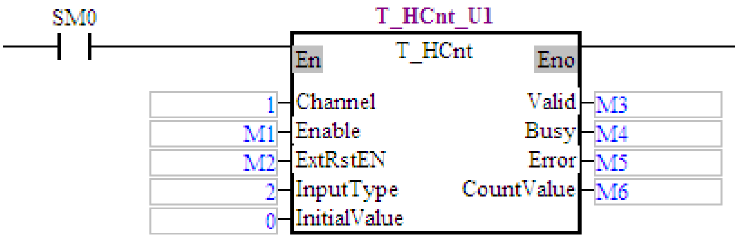
(2) Enable C204.

The program for step 1 and step 2 is shown below.

- Ladder diagram




- Function block




(3) If users want to clear the present counter value by means of an external signal, they have to write 16#A into K1SM204.

- Ladder diagram: K1SM204=16#A

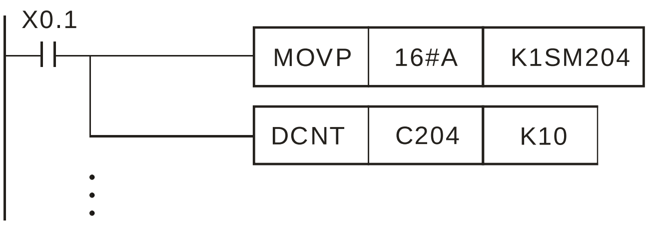
| SM207 | SM206 | SM205 | SM204 |
|---|---|---|---|
| 1 | 0 | 1 | 0 |

- Function block: The ExRstEn pin is set to ON.

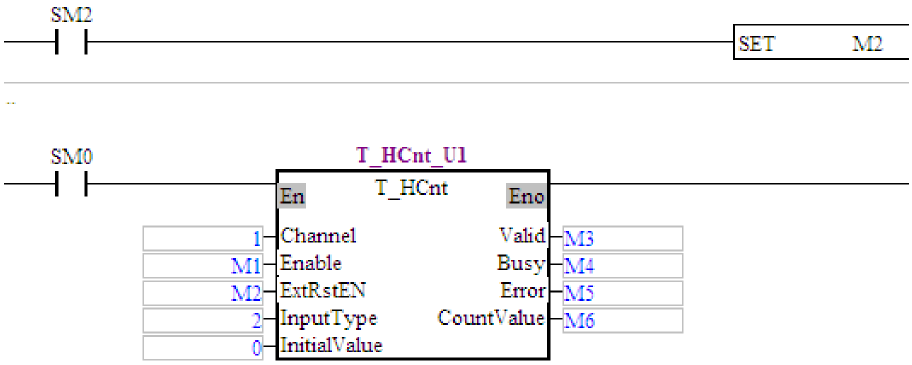
(4) C204 is enabled. If X0.1 is ON, the present value of C204 will become zero.

The program for step 3 and step 4 is shown below.

- Ladder diagram



■ Function block



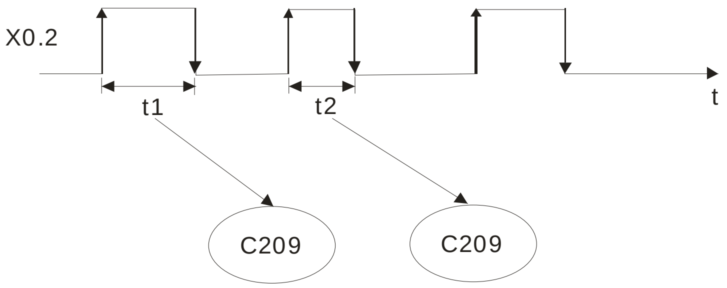

## 10.2 High-speed Timers

The setting of high-speed counters is described below.

<table>
<tr><th rowspan="2">Number</th><th rowspan="2">Counter</th><th colspan="2">Mode of measuring time</th><th rowspan="2">External signal</th><th rowspan="2">Storage device</th></tr>
<tr><th>Device</th><th>Setting value</th></tr>
<tr><td>0</td><td>C200</td><td>K1SM200</td><td rowspan="4"><table><tr><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>-</td><td>Enabling a timer</td><td>-</td><td>Selecting a mode</td></tr></table>Bit 2: Enabling a timer<br>Bit 0: (1) 0: General mode (The interval between the rising edge of a pulse and the falling edge of the pulse is measured.)<br>(2) 1: Cyclic mode (The interval between the rising edge of a pulse and the rising edge of the next pulse is measured.)</td><td>X0.0</td><td>C201</td></tr>
<tr><td>1</td><td>C204</td><td>K1SM204</td><td>X0.1</td><td>C205</td></tr>
<tr><td>2</td><td>C208</td><td>K1SM208</td><td>X0.2</td><td>C209</td></tr>
<tr><td>3</td><td>C212</td><td>K1SM212</td><td>X0.3</td><td>C213</td></tr>
</table>

**Example 1: Using C20s in general mode**

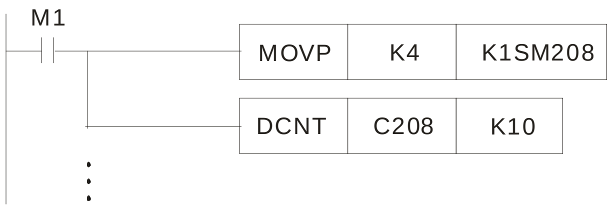
1. Users have to select the general mode, and enable the timer, that is, they have to write K4 into K1SM208.
2. C208 is enabled. The interval between the rising edge of a pulse received through X0.2 and the falling edge of the pulse is measured. The interval is written into C209. (Unit: 0.01 microseconds)



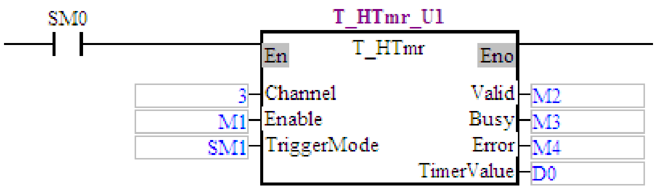

General mode
Unit: 0.01 μs

The program is shown below.

■ Ladder diagram

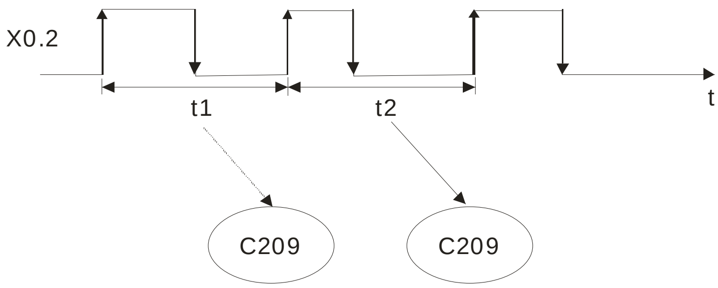



■ Function block




**Example 2: Using C208 in cyclic mode**

1. Users have to write K5 into K1SM208
2. C208 is enabled. The interval between the rising edge of a pulse received through X0.2 and the rising edge of the next pulse is measured. The interval is written into C209. (Unit: 0.01 microseconds)

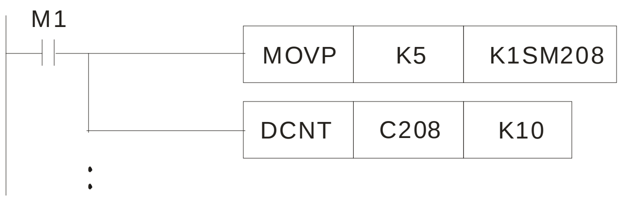



Cyclic mode

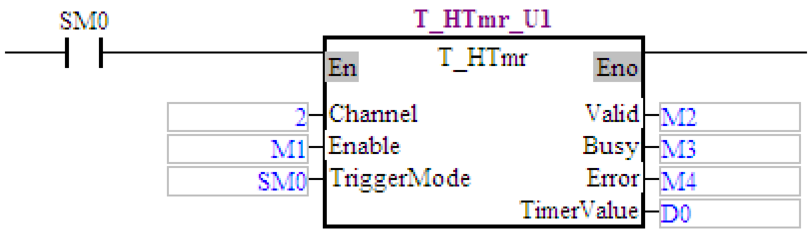
Unit: 0.01 μs

The program is shown below.

■ Ladder diagram




■ Function block




3. The cyclic mode is used to measure a frequency.

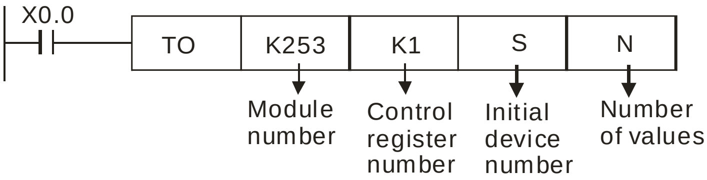
# Chapter 11 High-speed Capture and High-speed Comparison

## Table of Contents

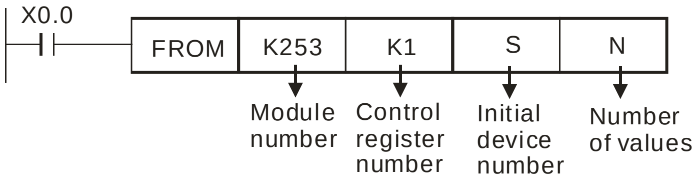
# 11.1 Format of an Instruction

An AH500 series motion control module sets and reads values by means of the instructions FROM and TO. The use of FROM/TO to set high-speed comparison and high-speed capture, and to read values is described below.

■ **Control**

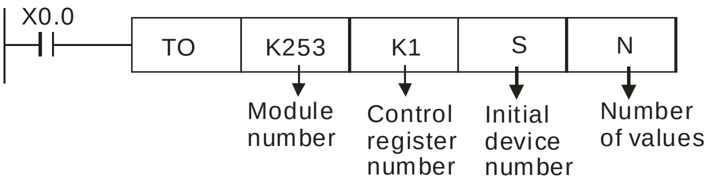



■ **Reading**




# 11.2 Comparison

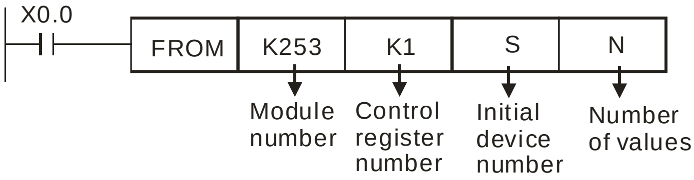
■ **Control**




● **Definitions**

| Device | Control |
|---|---|
| S | Initial group number n (n=0~7) |
| **$S_{+1}$** | **0** |
| ($S_{+3}$, $S_{+2}$) | Control registers whose group number is n |
| ($S_{+5}$, $S_{+4}$) | Data registers whose group number is n |
| ($S_{+7}$, $S_{+6}$) | Control registers whose group number is n+1 |
| ($S_{+9}$, $S_{+8}$) | Data registers whose group number is n+1 |
| : | : |
| ($S_{+31}$, $S_{+30}$) | Control registers whose group number is n+7 |
| ($S_{+33}$, $S_{+32}$) | Data registers whose group number is n+7 |
| $S_{+50}$ | Number of devices=2+m*4<br>m=Number of groups (8 groups at most can be used.) |

## ■ Reading




### ● Definitions

| Device | Reading the values in counters |
|---|---|
| S | Initial group number n (n=0~7) |
| **$S_{+1}$** | **0** |
| ($S_{+3}$, $S_{+2}$) | Control registers whose group number is n |
| ($S_{+5}$, $S_{+4}$) | Data registers whose group number is n |
| ($S_{+7}$, $S_{+6}$) | Control registers whose group number is n+1 |
| ($S_{+9}$, $S_{+8}$) | Data registers whose group number is n+1 |
| : | : |
| ($S_{+31}$, $S_{+30}$) | Control registers whose group number is n+7 |
| ($S_{+33}$, $S_{+32}$) | Data registers whose group number is n+7 |
| $S_{+50}$ | Number of devices=2+m*4<br>m=Number of groups (8 groups at most can be used.) |

## ■ Control/Reading

(1) The format of a control register in a high-speed comparison mode is described below.

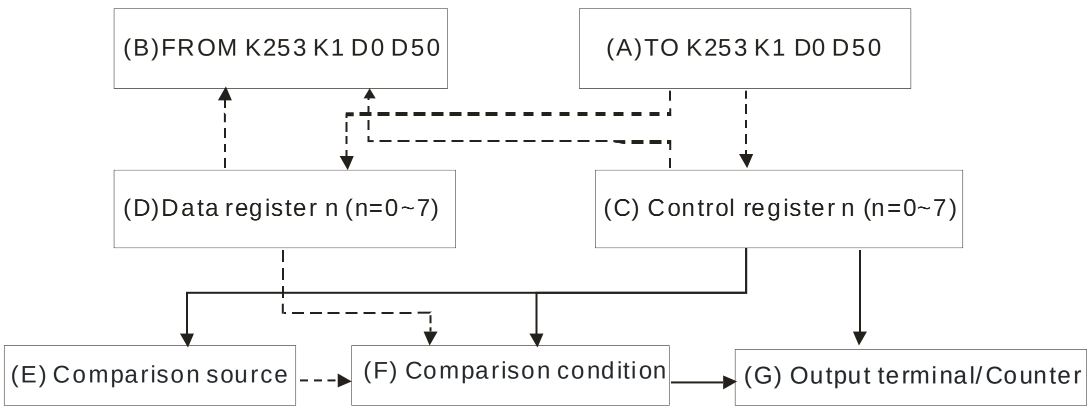
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Item | | | | | Comparison result | | | | Output action | | Condition | | Comparison source | | | |

| Item | Bit | Value | AH20MC-5A | AH10PM-5A/ AH15PM-5A | AH05PM-5A |
|---|---|---|---|---|---|
| Comparison source | [3-0] | 0 | Present position of the 1st axis | Present position of the 1st axis | Present position of the 1st axis |
| | | 1 | Present position of the 2nd axis | Present position of the 2nd axis | Present position of the 2nd axis |
| | | 2 | Present position of the 3rd axis | Present position of the 3rd axis | Present position of the 3rd axis |
| | | 3 | Present position of the 4th axis | Present position of the 4th axis | Present position of the 4th axis |
| | | 4 | Value in C200 | Value in C200 | Value in C200 |
| | | 5 | Value in C204 | Value in C204 | - |
| | | 6 | Value in C208 | Value in C208 | - |
| | | 7 | Value in C212 | Value in C212 | - |
| Comparison condition | [5-4] | 0 | Capture mode (Please refer to section 11.4 for more information.) | | - |
| | | 1 | Equal to ( = ) | Equal to ( = ) | Equal to ( = ) |
| | | 2 | Greater than or equal to (≧) | Greater than or equal to (≧) | Greater than or equal to (≧) |

| Item | Bit | Value | AH20MC-5A | AH10PM-5A/ AH15PM-5A | AH05PM-5A |
|---|---|---|---|---|---|
| Comparison condition | [5-4] | 3 | Less than or equal to (≦) | Less than or equal to (≦) | Less than or equal to (≦) |
| Output action | [7-6] | 0 | Set | Set | Set |
| | | 1 | Reset | Reset | Reset |
| | | 2, 3 | No output | No output | No output |
| Comparison result | [11-8] | 0 | Y0.8 | Y0.8 | Y0.8 |
| | | 1 | Y0.9 | Y0.9 | Y0.9 |
| | | 2 | Y0.10 | Y0.10 | - |
| | | 3 | Y0.11 | Y0.11 | - |
| | | 4 | Clearing the value in C200 | Clearing the value in C200 | Clearing the value in C200 |
| | | 5 | Clearing the value in C204 | Clearing the value in C204 | - |
| | | 6 | Clearing the value in C208 | Clearing the value in C208 | - |
| | | 7 | Clearing the value in C212 | Clearing the value in C212 | - |

The comparison value stored in data registers is a 32-bit value.

(2) A comparison is shown below. Users use FROM/TO to read/write values so that they can compare data.




※ The dotted lines are data procedures, and the solid lines are control procedures.

Block (A): The instruction TO is used to write data into control registers (block C) and data registers (block D).

Block (B): The instruction FROM is used to read data from control registers (block C) and data registers (block D).

Block (C): User set a comparison source (block E), a comparison condition (block F), and an output terminal (block G) in a control register in accordance with the value it receives by means of TO.

Block (D): The value that users write into data registers by means of the instruction TO is compared with a comparison source (block E).

Block (E): The present positions of four axes, the values in C200, C204, C208, and C212 are comparison sources. Please refer to Chapter 10 for more information about high-speed counters.

Block (F): There are three comparison conditions, they are equal to, greater than or equal to, and less than or equal to. If block D and block E meet the comparison condition set, the output terminal selected will be set to ON, the counter selected will be reset, the output terminal selected will be reset to OFF, or the counter selected will not be reset.

Block (G): If a comparison condition is met, Y0.8, Y0.9, Y0.10, Y0.11, C200, C204, C208, or C212 will be set or reset.

Procedure for a high-speed comparison: The instruction TO is used to write data into control registers and data registers (block A).→The comparison source set (block E) is compared with the value in data registers (block D). The comparison result meets the condition set (block F).→Y0.8, Y0.9, Y0.10, Y0.11, C200, C204, C208, or C212 will be set or reset (block G).

■ **Example**

【Description】

A manual pulse generator is used to generate pulses that are sent to C204 in AH10PM-5A.

Comparison conditions:

- If the value in C204 is greater than 100, Y0.9 will be set to ON.
- If the value in C204 is greater than 300, Y0.9 will be reset to OFF.

Two comparators are used in a program. One comparator is used to set Y0.9 to ON, and the other is used to reset Y0.9 to OFF. When Y0.9 is set to ON, no LED indicator on AH10PM-5A will indicate that Y0.9 is ON, but users can know whether Y0.9 is ON by means of its external wiring. As a result, the terminal C1 is connected to the terminal 24G, Y0.9 is connected to X0.2-, X0.2+ is connected to +24V, and X0.10 and X0.11 are connected to a manual pulse generator.

【Steps】

1. After O100 is started, the initial setting of two high-speed comparisons will be carried out.
   - D0=0→Initial group number n=0
   - D1=0
   - D20=10→Writing 10 values by means of the instruction TO (two groups of high-speed comparison values)
   - D60=10→Reading 10 values by means of the instruction FROM (two high-speed comparison values)
2. Two groups of high-speed comparison values are set when M1 is ON.
   - First group: The value in (D3, D2) is 16#125.→The comparison source set is C204. (The value of bit 3~bit 0 is 5.) The comparison condition set is greater than or equal to. (The value of bit 5~bit 4 is 2.) The output action selected is set. (The value of bit7~bit 6 is 0.) The terminal selected is Y0.9. (The value of bit11~bit 8 is 1.)
   - First group: The value in (D5, D4) is K100. If the value in C204 is greater or equal to K100, Y0.9 will be set to ON.
   - Second group: The value in (D7, D6) is 16#165.→The comparison source set is C204. (The value of bit 3~bit 0 is 5.) The comparison condition set is greater than or equal to. (The value of bit 5~bit 4 is 2.) The output action selected is reset. (The value of bit7~bit 6 is 1.) The terminal selected is Y0.9. (The value of bit11~bit 8 is 1.)
   - Second group: The value in (D9, D8) is K300. If the value in C204 is greater or equal to K300, Y0.9 will be reset to OFF.
3. The two high-speed comparisons are started when M2 is ON.

4. The setting of the two high-speed comparisons is read when M3 is ON.

MonitorTable (2)

| Device ... | Radix | Value | Comment |
|---|---|---|---|
| C204 | d32u | 0 | Y axis manual pulse generator of counter value |
| D44 | d32u | 100 | Read Data register From n=0 D5 ..D4 |
| D48 | d32s | 300 | Read Data register From n=1 D9 ..D8 |
| | | | |
| D40 | d16s | 0 | Read Data D0 |
| D41 | d16s | 0 | Read Data D1 |
| D42 | h32 | 00000124 | Read Data register to n=0 |
| D44 | d32u | 100 | Read Data register From n=0 D5 ..D4 |
| D46 | h32 | 00000164 | Read Data register From n=0 D7 ..D6 |
| D48 | d32s | 300 | Read Data register From n=1 D9 ..D8 |

5. When M4 is ON, K1 is moved to SM204~SM207. C204 is started when M5 is set to ON. (Mode of counting: Pulse/Direction)
6. Use the manual pulse generator, and check whether C204 counts.

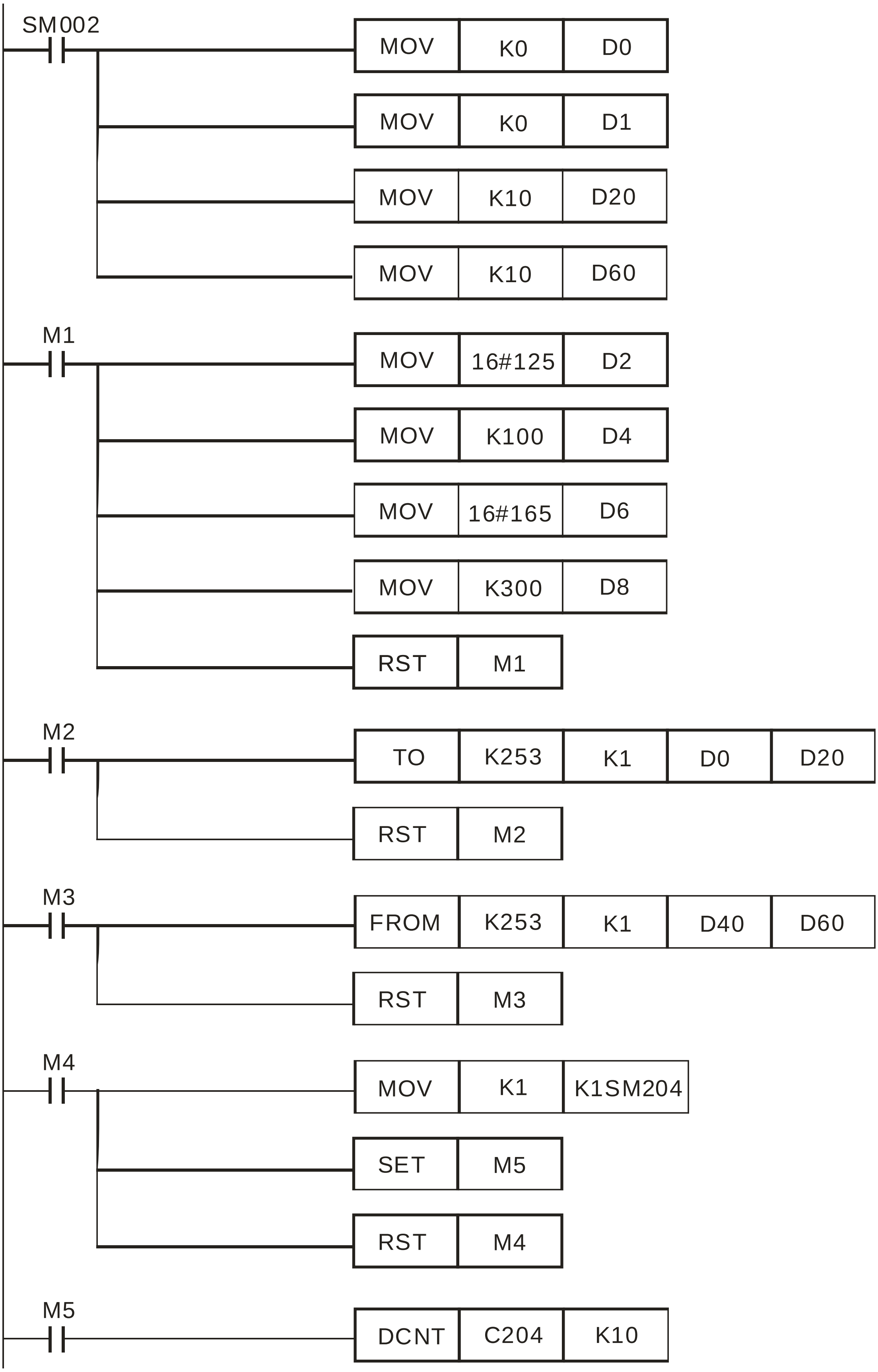
MonitorTable (2)

| Device ... | Radix | Value | Comment |
|---|---|---|---|
| C204 | d32u | $\boxed{95}$ | Y axis manual pulse generator of counter value |
| D44 | d32u | 100 | Read Data register From n=0 D5 ..D4 |
| D48 | d32s | 300 | Read Data register From n=1 D9 ..D8 |

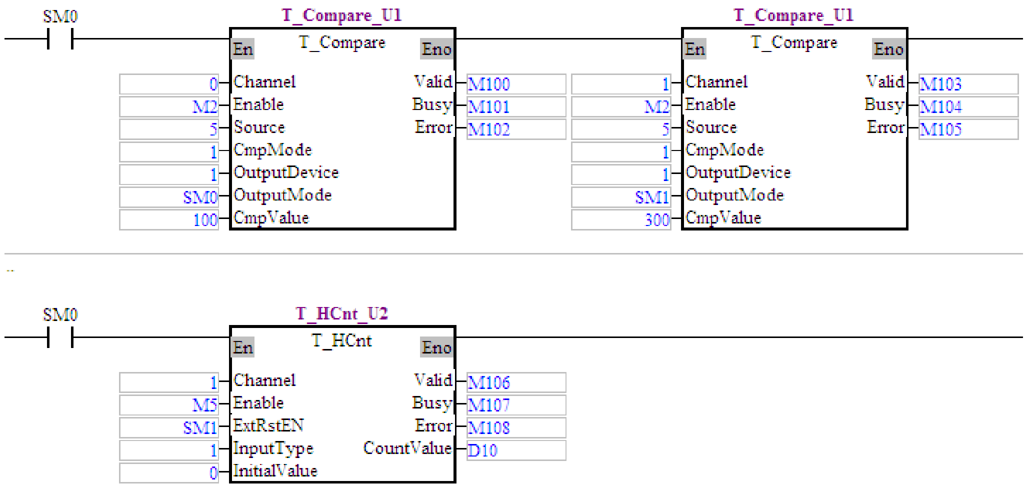
7. Use the manual pulse generator. Check whether X0.2 is ON by means of PMSoft when the value in C204 is greater than 100.
8. Use the manual pulse generator. Check whether X0.2 is OFF when the value in C204 is greater than 300.

【Program in PMSoft】

■ Ladder diagram

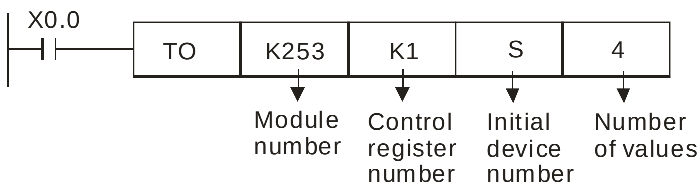


■ Function blocks




## 11.3 Clearing an Output

The output of a high-speed comparison can be cleared.




- **Definitions**

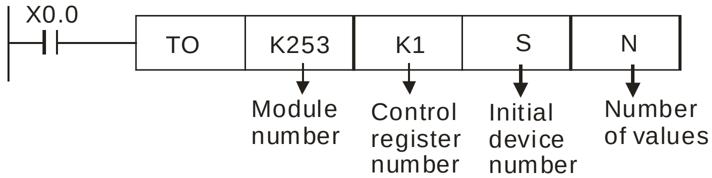
| Device | Clearing an output |
|---|---|
| S | 0 |
| **$S_{+1}$** | **1** |
| ($S_{+2}$, $S_{+3}$) | Output |

Users can set the function corresponding to the output of a high-speed comparison by means of $S_{+2}$, $S_{+3}$.

| Setting | Bit | AH20MC-5A | AH10PM-5A/ AH15PM-5A | AH05PM-5A |
|---|---|---|---|---|
| Comparison result | 0 | Y0.8 | Y0.8 | Y0.8 |
| | 1 | Y0.9 | Y0.9 | Y0.9 |
| | 2 | Y0.10 | Y0.10 | - |
| | 3 | Y0.11 | Y0.11 | - |
| | 4 | Clearing the value in C200 | Clearing the value in C200 | Clearing the value in C200 |
| | 5 | Clearing the value in C204 | Clearing the value in C204 | - |
| | 6 | Clearing the value in C208 | Clearing the value in C208 | - |
| | 7 | Clearing the value in C212 | Clearing the value in C212 | - |

## 11.4 Capture

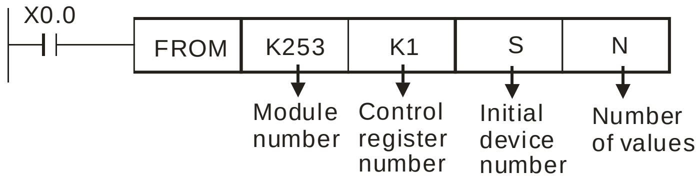
■ **Control**




● **Definitions**

| Device | Control Setting |
|---|---|
| S | Initial group number n (n=0~7) |
| **$S_{+1}$** | **0** |
| ($S_{+3}$, $S_{+2}$) | Control registers whose group number is n |
| ($S_{+5}$, $S_{+4}$) | Data registers whose group number is n |
| ($S_{+7}$, $S_{+6}$) | Control registers whose group number is n+1 |
| ($S_{+9}$, $S_{+8}$) | Data registers whose group number is n+1 |
| : | : |
| ($S_{+31}$, $S_{+30}$) | Control registers whose group number is n+7 |
| ($S_{+33}$, $S_{+32}$) | Data registers whose group number is n+7 |
| $S_{+50}$ | Number of devices=2+m*4<br>m=Number of groups (8 groups at most can be used.) |

■ **Reading**




● **Definitions**

| Device | Counter Status Reading |
|---|---|
| S | Initial group number n (n=0~7) |
| **$S_{+1}$** | **0** |
| ($S_{+3}$, $S_{+2}$) | Control registers whose group number is n |
| ($S_{+5}$, $S_{+4}$) | Data registers whose group number is n |
| ($S_{+7}$, $S_{+6}$) | Control registers whose group number is n+1 |
| ($S_{+9}$, $S_{+8}$) | Data registers whose group number is n+1 |
| : | : |
| ($S_{+31}$, $S_{+30}$) | Control registers whose group number is n+7 |
| ($S_{+33}$, $S_{+32}$) | Data registers whose group number is n+7 |
| $S_{+50}$ | Number of devices=2+m*4<br>m=Number of groups (8 groups at most can be used.) |

## Control/Reading

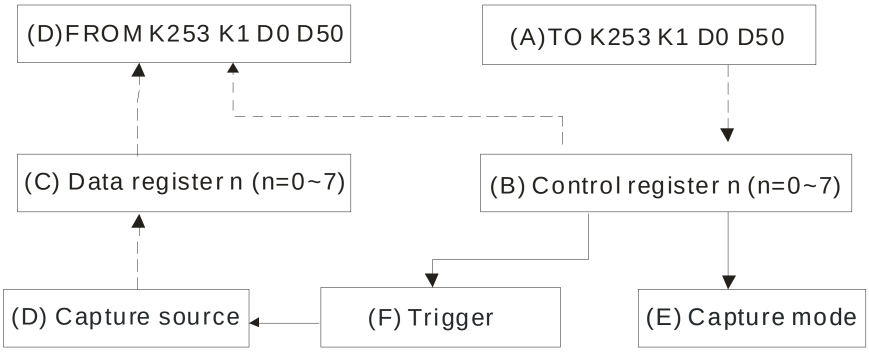
(1) The format of a control register in a high-speed capture mode is described below.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Item | Trigger | | | | | | | | | | Setting | | Capture source | | | |

| Item | Bit | Value | AH20MC-5A | AH10PM-5A/ AH15PM-5A | AH05PM-5A |
|---|---|---|---|---|---|
| Capture source | [3-0] | 0 | Present position of the 1st axis | Present position of the 1st axis | Present position of the 1st axis |
| | | 1 | Present position of the 2nd axis | Present position of the 2nd axis | Present position of the 2nd axis |
| | | 2 | Present position of the 3rd axis | Present position of the 3rd axis | Present position of the 3rd axis |
| | | 3 | Present position of the 4th axis | Present position of the 4th axis | Present position of the 4th axis |
| | | 4 | Value in C200 | Value in C200 | Value in C200 |
| | | 5 | Value in C204 | Value in C204 | - |
| | | 6 | Value in C208 | Value in C208 | - |
| | | 7 | Value in C212 | Value in C212 | - |
| Setting | [5-4] | 0 | Capture mode | | |
| External trigger | [15-12] | 0 | X0.0 | X0.0 | X0.0 |
| | | 1 | X0.1 | X0.1 | X0.1 |
| | | 2 | X0.2 | X0.2 | - |
| | | 3 | X0.3 | X0.3 | - |
| | | 4 | - | - | - |
| | | 5 | - | - | - |
| | | 6 | - | - | - |
| | | 7 | - | - | - |
| | | 8 | X0.8 | X0.8 | X0.8 |
| | | 9 | X0.9 | X0.9 | X0.9 |
| | | 10 | X0.10 | X0.10 | - |
| | | 11 | X0.11 | X0.11 | - |
| | | 12 | X0.12 | X0.12 | X0.12 |
| | | 13 | X0.13 | X0.13 | X0.13 |
| | | 14 | X0.14 | X0.14 | - |
| | | 15 | X0.15 | X0.15 | - |

The value captured is stored in data registers, and is a 32-bit value. Users can write an initial value into the data registers. After an input terminal is set to ON, the value captured will be updated.

(2) A deviation often occurs when the present position of an axis or the value in C200/C204/C208/C212 is read. To prevent a deviation from occurring, users read a value immediately by setting an input terminal to ON. Capture is described below.




Block (A): The instruction TO is used to write data into control registers (block C).

Block (B): Users set a capture source (block D), set bit 5~bit 4 to 0 (block E), and set a trigger (block F) in a control register.

Block (C): The capture of a value (block D) is triggered by an input terminal, and the value captured is stored in data registers.

Block (D): The present positions of four axes, the values in C200, C204, C208, and C212 are capture sources.

Block (E): Capture mode

Block (F): External trigger

Block (G): The instruction FROM is used to read data from control registers (block C) and data registers (block B). The values stored in the data registers are values captured.

Procedure for a high-speed capture: The instruction TO is used to write data into control registers (block A).→An input terminal is set to ON (block F).→The present position of the $1^{st}/2^{nd}/3^{rd}/4^{th}$ axis, or the value in C200/C204/C208/C212 is captured (block D). The value captured is stored in data registers (block C).→Users read the value captured by means of the instruction FROM.

■ Example

【Description】

Start the high-speed counter C204. The value in C204 is captured when X0.1 is set to ON.

【Steps】

1. When SM002 in O100 is ON, the initial setting of high-speed capture is carried out.
   - D0=0→Initial group number n=0
   - D1=0
   - D20=10→Writing 6 values by means of the instruction TO (Only one value is captured.)
   - D60=10→Reading 6 values by means of the instruction FROM (Only one value is captured.)
2. When M1 is ON, the high-speed capture is set.
   - The value in (D3, D2) is 16#1005.→The capture source set is C204. (The value of bit 3~bit 0 is 5). The mode selected is a capture mode. (The value of bit 5~bit 4 is 0.) The trigger selected is X0.1. (The value of bit 15~bit 12 is 1.)
   - The value in (D5, D4) is K100. Users can set (D5, D4) by themselves.
3. The high-speed capture is started when M2 is ON.

4. The setting of the high-speed capture is read when M3 is ON.

MonitorTable (2)

| Devic... | Radix | Value | Comment |
|---|---|---|---|
| C204 | d32u | 0 | Y axis manual pulse generator of counter valu |
| D44 | d32u | 100 | Read Data register From n=0 D5 ..D4 |
| | | | |
| D40 | d16s | 0 | Read Data D0 |
| D41 | d16s | 0 | Read Data D1 |
| D42 | h32 | 00005004 | Read Data register to n=0 |
| D44 | d32u | 100 | Read Data register From n=0 D5 ..D4 |

5. When M4 is ON, K1 is moved to SM204~SM207. C204 is started when M5 is set to ON. (Mode of counting: Pulse/Direction)
6. Use a manual pulse generator, and check whether C204 counts.

MonitorTable (2)

| Devic... | Radix | Value | Comment |
|---|---|---|---|
| C204 | d32u | 244 | Y axis manual pulse generator of counter valu |
| D44 | d32u | 100 | Read Data register From n=0 D5 ..D4 |
| | | | |
| D40 | d16s | 0 | Read Data D0 |
| D41 | d16s | 0 | Read Data D1 |
| D42 | h32 | 00005004 | Read Data register to n=0 |
| D44 | d32u | 100 | Read Data register From n=0 D5 ..D4 |

7. Use the manual pulse generator, and set X0.1 to ON.
8. The value captured is read when M3 is ON. When X0.1 is ON, the value in C204 is captured. The value captured is 677.

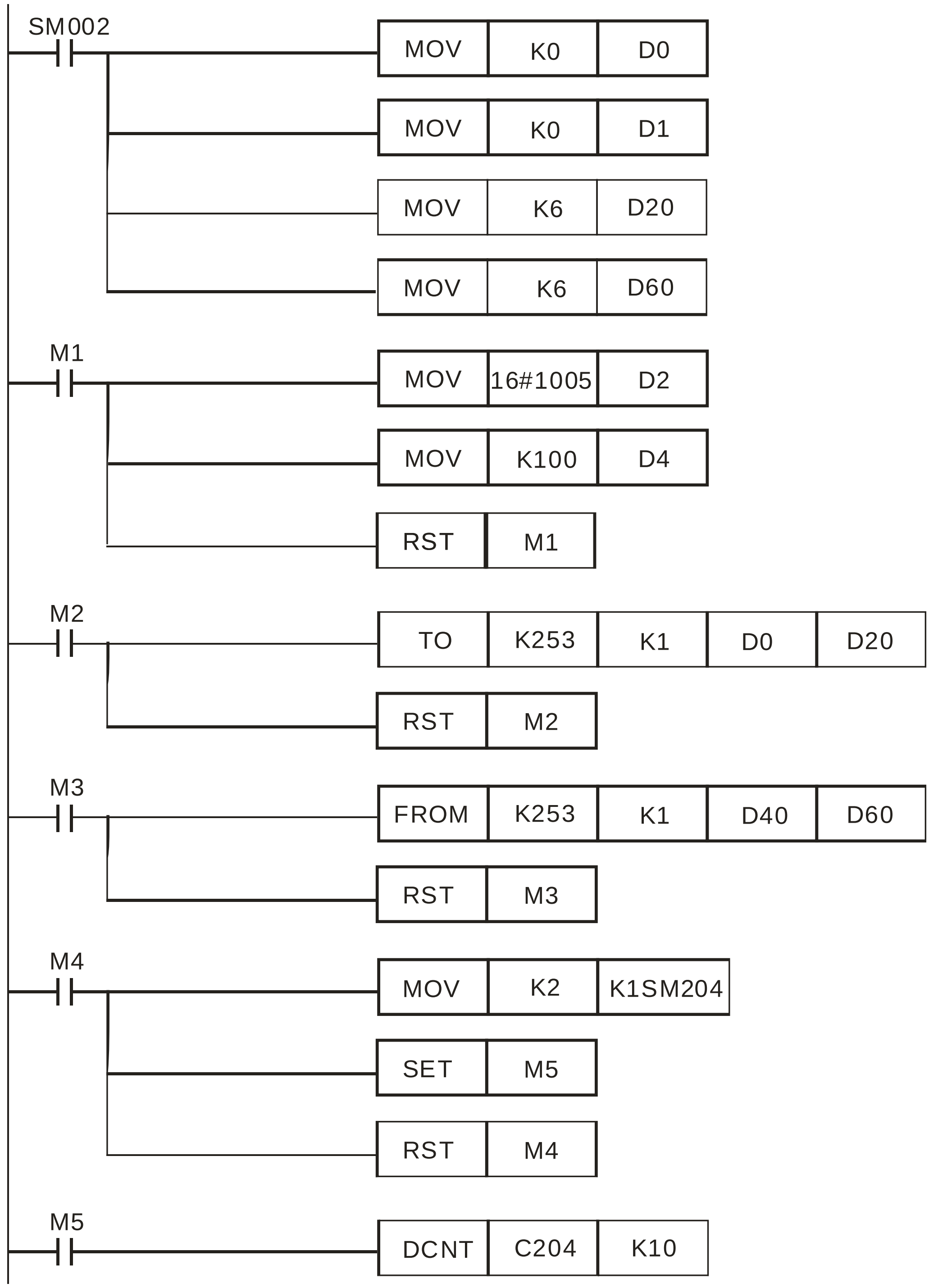
MonitorTable (2)

| Devic... | Radix | Value | Comment |
|---|---|---|---|
| C204 | d32u | 726 | Y axis manual pulse generator of counter valu |
| D44 | d32u | 677 | Read Data register From n=0 D5 ..D4 |
| | | | |
| D40 | d16s | 0 | Read Data D0 |
| D41 | d16s | 0 | Read Data D1 |
| D42 | h32 | 00005004 | Read Data register to n=0 |
| D44 | d32u | 677 | Read Data register From n=0 D5 ..D4 |

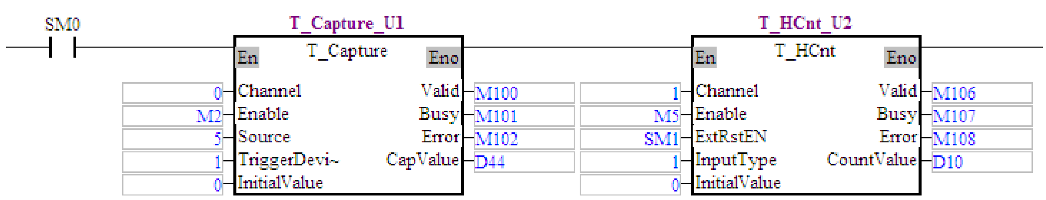
When X0.1 is ON, the value in C204 is captured.

【Program in PMSoft】

- Ladder diagram

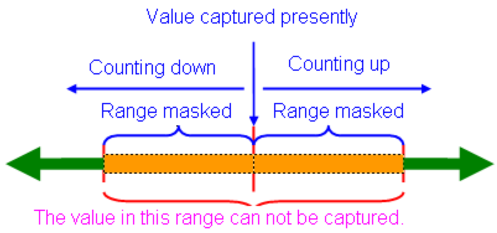



- Function blocks

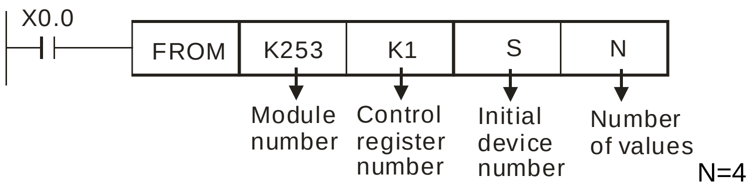


## 11.5 Masking

A value can be masked in a high-speed capture mode. If the relative difference between the value captured this time and the value captured last time is in the range which can be masked, the signal which triggers the capture of the value this time will be disregarded.




■ Setting




- Definitions

| Device | Setting |
|---|---|
| S | 0 |
| **$S_{+1}$** | **2** |
| ($S_{+3}$, $S_{+2}$) | Value indicating the range which is masked |

After masking is started, it will be applied to eight values captured.

# Chapter 12 Setting an Ethernet Network

## Table of Contents

## 12.1 Functions

The Ethernet port on an AH500 series motion control module can exchange data with a network device through a general networking cable. The Ethernet port on an AH500 series motion control has the following functions.

- It can be connected to PMSoft. A program can be uploaded/downloaded and monitored.
- It can function as a standard Modbus TCP slave.

## 12.2 Specifications

- Ethernet connector

| Item | Specifications |
|---|---|
| Transmission type | Ethernet |
| Electrical isolation | 500 V DC |
| Connector | Removable connector (5.08 mm) |
| Transmission cable | Four communication cables |

- Communication

| Item | Specifications |
|---|---|
| Data type | TCP/IP |
| Serial transmission speed | 10 M/100 M (bit/second) |
| Maximum transmission distance | 100 meters |

## 12.3 Introduction of Parameters

SR808 and SR809: Ethernet IP address

[Description]

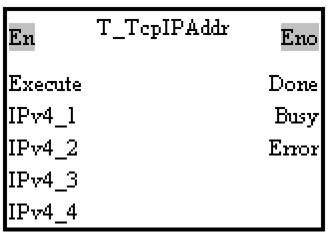
If users want to set the IP address of an AH500 series motion control module, two registers will used. The initial IP address of an AH500 series motion control module is 192.168.0.100.

| SR809 | | SR808 | |
|---|---|---|---|
| High byte | Low byte | High byte | Low byte |
| 192 | 168 | 0 | 100 |

## 12.4 Communication Function of PMSoft

- Setting an Ethernet IP

  The default Ethernet IP address of an AH500 series motion control module is 192.168.0.100. Users can change the Ethernet IP address of an AH500 series motion control module by means of SR808 and SR809. After an AH500 series motion control module is disconnected, its IP address will not be retained. Its IP address becomes 192.168.0.100 after it is supplied with power again. Alternatively, the users can set an IP address by means of a motion control function block. The motion control function block which can be used to set an IP address is shown below.




  The IPv4_1 input pin, the IPv4_2 input pin, the IPv4_3 input pin, and the IPv4_4 input pin are used to set an IP address. After the setting of the input pins is complete, the Done output pin will be ON.

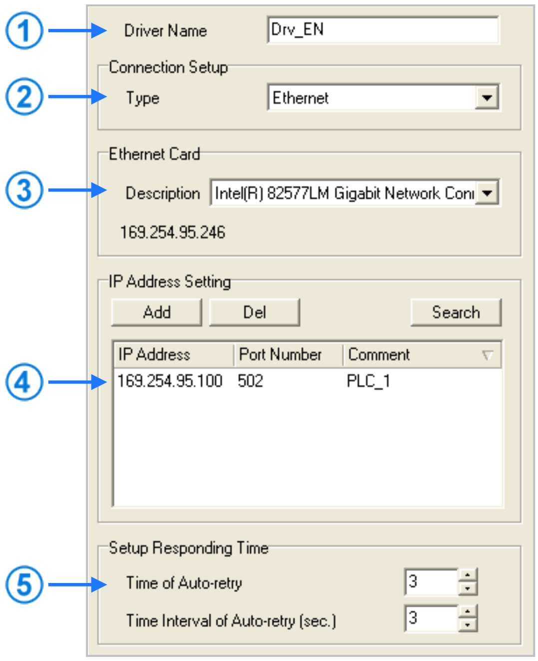
■ Connecting the Ethernet port on an AH500 series motion control module to PMSoft

If an AH500 series motion control module equipped with an Ethernet port is connected to PMSoft, users can upload/download and monitor a program through Ethernet.

- Wiring hardware

  Users can connect the Ethernet port on an AH500 series motion control module to an Ethernet port on a personal computer by means of a general networking cable. After an AH500 series motion control module is connected to a personal computer, the Ethernet connection LED indicator on the AH500 series motion control module will be ON. If the Ethernet connection LED indicator is not ON, users have to check whether the setting of the module or the personal computer is incorrect.

- Setting COMMGR




① Users can type a driver name in the **Driver Name** box.

② Select **Ethernet** in the **Type** drop-down list box in the **Connection Setup** section.

③ Select a network interface card in the **Description** drop-down list box. The IP address assigned to the network interface card selected is displayed in the lower left corner of the **Ethernet Card** section.

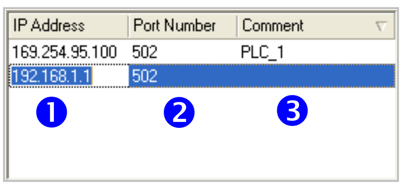
④ Owing to the characteristics of Ethernet, a computer can communicate with all devices on a network. Users can create the IP addresses of the devices connected to this driver in the **IP Address Setting** section.

➢ After users click **Add** to add a new IP address to the list of IP addresses in the **IP Address Setting** section, they can type related information in the **IP Address** cell, the **Port Number** cell, and the **Comment** cell.

❶ Users can type the IP address of a device connected in this cell.

❷ Users can type the communication port number specified.

❸ Users can type a comment in this cell.



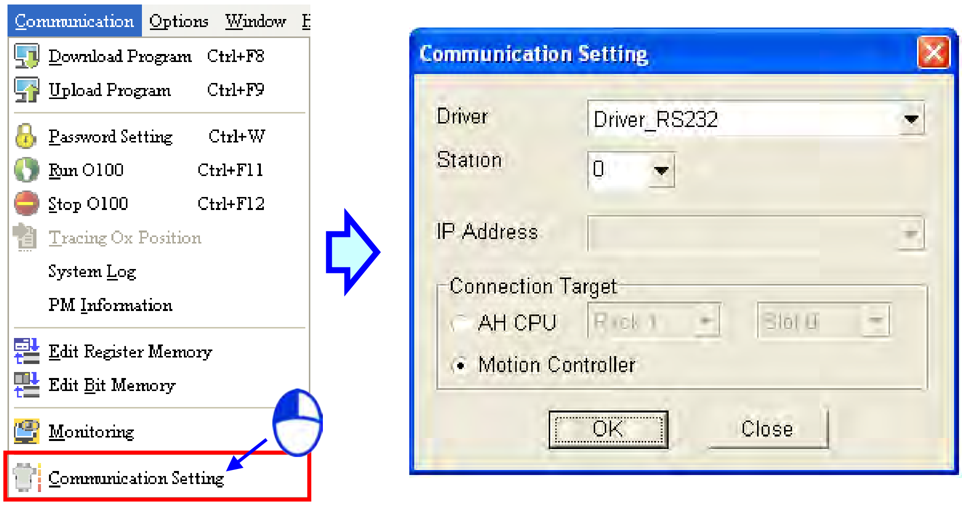
➢ After users select an IP address, they can click **Del** or press DEL on the keyboard to delete the IP address from the list.

⑤ Users can select the number of times the sending of a command is retried if a connection error occurs in the **Time of Auto-retry** box, and select an interval of retrying the sending of a command in the **Time Interval of Auto-retry** box.

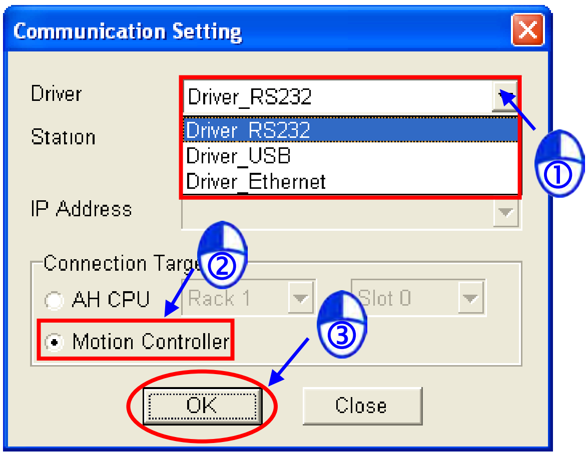
- Steps of setting PMSoft

Connect a computer to the Ethernet port on an AH500 series motion control module in the way described below, and then follow the steps described below.

(1) Start PMSoft, and then click **Communication Settings…** on the **Tools** menu.

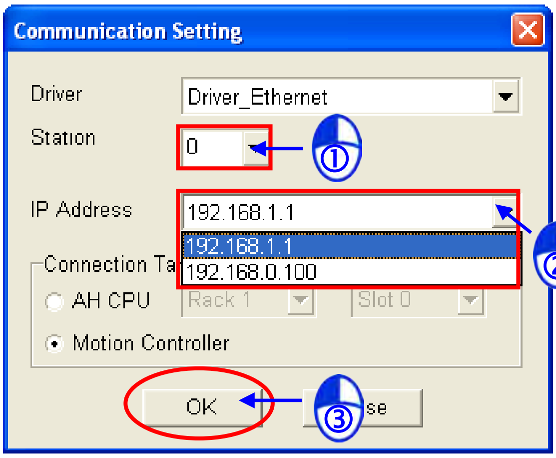



(2) Select a driver in the **Driver** drop-down list box. Before users create a connection between PMSoft and an AH500 series motion control module, they have to make sure that the driver is started in COMMGR. Select the **Motion Controller** option button, and click **OK**. The communication setting varies with the driver selected.



➢ **Ethernet**

Users have to select the station address of the AH500 series motion control module connected to the computer in the **Station** drop-down list box. If the station address selected is 0, a broadcast communication will be carried out. The users also have to select the IP address created in COMMGR in the **IP Address** drop-down list box.




## 12.5 Modbus Communication

- Setting an Ethernet IP address

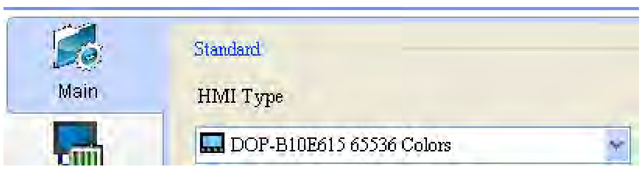
  The default Ethernet IP address of an AH500 series motion control module is 192.168.0.100. Users can change the Ethernet IP address of an AH500 series motion control module by means of SR808 and SR809, or by means of the motion control function block T_TcpIPAddr.

- An AH500 series motion control module can function as a Modbus TCP slave.

  If users connect an AH500 series motion control module by means of Ethernet, the AH500 series motion control module can function as a Modbus TCP slave. If an AH500 series motion control module is connected to a human-machine interface, the steps of setting the human-machine interface will be as follows.

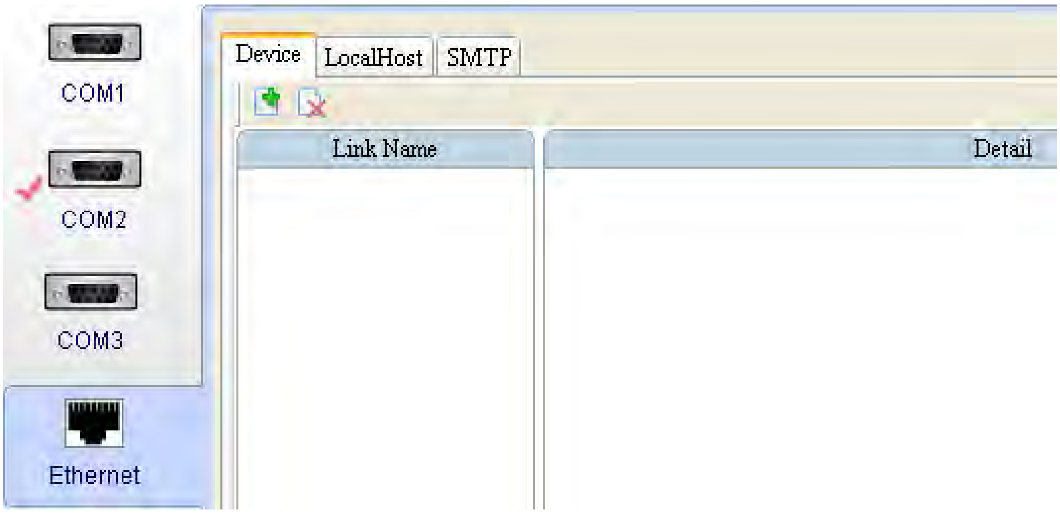
  1. Click **Configuration…** on the **Options** menu.


  2. Click the **Main** tab, and then select **DOP-B10E615 65536 Colors** in the **HMI Type** drop-down list box.

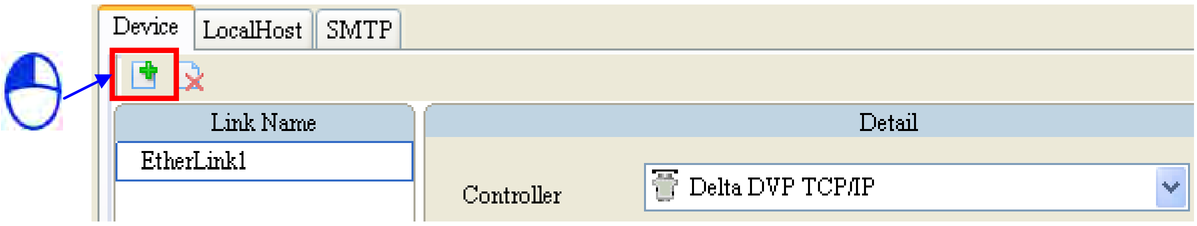


3. Click **Communication Setting** on the **Options** menu.

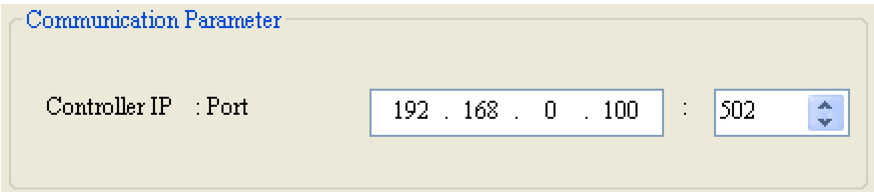

4. Click the **Ethernet** tab.




5. After users click , they have to type a link name in the **Link Name** box, and select **Delta DVP TCP/IP** in the **Controller** drop-down list box.




6. The users have to set the IP address of the AH500 series motion control module in the **Communication Parameter** section.




7. After the users select the link name created in step 5 in the **Input** window for an element, they can operate the memory defined by the element by means of Ethernet.


Twelve connections at most can be created. Twelve connections can be created simultaneously.

## 12.6 Troubleshooting

| Problem | Remedy |
|---|---|
| The Ethernet connection LED indicator on an AH500 series motion control module is not ON. | Check whether a networking cable is connected to the AH500 series motion control module correctly. |
| An AH500 series motion can not be connected to PMSoft. | Check whether the IP address of the AH500 series motion control module is correct. If the IP address of the AH500 series motion control module and the IP address of the driver created are not in the same domain, the AH500 series motion can not be connected to PMSoft. |
| A Modbus TCP connection can not be created. | 1. Check whether the setting of the IP address of the server to which a client is connected is correct. The number of Modbus TCP connections can not be greater than 12.<br>2. Check whether the setting of a station address is correct. If users do not know the station address of the AH500 series motion control module which is connected, they can set the station address to 0. |

MEMO

12

# Chapter 13 Expansion Storage Device 13

## Table of Contents

## 13.1 Functions

An AH500 series motion control module is embedded with a memory card slot for external memory extension. The slot is compatible with the memory card formats FAT16 and FAT 32, and the maximum storage is 4 GB. The four functions of the memory card for an AH500 series motion control module are described below.

1. G code reading and execution
2. Device backup and restoration
3. Program backup and restoration
4. Firmware update

The files for the four functions above are saved in the following paths in a memory card.

1. \AHMotion\Gcode\
2. \AHMotion\Device\
3. \AHMotion\Program\
4. \AHMotion\bin\

13

## 13.2 Parameters

- List of parameters

| Parameter | Function |
|---|---|
| SR200 | Start address of an M device for a memory card backup |
| SR201 | End address of an M device for a memory card backup |
| SR202 | Start address of a timer for a memory card backup |
| SR203 | End address of a timer for a memory card backup |
| SR204 | Start address of a 16-bit counter for a memory card backup |
| SR205 | End address of a 16-bit counter for a memory card backup |
| SR206 | Start address of a 32-bit counter for a memory card backup |
| SR207 | End address of a 32-bit counter for a memory card backup |
| SR208 | Start address of an S device for a memory card backup |
| SR209 | End address of an S device for a memory card backup |
| SR210 | Start address of a D device for a memory card backup |
| SR211 | End address of a D device for a memory card backup |
| SR212 | Start address of a W device for a memory card backup |
| SR213 | End address of a W device for a memory card backup |
| SR214 | Control register for a memory card backup |

Description of parameters:

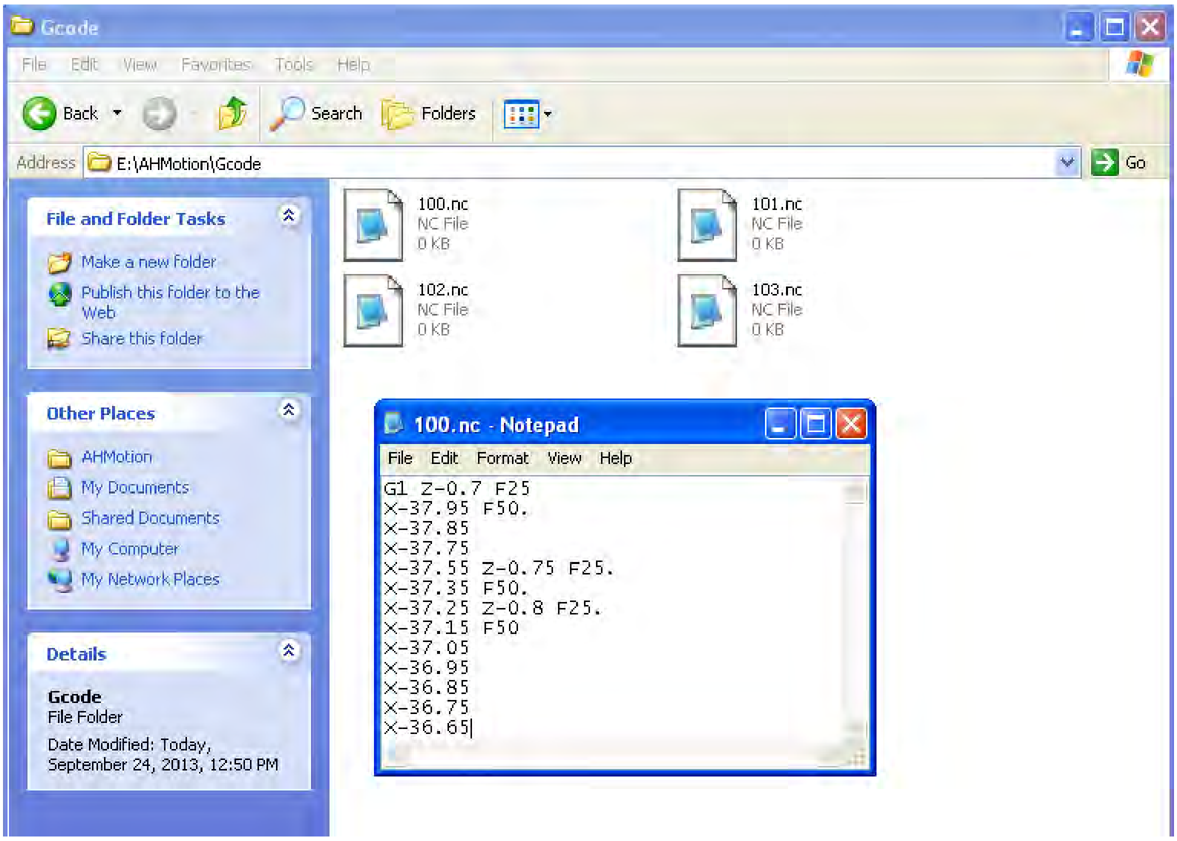
1. **SR200, SR201: Start/End address of an M device for a memory card backup**
   [Description]
   To perform restoration by a memory card, the parameters are used to set the start/end address of an M device. If the restoration addresses include SM devices, the SM devices will not be restored, but the rest of the devices within the restoration range will still be restored.
2. **SR202, SR203: Start/End address of a timer for a memory card backup**
   [Description]
   To perform restoration by a memory card, the parameters are used to set the start/end address of a timer.
3. **SR204, SR205: Start/End address of a 16-bit counter for a memory card backup**
   [Description]
   To perform restoration by a memory card, the parameters are used to set the start/end address of a 16-bit counter.

4. **SR206, SR207: Start/End address of a 32-bit counter for a memory card backup**
   [Description]
   To perform restoration by a memory card, the parameters are used to set the start/end address of a 32-bit counter
5. **SR208, SR209: Start/End address of an S device for a memory card backup**
   [Description]
   To perform restoration by a memory card, the parameters are used to set the start/end address of an S device.
6. **SR210, SR211: Start/End address of a D device for a memory card backup**
   [Description]
   To perform restoration by a memory card, the parameters are used to set the start/end address of a D device. If the restoration addresses includes SR devices, the SR devices will not be restored, but the rest of the devices within the restoring range will still be restored.
7. **SR212, SR213: Start/End address of a W device for a memory card backup**
   [Description]
   To perform restoration by a memory card, the parameters are used to set the start/end address of a W device.
8. **SR214:Control register for a memory card backup**
   [Description]
   Definition of registers:

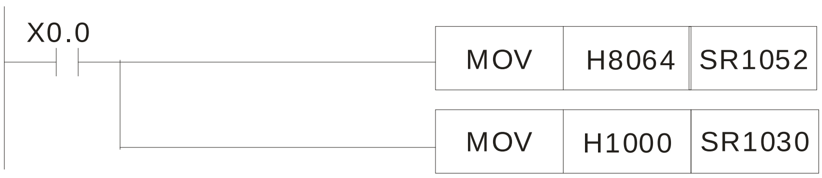
| Bit | Name | Description | Reset value |
|---|---|---|---|
| [12-0] | File name | For a backup, it stands for the file name stored in a memory card.<br>For restoration, it stands for the file name retrieved from a memory card | 0x0000 |
| 13 | Control command | To restore devices or programs, set the value of bit 13 to 0.<br>To backup devices or programs, set the value of bit 13 to 1.<br>The backup of devices includes every device. | 0 |
| 14 | Program backup/restoration | To backup or restore programs, set the value of bit 14 to 1.<br>When the operation is completed, the value of bit 14 is automatically reset to 0. | 0 |
| 15 | Device backup/restoration | To backup or restore devices, set the value of bit 15 to 1.<br>When the operation is completed, the value of bit 15 is automatically reset to 0. | 0 |

## 13.3 Reading and Executing G-codes

The regular storage path for NC files is under the root directory \AHMotion\Gcode\ of a memory card. The files are named 100~199 with a sub-name of NC (regardless of capitalization). Format of the files is the same as text files. Each folder contains up to 100 G code files ready for use.




The 100 files stand for motion subroutines Ox100~199 (subsequent to the built-in subroutines Ox0~Ox99). The file name 100 refers to Ox100, the file name 101 refer to Ox101, and so forth. So the file name 199 refers to Ox199. As shown in the figure below, when Ox100 is executed, the AH500 series motion control module will first open and access the file \AHMotion\Gcode\1.NC and then proceed with code transfer and processing.



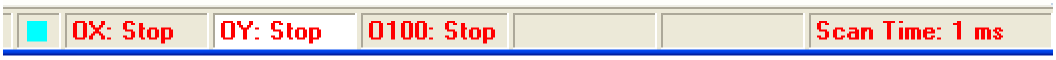

## 13.4 Device Backup and Restoration

An AH500 series motion control module provides the use of a memory card as the external storage space for device backup and restoration.

### 13.4.1 Backup

The operation procedures for device backup are described below,

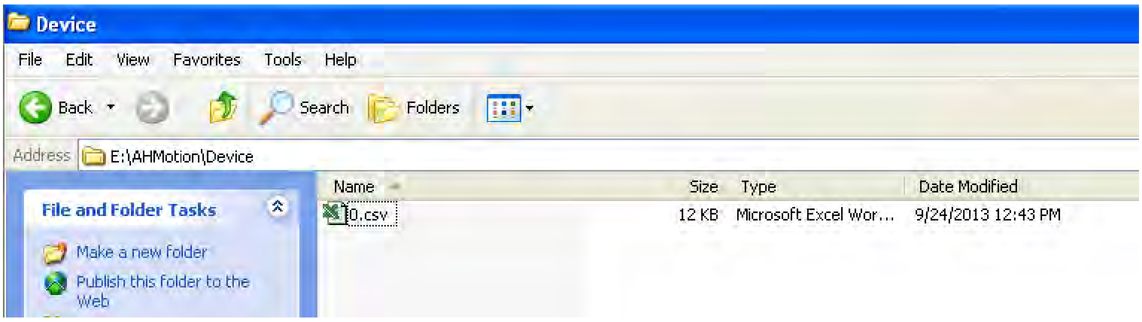
1. Monitor AH20MC-5A, AH10PM-5A, or AH15PM-5A by PMSoft, and stop the operation under the monitoring. Alternatively, set SM072 to 0 to stop 0100.




2. Set SR214 by a device monitoring table or by external communication.
   - Set bit 12~bit 0 in SR214. The values of bit 12~bit 0 in SR124 represents a backup file name.
   - To backup, set bit 13 in SR214 to 1.
   - To backup devices, set bit 15 in SR214 to 1.

| Device No. | Radix | Value | Comment |
|---|---|---|---|
| SR214 | b16 | 1010_0000_0000_0000 | |
| | | | |
| | | | |
| | | | |

For example, if the value in SR214 is A000H, devices will be backed up, the name of the backup file will be 0, and the file 0.csv can be found in \AHMotion\Device\ in a memory card.




The CSV file embodies the data in the devices. As it includes every device, no individual device is to be excluded from being backup.

Device parameters can be changed via the CSV file in a general Windows environment. The format of the CSV file is as shown below.

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | AHMotion SD Format V0.3 | | | | | | | | | |
| 2 | PLC Type : | | | | | | | | | |
| 3 | PLC FWVer : | | | | | | | | | |

The first line indicates the format version of the memory cards for an AH500 series motion control module.

PLC Type: The model name of an AH500 series motion control module

PLC FWVer: Firmware version of a model

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 5 | M | | | | | | | | | |
| 6 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 7 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 8 | 0 | 0 | 1 | ... | | | | | | |
| 9 | | | | | | | | | | |
| 10 | T | | | | | | | | | |
| 11 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 12 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 13 | 0 | 0 | 1 | ... | | | | | | |
| 14 | | | | | | | | | | |
| 15 | C16b | | | | | | | | | |
| 16 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 17 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 18 | 0 | 0 | 1 | ... | | | | | | |
| 19 | | | | | | | | | | |
| 20 | C32b | | | | | | | | | |
| 21 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 22 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 23 | 0 | 0 | 1 | ... | | | | | | |
| 24 | | | | | | | | | | |
| 25 | S | | | | | | | | | |
| 26 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 27 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 28 | 0 | 0 | 1 | ... | | | | | | |
| 29 | | | | | | | | | | |
| 30 | C16W | | | | | | | | | |
| 31 | 1234 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 32 | 0 | 0 | 4444 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 33 | 0 | 0 | 0 | ... | | | | | | |
| 34 | | | | | | | | | | |
| 35 | C32W | | | | | | | | | |
| 36 | 0 | 0 | 0 | 44s | 0 | 0 | 0 | 0 | 0 | 0 |
| 37 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 38 | 0 | 0 | 0 | ... | | | | | | |
| 39 | | | | | | | | | | |
| 40 | D | | | | | | | | | |
| 41 | 0 | 0 | 0 | 44s | 0 | 0 | 0 | 0 | 0 | 0 |
| 42 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 43 | 0 | 0 | 0 | ... | | | | | | |
| 44 | | | | | | | | | | |
| 45 | W | | | | | | | | | |
| 46 | 0 | 0 | 0 | 44s | 0 | 0 | 0 | 0 | 0 | 0 |
| 47 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 48 | 0 | 0 | 0 | ... | | | | | | |

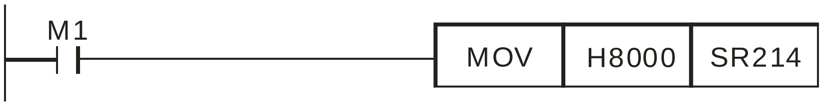
A table of values in all the devices is below the description of a PLC type and a firmaware version is. Users can modify the contents of the devices through this table in which M stands for an M device; T stands for a timer; C16b and C16W stand for the state and value of a 16-bit counter; C32b and C32W stand for the state and value of a 32-bit counter;D stands for a D device; and W stands for a W device.

### 13.4.2 Restoration

Before restoring devices, the file to be restored should be placed under the directory \AHMotion\Device\ of a memory card. The operation procedures for restoration are described below.

1. Set ranges for device restoration. Use SR200~SR213 to define restoration ranges.
2. Set SR214.
    - Set bit 12~bit 0 in SR214. The values of bit 12~bit 0 in SR214 represent a backup file name.

- ◆ To restore, set bit 13 in SR214 to 0.
- ◆ To backup devices, set bit15 in SR214 to 1.



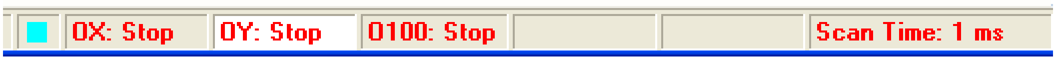

For example, if the value in SR214 is 8000H, the file \AHMotion\Device\0.csv in a memory card will be restored to the motion control module.

# 13.5 Program Backup and Restoration

## 13.5.1 Backup

The operation for a program backup is described below,

1. Monitor an AH500 series motion control module by PMSoft, and stop the operation under the monitoring. Alternatively, set SM072 to 0 to stop O100.




2. Set SR214 by a device monitoring table or external communication.
   - ◆ Set bit 12~bit 0 in SR214 to bit 0. The values of bit 12~bit 0 in SR214 represent a backup file name.
   - ◆ To backup, set bit 13 in SR214 to 1.
   - ◆ To backup the program in the AH500 series motion control module, set bit 14 in SR214 to 1.

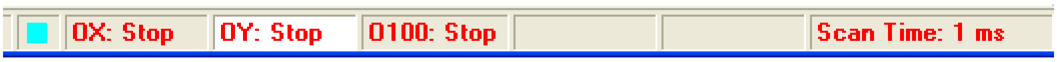
| Device No. | Radix | Value | Comment |
|---|---|---|---|
| SR214 | b16 | 0110_0000_0000_0000 | |
| | | | |
| | | | |
| | | | |

After the program backup is completed, a .raw file which includes the password for the backup program will be generated in \AHMotion\Program\ in a memory card. The table above shows that the name of the .raw file is 0.raw.

### 13.5.2 Restoration

When supplied with power, an AH500 series motion control module will automatically scan the \AHMotion\Program\ directory for program restoration. If 0.raw file exists in the directory, the file will be restored to the motion control module automatically. Otherwise, the file can also be restored through the following procedures.

1. Monitor the AH500 series motion control module by PMSoft, and stop the operation under the monitoring. Alternatively, set SM072 to 0 to stop O100.




2. Set SR214 by a device monitoring table or external communication.
   - ◆ Set bit12~bit 0 in SR214. The values of bit 12~bit 0 in SR214 represent a backup filename.
   - ◆ To restore, set bit13 in SR214 to 0.
   - ◆ To backup the program in the motion control module, set bit 14 in SR214 to 1.

13

| Device No. | Radix | Value | Comment |
|---|---|---|---|
| SR214 | b16 | 0100_0000_0000_0001 | |
| | | | |
| | | | |
| | | | |

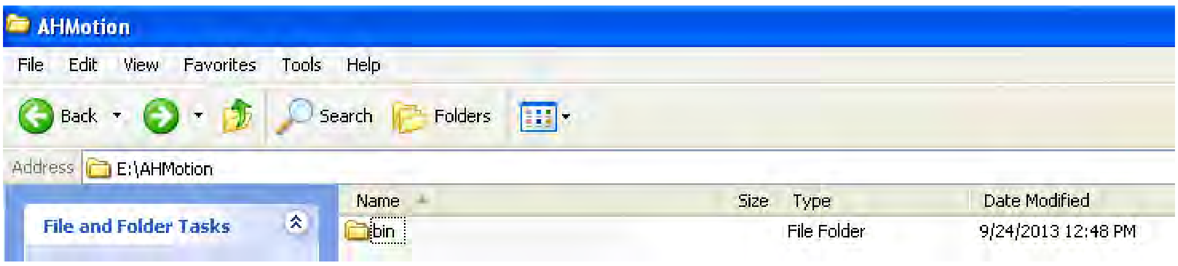
The table above shows how the file 1.raw is restored to the motion control module.
All the restoration operation includes the copies of program passwords. When a file is restored to an AH500 series motion control module, the RUN LED indicator on the motion control module blinks. When the blinking stops, the restoration is completed.

## 13.6 Updating Firmware

The firmware update of an AH500 series motion control module can be performed by memory cards. The operation procedures are as follows.

1. Flick the update switch to the firmware update mode.


2. Create the folder AHMotion\bin under the root directory of a memory card.



3. Save different .bin files in the folder AHMotion\bin according to the model of an AH500 series motion control module.
   - i AH20MC-5A: C5A20MC.bin and G5A20MC.bin are saved in the folder.
   - ii AH10PM-5A: C5A10PM.bin and G5A10PM.bin are saved in the folder.
   - iii AH15PM-5A: C5A15PM.bin and G5A15PM.bin are saved in the folder.
   - iv AH05PM-5A: C5A05PM.bin and G5A05PM.bin are saved in the folder.
4. Insert a memory card into the memory card slot, and supply the module with power. When the RUN LED indicator blinks, the firmware of the motion control module is updated. After the update is completed, the RUN LED indicator will be ON. If an error occurs during the update, the ERROR LED indicator will be ON.
5. After the update is complete, move the switch to its original direction.


After the update is completed, supply the motion control module with power again, and run the motion control module.

**MEMO**

# Chapter 14 DMCNET

## Table of Contents

## 14.1 Functions

AH20MC-5A is equipped with a DMCNET port. A DMCNET is Delta motion control communication. It is a real-time system. It only takes 1 millisecond to update the commands sent to the twelve axes in AH20MC-5A. There are several modes of returning home. The twelve axes in AH20MC-5A can move synchronously. They are divided into four groups so that three-axis helical/linear interpolation can be used. They are divided into six groups so that two-axis linear/circular interpolation can be used.

- A DMCNET supports twelve Delta ASDA-A2-F AC servo drives.
- Users can write the values of parameters into a servo drive and read the values of parameters from the servo drive by means of a DMCNET.
- User can instruct an axis to return home by means of a DMCNET, and axes can move synchronously by means of a DMCNET.

## 14.2 Specifications

- Connector

| Item | Specifications |
|---|---|
| Transmission type | DMCNET |
| Electrical isolation | 500 V DC |
| Connector | Removable connector (5.08 mm) |
| Transmission cable | Four communication cables |



- Communication

| Item | Specifications |
|---|---|
| Data type | Static frame and dynamic frame |
| Serial transmission speed | There are two channels. The serial transmission speed of a channel is 10 megabits per second. |
| Maximum transmission distance | 20 meters (A 120 ohm terminal resistor is required.) |

## 14.3 Parameters

■ Parameter table

| Parameter | Axis 1 | Axis 2 | Axis 3 | Axis 4 | Axis 5 | Axis 6 | Axis 7 | Axis 8 |
|---|---|---|---|---|---|---|---|---|
| Command sent to the servo drive for the axis specified on a DMCNET | SR1072 | SR1172 | SR1272 | SR1372 | SR1472 | SR1572 | SR1672 | SR1772 |
| Status of the servo drive for the axis specified on a DMCNET | SR1073 | SR1173 | SR1273 | SR1373 | SR1473 | SR1573 | SR1673 | SR1773 |
| Servo drive error code[*1] | SR1074 | SR1174 | SR1274 | SR1374 | SR1474 | SR1574 | SR1674 | SR1774 |
| Servo drive error code[*2] | SR1075 | SR1175 | SR1275 | SR1375 | SR1475 | SR1575 | SR1675 | SR1775 |
| Writing data into the servo drive for the axis specified on a DMCNET/Reading data from the servo drive for the axis specified on a DMCNET[*1] | SR1076 | SR1176 | SR1276 | SR1376 | SR1476 | SR1576 | SR1676 | SR1776 |
| Value written into the servo drive for the axis specified on a DMCNET/Value read from the servo drive for the axis specified on a DMCNET[*2] | SR1077 | SR1177 | SR1277 | SR1377 | SR1477 | SR1577 | SR1677 | SR1777 |
| Parameter position in the servo drive for the axis specified on a DMCNET<br>Way in which the axis specified on a DMCNET returns home | SR1078 | SR1178 | SR1278 | SR1378 | SR1478 | SR1578 | SR1678 | SR1778 |

| Parameter | Axis 9 | Axis 10 | Axis 11 | Axis 12 | Axis 13 | Axis 14 | Axis 15 | Axis 16 |
|---|---|---|---|---|---|---|---|---|
| Command sent to the servo drive for the axis specified on a DMCNET | SR1872 | SR1972 | SR2072 | SR2172 | SR2272 | SR2372 | SR2472 | SR2572 |
| Status of the servo drive for the axis specified on a DMCNET | SR1873 | SR1973 | SR2073 | SR2173 | SR2273 | SR2373 | SR2473 | SR2573 |
| Servo drive error code[*1] | SR1874 | SR1974 | SR2074 | SR2174 | SR2274 | SR2374 | SR2474 | SR2574 |
| Servo drive error code[*2] | SR1875 | SR1975 | SR2075 | SR2175 | SR2275 | SR2375 | SR2475 | SR2575 |
| Writing data into the servo drive for the axis specified on a DMCNET/Reading data from the servo drive for the axis specified on a DMCNET[*1] | SR1876 | SR1976 | SR2076 | SR2176 | SR2276 | SR2376 | SR2476 | SR2576 |
| Value written into the servo drive for the axis specified on a DMCNET/Value read from the servo drive for the axis specified on a DMCNET[*2] | SR1877 | SR1977 | SR2077 | SR2177 | SR2277 | SR2377 | SR2477 | SR2577 |
| Parameter position in the servo drive for the axis specified on a DMCNET<br>Way in which the axis specified on a DMCNET returns home | SR1878 | SR1978 | SR2078 | SR2178 | SR2278 | SR2378 | SR2478 | SR2578 |

*1. Low word of the parameter

*2. High word of the parameter

- Introduction of the parameters
  - SR1072, SR1172, SR1272, SR1372, SR1472, SR1572, SR1672, SR1772, SR1872, SR1972, SR2072, SR2172, SR2272, SR2372, SR2472, SR2572: Command sent to the servo drive for the axis specified on a DMCNET

[Description]

| 1st axis | | 2nd axis | | 3rd axis | | 4th axis | |
|---|---|---|---|---|---|---|---|
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR1072 | - | SR1172 | - | SR1272 | - | SR1372 |
| **5th axis** | | **6th axis** | | **7th axis** | | **8th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR1472 | - | SR1572 | - | SR1672 | - | SR1772 |
| **9th axis** | | **10th axis** | | **11th axis** | | **12th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR1872 | - | SR1972 | - | SR2072 | - | SR2172 |
| **13th axis** | | **14th axis** | | **15th axis** | | **16th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR2272 | - | SR2372 | - | SR2472 | - | SR2572 |

Usage of the registers:

| Value (Hex) | Command sent to the servo drive for the axis specified on a DMCNET | Value (Hex) | Command sent to the servo drive for the axis specified on a DMCNET |
|---|---|---|---|
| **xxx1** | Setting the servo drive used to OFF | **x1xx** | Writing the value of a 16-bit parameter into the servo drive used |
| **xxx2** | Setting the servo drive used to ON | **x2xx** | Writing the value of a 32-bit parameter into the servo drive used |
| **xxx3** | Resetting NMT | **x3xx** | Reading the value of a parameter in the servo drive used |
| **xxx4** | Resetting an error flag | **x4xx** | Instructing the servo drive used to return home by means of a DMCNET |
| | | **x5xx** | Setting a DMCNET motion mode by means of DMCNET |

- Setting the servo drive used to OFF/ON: User can set the servo drive used to OFF or ON. After the servo drive used is set, the status of the servo drive will be shown by a special data register.
- Resetting NMT: The servo drive used can be instructed to reset DMCNET communication. DMCTNET communication can be reset, whether the servo drive used is connected.
- Resetting an error flag: After an error occurs in the servo drive used, users can reset the error flag for the error. The error flag for the error occurring in a servo drive can be reset only when the servo drive is connected.
- Writing the value of a 16-bit/32-bit parameter into the servo drive used: The servo drive used determines whether a parameter is a 16-bit parameter or a 32-bit parameter. When the value of a parameter in the servo drive used is read, it is not necessary to consider whether the parameter is a 16-bit parameter or a 32-bit parameter.
- A DMCNET can be used to instruct the servo drive used to return home, and can be used to set a DMCNET motion mode.
  - If a DMCNET is used to instruct a servo drive to return home, users have to set the mode of returning home.
  - After a DMCNET is used to set a DMCNET motion mode, users have to control a servo drive by means of uniaxial motion.

◆ SR1073, SR1173, SR1273, SR1373, SR1473, SR1573, SR1673, SR1773, SR1873, SR1973, SR2073, SR2173, SR2273, SR2373, SR2473, SR2573: Status of the servo drive for the axis specified on a DMCNET

[Description]

| 1st axis | | 2nd axis | | 3rd axis | | 4th axis | |
|---|---|---|---|---|---|---|---|
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR1073 | - | SR1173 | - | SR1273 | - | SR1373 |
| **5th axis** | | **6th axis** | | **7th axis** | | **8th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR1473 | - | SR1573 | - | SR1673 | - | SR1773 |
| **9th axis** | | **10th axis** | | **11th axis** | | **12th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR1873 | - | SR1973 | - | SR2073 | - | SR2173 |
| **13th axis** | | **14th axis** | | **15th axis** | | **16th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR2273 | - | SR2373 | - | SR2473 | - | SR2573 |

Usage of the registers:

| Value (Hex) | Status of the servo drive for the axis specified on a DMCNET | Value (Hex) | Status of the servo drive for the axis specified on a DMCNET |
|---|---|---|---|
| **xxx0** | The servo drive used is disconnected. | **x0xx** | A control command is done. |
| **xxx1** | The servo drive used is OFF. | **x1xx** | A control command is running. |
| **xxx2** | The servo drive used is ON. | **x2xx** | Control command error |
| | | | |
| **xx1x** | An error occurs in the servo drive used. | | |

- If the status of a servo drive is that the servo drive is disconnected, the servo drive is not connected. If a servo drive is actually connected, but its status is that it is disconnected, it may be because the module used can not make sure of the status of the servo drive, and users can reset the servo drive.
  1. **Instruct the servo drive used to reset NMT.**
  2. **After the servo drive used is instructed to reset DMCNET communication, the status of the servo drive used will be that the servo drive is OFF.**
- The command sent to a servo drive determines whether the status of the servo drive is that the servo drive is ON/OFF.
- Bit 4 in SR1073 (SR1173, SR1273, SR1373…) is an error flag. If bit 4 in SR1073 (SR1173, SR1273, SR1373…) is set to 1, users can reset it by means of a command.
- After the value of bit 11~bit 8 in SR1072 (SR1172, SR1272, SR1372…) becomes 1/2/3/4/5, the status of the command sent to the servo drive used will be indicated by bit 11~bit 8 in SR1073 (SR1173, SR1273, SR1373…).

14

◆ SR1075, SR1074, SR1175, SR1174, SR1275, SR1274, SR1375, SR1374, SR1475, SR1474, SR1575, SR1574, SR1675, SR1674, SR1775, SR1774, SR1875, SR1874, SR1975, SR1974, SR2075, SR2074, SR2175, SR2174, SR2275, SR2274, SR2375, SR2374, SR2475, SR2474, SR2575, SR2574: Servo drive error code

[Description]

| 1st axis | | 2nd axis | | 3rd axis | | 4th axis | |
|---|---|---|---|---|---|---|---|
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| SR1075 | SR1074 | SR1175 | SR1174 | SR1275 | SR1274 | SR1375 | SR1374 |
| **5th axis** | | **6th axis** | | **7th axis** | | **8th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| SR1475 | SR1474 | SR1575 | SR1574 | SR1675 | SR1674 | SR1775 | SR1774 |
| **9th axis** | | **10th axis** | | **11th axis** | | **12th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| SR1875 | SR1874 | SR1975 | SR1974 | SR2075 | SR2074 | SR2175 | SR2174 |
| **13th axis** | | **14th axis** | | **15th axis** | | **16th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| SR2275 | SR2274 | SR2375 | SR2374 | SR2475 | SR2474 | SR2575 | SR2574 |

Servo drive error code: After an error occurs in a servo drive, the servo drive will send an error code. Please refer to the operation manual for the servo drive used for more information about the definitions of error codes and troubleshooting. After the error flag for an error is reset, the error code for the error will be cleared to 0.



◆ SR1077, SR1076, SR1177, SR1176, SR1277, SR1276, SR1377, SR1376, SR1477, SR1476, SR1577, SR1576, SR1677, SR1676, SR1777, SR1776, SR1877, SR1876, SR1977, SR1976, SR2077, SR2076, SR2177, SR2176, SR2277, SR2276, SR2377, SR2376, SR2477, SR2476, SR2577, SR2576: Value written into the servo drive for the axis specified on a DMCNET/Value read from the servo drive for the axis specified on a DMCNET

[Description]

| 1st axis | | 2nd axis | | 3rd axis | | 4th axis | |
|---|---|---|---|---|---|---|---|
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| SR1077 | SR1076 | SR1177 | SR1176 | SR1277 | SR1276 | SR1377 | SR1376 |
| **5th axis** | | **6th axis** | | **7th axis** | | **8th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| SR1477 | SR1476 | SR1577 | SR1576 | SR1677 | SR1676 | SR1777 | SR1776 |
| **9th axis** | | **10th axis** | | **11th axis** | | **12th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| SR1877 | SR1876 | SR1977 | SR1976 | SR2077 | SR2076 | SR2177 | SR2176 |
| **13th axis** | | **14th axis** | | **15th axis** | | **16th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| SR2277 | SR2276 | SR2377 | SR2376 | SR2477 | SR2476 | SR2577 | SR2576 |

Value written into the servo drive for the axis specified on a DMCNET/Value read from the servo drive for the axis specified on a DMCNET: Users can read data from a servo drive by means of DMCNET, and write data into a servo drive by means of a DMCNET. If an error occurs in the reading/writing of data, an error code will be stored in (SR1077, SR1076) ((SR1177, SR1176), (SR1277, SR1276), (SR1377, SR1376)…).

◆ SR1078, SR1178, SR1278, SR1378, SR1478, SR1578, SR1678, SR1778, SR1878, SR1978, SR2078, SR2178, SR2278, SR2378, SR2478, SR2578: Parameter position in the servo drive for the axis specified on a DMCNET/Way in which the axis specified on a DMCNET returns home

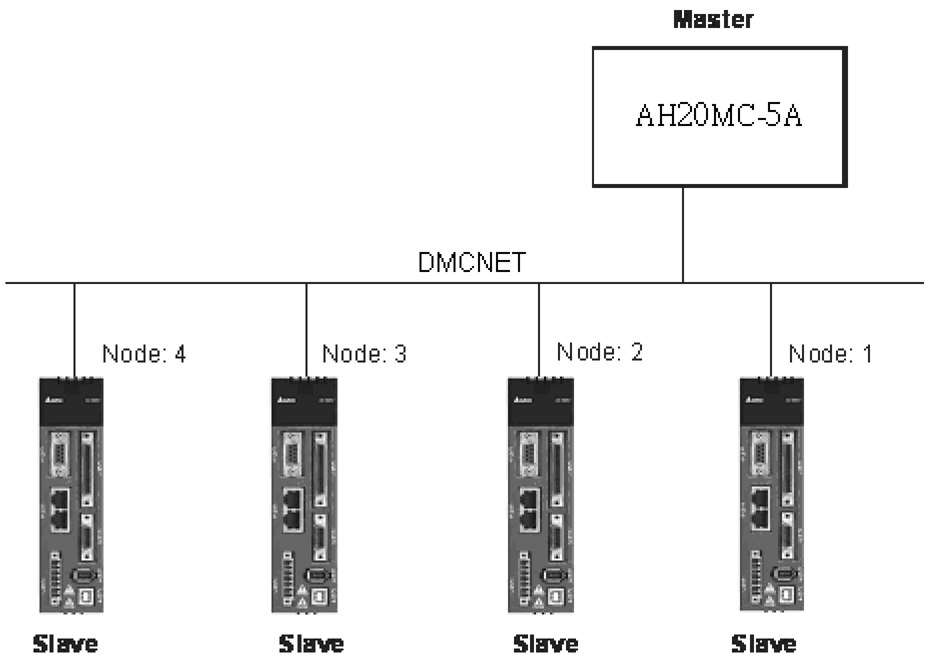
[Description]

| 1st axis | | 2nd axis | | 3rd axis | | 4th axis | |
|---|---|---|---|---|---|---|---|
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR1078 | - | SR1178 | - | SR1278 | - | SR1378 |
| **5th axis** | | **6th axis** | | **7th axis** | | **8th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR1478 | - | SR1578 | - | SR1678 | - | SR1778 |
| **9th axis** | | **10th axis** | | **11th axis** | | **12th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR1878 | - | SR1978 | - | SR2078 | - | SR2178 |
| **13th axis** | | **14th axis** | | **15th axis** | | **16th axis** | |
| **HW** | **LW** | **HW** | **LW** | **HW** | **LW** | **HW** | **LW** |
| - | SR2278 | - | SR2378 | - | SR2478 | - | SR2578 |

- Parameter position in the servo drive for the axis specified on a DMCNET: The value of the high byte in SR1078 (SR1178, SR1278, SR1378…) indicates a group number, and the value of the low byte in SR1078 (SR1178, SR1278, SR1378…) indicates a parameter number. If users want to use P1-44, the value of the high byte in SR1078 (SR1178, SR1278, SR1378…) will be 1, and value of the low byte in SR1078 (SR1178, SR1278, SR1378…) will be 44, that is, the value in SR1078 (SR1178, SR1278, SR1378…) will be 16#012C.
- Way in which the axis specified on a DMCNET returns home: Users can set the way in which the axis specified on a DMCNET returns home. The value in SR1078 (SR1178, SR1278, SR1378…) is in the range of 1 to 35. Please refer to section 14.6 for more information about modes of returning home.

## 14.4 DMCNET Connection

■ Setting a connection



- Hardware configuration
  - Wiring hardware

    When users wire DMCNET hardware, they have to use a Delta DMCNET cable, and install a Delta DMCNET terminal resistor ASD-TR-DM0008 in the whole system created to make communication stable. The length of a connection can not exceed 30 meters.

DMCNET terminal resistor: ASD-TR-DM0008

  - Checking the firmware version of a servo drive
    1. Check whether the value of P0-00 in a servo drive indicates a version which is 1.744 or above. 7 represents ASDA-A2-F. If 7 does not appear, users have to replace the servo drive.
    2. If the value of P0-00 is 1.744, users have to check whether the value of P5-00 indicates a version which is 873 or above.
  - Setting an ASDA-A2 series AC servo drive

    Before users create a DMCNET connection, they have to set a servo drive to DMCNET mode. The steps of setting a servo drive to DMCNET mode are as follows.
    1. Set P1-01 in an ASDA-A2 series AC servo drive to 16#0B. (Set an ASDA-A2 series AC servo drive to DMCNET mode.)
    2. Set P3-00 in the ASDA-A2 series AC servo drive. The value of P3-00 in an ASDA-A2 series AC servo drive indicates the node ID of the ASDA-A2 series AC servo drive. It is in the range of 16#01 to 16#0C. The node ID of an ASDA-A2 series AC servo drive is in the range of 1 to 12. The node ID of a servo drive can not be the same as the node ID of another servo drive. Node ID 1 represents the first axis, node ID 2 represents the second axis, and node ID 3~node ID 12 represent the third axis~the twelfth axis. There must be an ASDA-A2 series AC servo drive whose node ID is 1 on a DMCNET. If there are two ASDA-A2 series AC servo drives, the node ID of one servo drive must be 1, the node ID of the other servo drive must be in the range of 2~12. If there is only one ASDA-A2 series AC servo drive, the node ID of the ASDA-A2 series AC servo drive must be 1, otherwise a DMCNET connection can not be created.
    3. Set P3-01 to 16#0203. (Set P3-01 to Delta DMCNET mode.)
    4. Set P0-02 to 16#120. (Check the status of the connection created.)
- Checking the status of a connection

  Users can check whether a servo drive is connected by means of bit 0~bit 3 in SR1073 (SR1173, SR1273…). If the status of a servo drive is that the servo drive is not connected, and the module used does not find the servo drive, users can follow the steps below.
  1. Instruct the servo drive used to reset NMT by means of bit 0~bit 3 in SR1072 (SR1172, SR1272…).
  2. OO-OO is shown on the display of the servo drive. The servo drive is resetting NMT.
  3. After the servo drive reset NMT, the users can check whether the status of the servo drive is that the servo drive is ON/OFF by means of bit 0~bit 3 in SR1073 (SR1173, SR1273…).

Users can check the status of a servo drive by setting P0-02 in the servo drive to 16#120. After P0-02 in a servo drive is set to 16#120, users can view the value shown on the display of the servo drive.

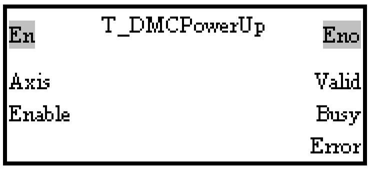
| Value shown on the display of a servo drive | Description |
|---|---|
| 16#06 | The servo drive is waiting to connect to a DMCNET. |
| 16#80 | The servo drive connects to a DMCNET successfully. |
| 16#111 | The servo drive is connected to an AH500 series motion control module. |

- Starting/Stopping a servo drive
  - ◆ Using special data registers

    If the status of a servo drive is that the servo drive is OFF, users can start or stop the servo drive.

    1. If the value of bit 0~bit 3 in SR1072 (SR1172, SR1272…) is 1, the servo drive used will be stopped. If the value of bit 0~bit 3 in SR1072 (SR1172, SR1272…) is 2, the servo drive used will be started.
    2. After a command is sent to a servo drive, users can check whether the servo drive is started by means of bit 0~bit 3 in SR1073 (SR1173, SR1273…).
  - ◆ Using a motion control function block

    Users can start/stop a servo drive by means of the motion control function block T_DMCPowerUp.




Please refer to Chapter 5 for more information about the input pins and the output pins in the motion control function block T_DMCPowerUp.

## 14.5 Reading Data from a Servo Drive / Writing Data into a Servo Drive

Users can change or read the values of parameters in a servo drive on a DMCNET by means of the AH500 series motion control module which is connected to the servo drive. They can only set one servo drive at a time. After one servo drive is set, they can set another servo drive in the same way. Before the users set a servo drive, they have to check whether the servo drive is connected by means of bit 0~bit 3 in SR1073 (SR1173, SR1273…). After the users make sure that a servo drive is connected, they can use other register to write data into the servo drive, and read data from the servo drive.

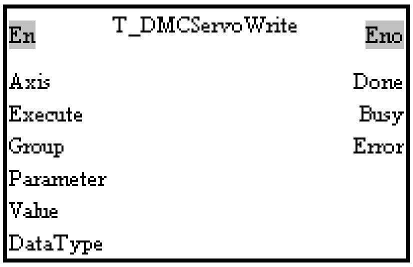
- Steps of writing a value into a servo drive
  - ◆ Using special data registers
    1. Users have to make sure of the node ID of the servo drive used. If the node ID of the servo drive used is 1, the first axis will be used. If the node ID of the servo drive used is 2, the second axis will be used. If the node ID of the servo drive used is 3/4/5/6/7/8/9/10/11/12, the 3$^{rd}$/4$^{th}$/5$^{th}$/6$^{th}$/7$^{th}$/8$^{th}$/9$^{th}$/10$^{th}$/11$^{th}$/12$^{th}$ axis will be used. Only one axis can be selected at a time.
    2. The users have to check whether the status of the servo drive used is that the servo drive is OFF/ON by means of bit 0~bit 3 in SR1073 (SR1173, SR1273…).
    3. After the users refer to the description of a parameter, they can know the group number assigned to the parameter, the parameter number assigned to the parameter, whether the parameter is a 16-bit/32-bit parameter, and whether the parameter can be set when

the servo drive used is ON. For example, the group number assigned to the parameter P1-44 is 1, the parameter number assigned to it is 44, and the parameter is a 32-bit parameter.

4. If a parameter in the servo drive can be set only when the servo drive is OFF, the users have to set the servo drive to OFF. If there is no such limitation, the step can be skipped.
5. The users have to set a parameter position in the servo drive by means of SR1078 (SR1178, SR1278…). If they want to use P1-44, the value of the high byte in SR1078 (SR1178, SR1278…) will be 16#01, and value of the low byte in SR1078 (SR1178, SR1278…) will be 16#2C, that is, the value in SR1078 (SR1178, SR1278…) will be 16#012C.
6. The users have to set the value which will be written into a servo drive by means of (SR1077, SR1076) ((SR1177, SR1176), (SR1277, SR1276)…).
7. The users have to set bit 11~bit 8 in SR1072 (SR1172, SR1272…). If the value of bit 11~bit 8 in SR1072 (SR1172, SR1272…) is 1, the value written into the servo drive will be a 16-bit value. If the value of bit 11~bit 8 in SR1072 (SR1172, SR1272…) is 2, the value written into the servo drive will be a 32-bit value. For example, the parameter P1-44 is a 32-bit parameter, and therefore the bit 11~bit 8 in SR1072 (SR1172, SR1272…) must be 2.
8. The users can check whether the writing of a value is correct by means of bit 11~bit 8 in SR1073 (SR1173, SR1273…). If an error occurs, an error code will be stored in (SR1077, SR1076) ((SR1177, SR1176), (SR1277, SR1276)…). If the writing of a value is successful, the value of bit 11~bit8 in SR1073 (SR1173, SR1273…) will be 0.
9. The users can write a value into another servo drive or read a value from another servo drive only after the setting of the servo used is complete, or an error occurs in the servo used.

◆ Using a motion control function block

Users can write a value into a servo drive by means of the motion control function block T_DMCServoWrite.




Please refer to Chapter 5 for more information about the input pins and the output pins in the motion control function block T_DMCServoWrite.

● Steps of reading a value from a servo drive

◆ Using special data registers

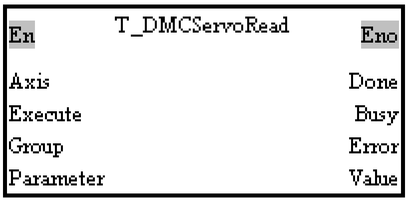
1. Users have to make sure of the node ID of the servo drive used. If the node ID of the servo drive used is 1, the first axis will be used. If the node ID of the servo drive used is 2, the second axis will be used. If the node ID of the servo drive used is 3/4/5/6/7/8/9/10/11/12, the $3^{rd}$/$4^{th}$/$5^{th}$/$6^{th}$/$7^{th}$/$8^{th}$/$9^{th}$/$10^{th}$/$11^{th}$/$12^{th}$ axis will be used. Only one axis can be selected at a time.
2. After the users refer to the description of a parameter, they can know the group number assigned to the parameter, and the parameter number assigned to the parameter. For example, the group number assigned to the parameter P1-44 is 1, and the parameter number assigned to it is 44.
3. The users have to set a parameter position in the servo drive by means of SR1078 (SR1178, SR1278…). If they want to use P1-44, the value of the high byte in SR1078 (SR1178, SR1278…) will be 16#01, and value of the low byte in SR1078 (SR1178, SR1278…) will be 16#2C, that is, the value in SR1078 (SR1178, SR1278…) will be

16#012C.

4. The users have to write 3 into bit 11~bit 8 in SR1072 (SR1172, SR1272…).
5. The users can check whether the reading of a value is correct by means of bit 11~bit 8 in SR1073 (SR1173, SR1273…). If an error occurs, an error code will be stored in (SR1077, SR1076) ((SR1177, SR1176), (SR1277, SR1276)…). If the reading of a value is successful, the value of bit 11~bit8 in SR1073 (SR1173, SR1273…) will be 0.
6. After a value is read successfully, the users can know the value by means of (SR1077, SR1076) ((SR1177, SR1176), (SR1277, SR1276)…).
7. The users can write a value into another servo drive or read a value from another servo drive only after the setting of the servo used is complete, or an error occurs in the servo used.

◆ Using a motion control function block

Users can read a value from a servo drive by means of the motion control function block T_DMCServoRead.




Please refer to Chapter 5 for more information about the input pins and the output pins in the motion control function block T_DMCServoRead.

## 14.6 DMCNET Motion Control

■ Retuning home

◆ Using special data registers

Before users instruct the servo drive used to return home by means of a DMCNET, they have to make sure of the following points.

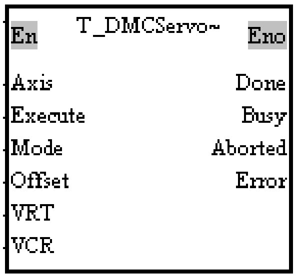
1. The value of bit 0~bit 3 in SR1073 (SR1173, SR1273…) is 2, that is, the servo drive used is ON.
2. When the servo drive used touches the left/right limit switch set and DOG's signal, it determines its operation. The users have to connect a left/right limit switch and DOG to the connector on the servo drive used. The users have to refer to section 3.4.2 in ASDA-A2 Series User Manual for more information.

After the users make sure of the points above, they can set a mode of returning home.

1. The users have to write 4 into bit 11~bit 8 in SR1072 (SR1172, SR1272…).
2. The users have to set a way in which the servo drive used returns home by means of SR1078 (SR1178, SR1278…). After SR1078 (SR1178, SR1278…) is set, the servo drive used will begin to return home.
3. When the servo drive used returns home, the value of bit11~bit 8 in SR1073 (SR1173, SR1273…) is 1. After the servo drive used returns home, the value of bit 11~bit 8 in SR1073 (SR1173, SR1273…) will become 0.
4. After the servo drive used returns home, its present position will be the same as the present position of its corresponding axis.

- Using a motion control function block

  Users can instruct a servo drive to return home by means of the motion control function block T_DMCServoHoming.

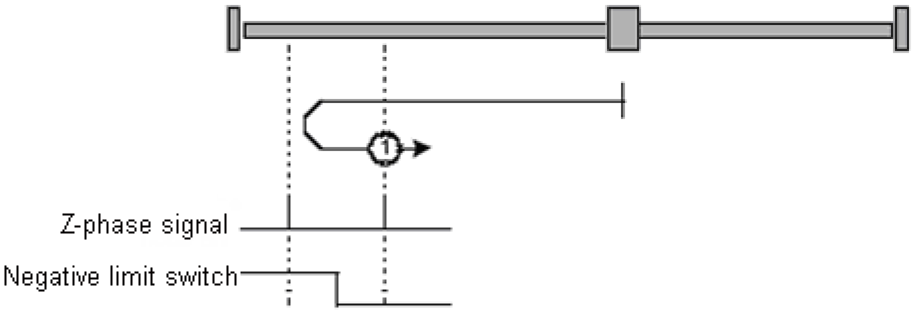



  Please refer to Chapter 5 for more information about the input pins and the output pins in the motion control function block T_DMCServoHoming.

There are several modes of returning home. These modes are described below.

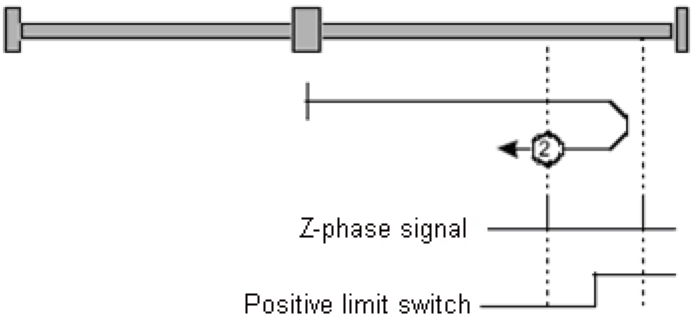
- The value in SR1078 (SR1178, SR1278…) is 1.

  The motor used rotates clockwise. After it comes into contact with the negative limit switch specified, it will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.




- The value in SR1078 (SR1178, SR1278…) is 2.

  The motor used rotates counterclockwise. After it comes into contact with the positive limit switch specified, it will rotate clockwise until a transition in a Z-phase signal from low to high occurs.




- The value in SR1078 (SR1178, SR1278…) is 3 or 4.
  1. The value in SR1078 (SR1178, SR1278…) is 3.

     The motor used rotates counterclockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.
  2. The value in SR1078 (SR1178, SR1278…) is 4.

     The motor used rotates counterclockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

If the motor used comes into contact with a limit switch when it returns home, it will stop after an error code is generated.

- The value in SR1078 (SR1178, SR1278…) is 5 or 6.

1. The value in SR1078 (SR1178, SR1278…) is 5.

   The motor used rotates clockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.

2. The value in SR1078 (SR1178, SR1278…) is 6.

   The motor used rotates clockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

If the motor used comes into contact with a limit switch when it returns home, it will stop after an error code is generated.

- The value in SR1078 (SR1178, SR1278…) is 7, 8, 9, or 10.

If the value in SR1078 (SR1178, SR1278…) is 7, 8, 9, or 10, the motor used will rotate counterclockwise, and search for a transition in DOG's signal. There are three situations.

a. The motor used does not find a transition in DOG's signal or does not come into contact with the positive limit switch specified.

   Mode 7: The motor used rotates counterclockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.

   Mode 8: The motor used rotates counterclockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

   Mode 9: The motor used rotates counterclockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.

   Mode 10: The motor used rotates counterclockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

b. The motor used rotates when DOG's signal is ON.

   Mode 7: The motor used rotates clockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.

   Mode 8: The motor used rotates clockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

   Mode 9: The motor used rotates counterclockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high.

   Mode 10: The motor used rotates counterclockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

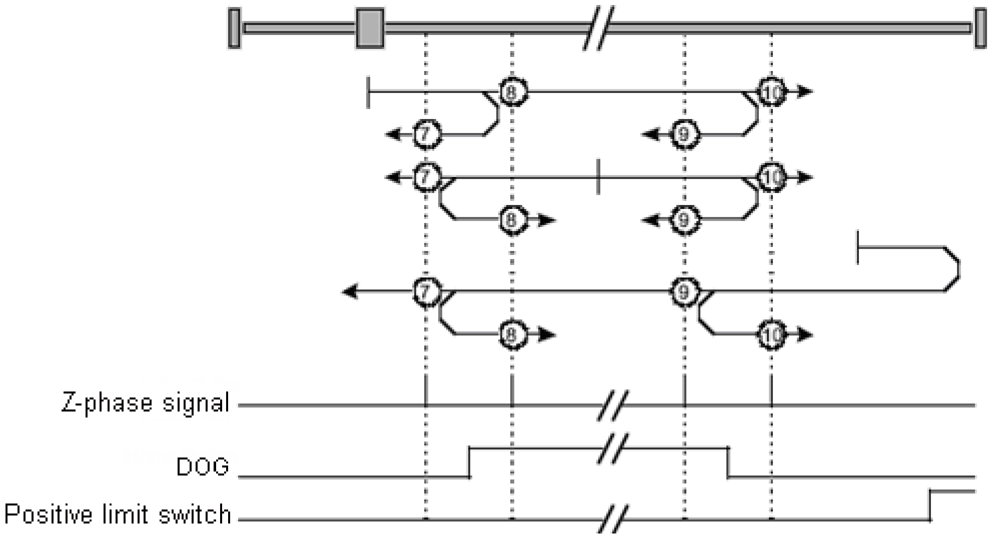
c. The motor used comes into contact with the positive limit switch specified.

   Mode 7: The motor used rotates counterclockwise. After the motor comes into contact with the positive limit switch specified, it will rotate clockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.

   Mode 8: The motor used rotates counterclockwise. After the motor comes into contact with the positive limit switch specified, it will rotate clockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

   Mode 9: The motor used rotates counterclockwise. After the motor comes into contact with the positive limit switch specified, it will rotate clockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.

Mode 10: The motor used rotates counterclockwise. After the motor comes into contact with the positive limit switch specified, it will rotate clockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.






- The value in SR1078 (SR1178, SR1278…) is 11, 12, 13, or 14.

  If the value in SR1078 (SR1178, SR1278…) is 11, 12, 13, or 14, the motor used will rotate clockwise, and search for a transition in DOG's signal. There are three situations.

  a. The motor used does not find a transition in DOG's signal or does not come into contact with the negative limit switch specified.

     Mode 11: The motor used rotates clockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

     Mode 12: The motor used rotates clockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.

     Mode 13: The motor used rotates clockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

     Mode 14: The motor used rotates clockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.

  b. The motor used rotates when DOG's signal is ON.

     Mode 11: The motor used rotates counterclockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

     Mode 12: The motor used rotates counterclockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.

     Mode 13: The motor used rotates clockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high.

     Mode 14: The motor used rotates clockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

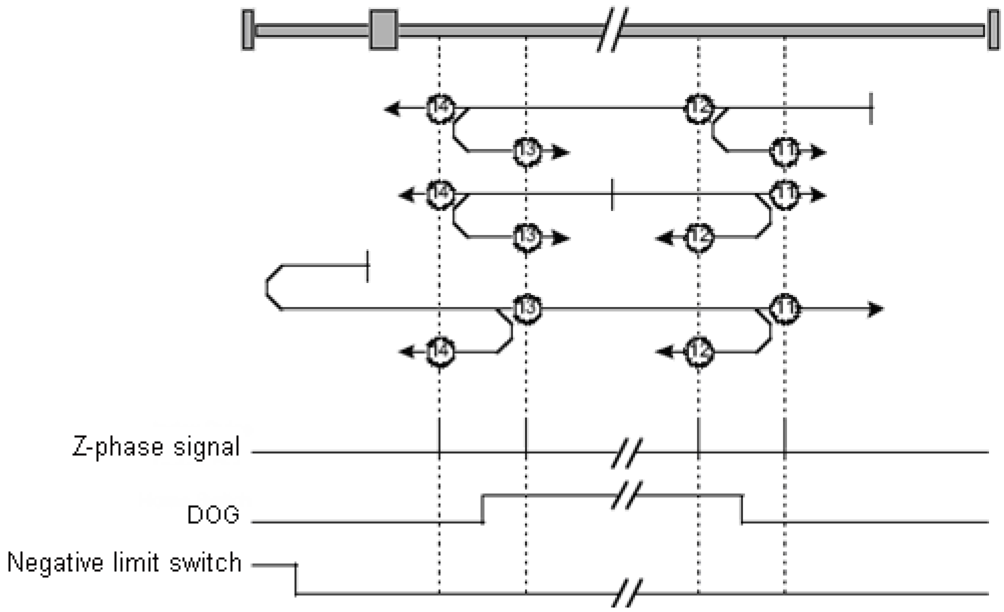
c. The motor used comes into contact with the negative limit switch specified.

Mode 11: The motor used rotates clockwise. After the motor comes into contact with the negative limit switch specified, it will rotate counterclockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

Mode 12: The motor used rotates clockwise. After the motor comes into contact with the negative limit switch specified, it will rotate counterclockwise. After a transition in DOG's signal from high to low occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.

Mode 13: The motor used rotates clockwise. After the motor comes into contact with the negative limit switch specified, it will rotate counterclockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate counterclockwise until a transition in a Z-phase signal from low to high occurs.

Mode 14: The motor used rotates clockwise. After the motor comes into contact with the negative limit switch specified, it will rotate counterclockwise. After a transition in DOG's signal from low to high occurs, the motor will rotate clockwise until a transition in a Z-phase signal from low to high occurs.




- The value in SR1078 (SR1178, SR1278…) is in the range of 17 to 30.
  The way in which the motor used operates when the value in SR1078 (SR1178, SR1278…) is in the range of 17 to 30 is similar to the way in which the motor used operates when the value in SR1078 (SR1178, SR1278…) is in the range of 1 to 14. If the value in SR1078 (SR1178, SR1278…) is in the range of 1 to 14, motor specified stops when a transition in a Z-phase signal from low to high occurs. If the value in SR1078 (SR1178, SR1278…) is in the range of 17 to 30, the motor specified stops when a transition in DOG's signal from low to high occurs.
- The value in SR1078 (SR1178, SR1278…) is 33.
  The motor used rotates clockwise until a transition in a Z-phase signal from low to high occurs.
- The value in SR1078 (SR1178, SR1278…) is 34.
  The motor used rotates counterclockwise until a transition in a Z-phase signal from low to high occurs.
- The value in SR1078 (SR1178, SR1278…) is 35.
  Users can change the value which indicates the present position of the motor used to another value.

■ DMCNET motion

In a DMCNET motion mode, AH20MC-5A can update the commands sent to the twelve axes simultaneously. It only takes 1 millisecond to update the commands sent to the twelve axes in AH20MC-5A. After a special data register is set to DMCNET motion mode, a servo drive will be set to DMCNET motion mode. Some of the twelve axes in AH20MC-5A can be instructed to return home, and the others can operate in DMCNET motion modes. If servo drives operate in DMCENT motion modes, they will support uniaxial motion and multiaxial motion.

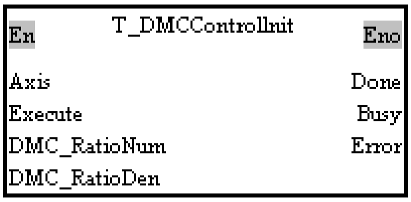
The steps of setting a DMCNET motion mode are as follows.

1. Users have to make sure that the value of bit 0~bit 3 in SR1073 (SR1173, SR1273…) is 2.
2. The users have to write 5 into bit 11~bit 8 in SR1072 (SR1172, SR1272…).
3. The users have to make sure that the value of bit 11~bit 8 in SR1073 (SR1173, SR1273…) is 1, that is, they have to make sure that the servo drive specified operates in a DMCNET motion mode.
4. The users can control the axis specified by means of uniaxial motion. They can set special data registers for uniaxial motion.
5. If the users want to control axes by means of multiaxial interpolation, they can write G-codes.
6. If users want to instruct the axis specified to return home, they have to write 4 into bit 11~bit 8 in SR1072 (SR1172, SR1272…), and select a mode of returning home described above.

Users can instruct some axes in AH20MC-5A to return home, and the others to operate in DMCNET motion modes simultaneously. The axes which are instructed to return home can not be controlled by uniaxial motion and multiaxial motion. The motion of a servo drive can be controlled only when the servo drive operates in a DMCNET motion mode.

Note: Before a servo drive is set to DMCNET motion mode, the value indicating the present position of the servo drive must be the same as the value indicating the present position of the axis specified. Before users set a DMCNET motion mode by means of a DMCNET, they have to change the value indicating the present position of the motor specified to the value stored in (SRmn33, SRmn32) by writing 35 into SR1078 (SR1178, SR1278…). (mn=10~25)

Users can use the motion control function block T_DMCControlInit to initialize the servo drive specified on a DMCNET.




If users use the motion control function block T_DMCControlInit to initialize the axis specified on a DMCNET, the servo drive specified will be started, the AH500 series motion control module used will be synchronized with the servo drive specified, and the servo drive specified will operate in a synchronous control mode. Please refer to Chapter 5 for more information about the motion control function block T_DMCControlInit.

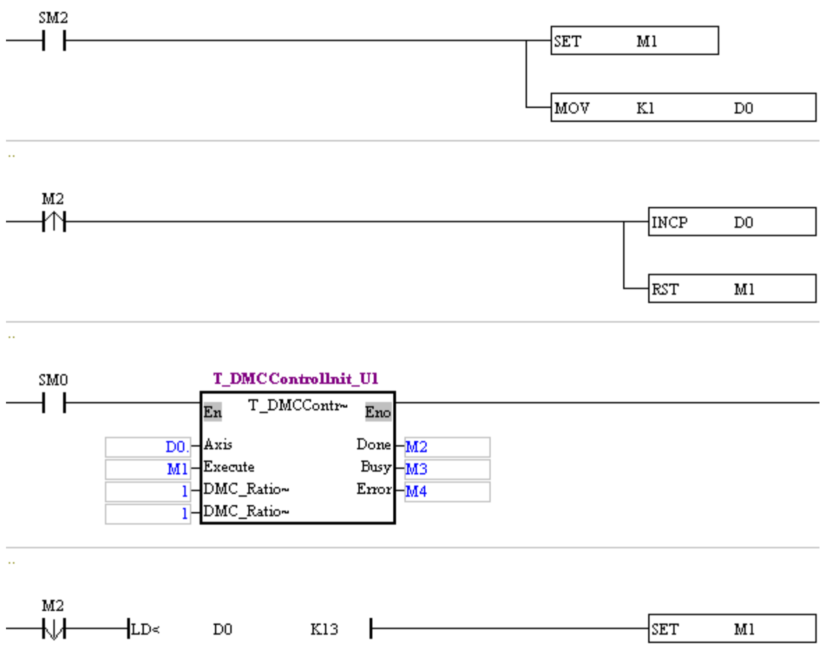
# 14.7 Examples

## 14.7.1 Connecting an Incremental Servo Drive

- Setting a servo drive
  - Set P2-08 to 16#10. (Restore the servo drive to its factory setting.)
  - Turn off the servo drive.
  - Turn on the servo drive.
  - Set P1-01 to 16#0B. (Set the servo drive to DMCNET mode.)
  - Set P3-01 to 16#203. (Set P3-01 to Delta DMCNET mode.)
  - Set P2-15 to 16#0000. (Remove the negative limit switch which is connected.) (Users can set P2-15 by themselves.)
  - Set P2-16 to 16#0000. (Remove the positive limit switch which is connected.) (Users can set P2-16 by themselves.)
  - Set P2-17 to 16#0000. (Remove the function of stopping the servo drive in an emergency.) (Users can set P2-17 by themselves.)
  - Set P3-00. (Set the station address of the servo drive.) The value of P3-00 must be in the range of 16#01 to 16#0C.

    * There must be a servo drive whose station address is 16#01 on a DMCNET.
  - Turn off the servo drive, and then turn on the servo drive.
  - Set P0-02 to 16#120. If the servo drive is connected successfully, the value shown on the display of the servo drive will be 16#80. If the value shown on the display is 16#06, users have to check whether there is a servo drive whose station address is 16#01.
- Writing a program

  Write T_DMCControlInit in a program for AH20MC-5A.




In this example, after the PLC used begins to operate, the servo drive whose station address is 1 will be initialized first. After the initialization of the servo drive whose station address is 1 is complete, the value shown on the display of the servo drive will be 16#111, the servo drive will be operable, and the servo drive will be ON.

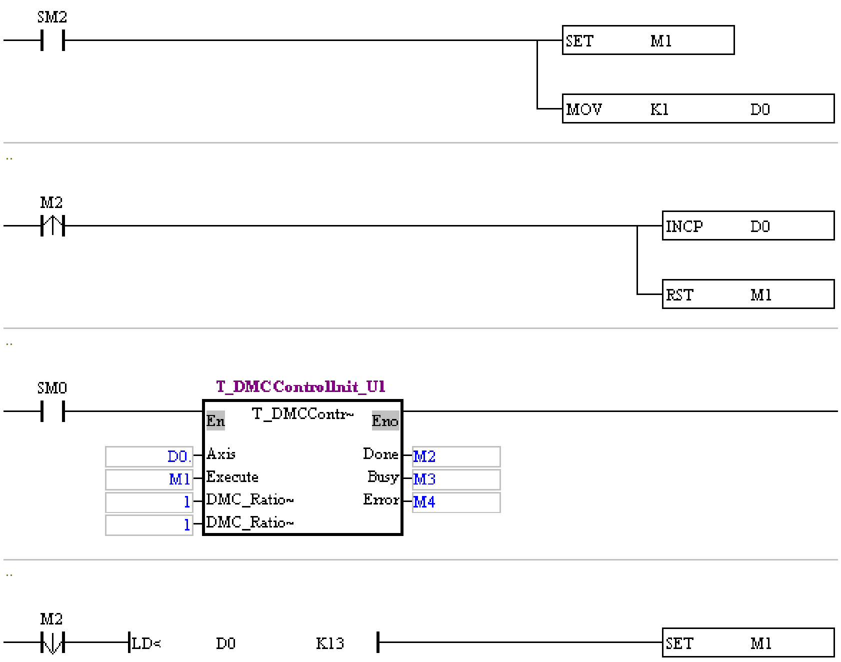
* When the motion control function block T_DMCControlInit is used, only one servo drive can be initialized at a time.

## 14.7.2 Connecting an Absolute Servo Drive

- Setting a servo drive
  - Set P2-08 to 16#10. (Restore the servo drive to its factory setting.)
  - Turn off the servo drive.
  - Turn on the servo drive.
  - Set P1-01 to 16#0B. (Set the servo drive to DMCNET mode.)
  - Set P3-01 to 16#203. (Set P3-01 to Delta DMCNET mode.)
  - Set P2-69 to 16#0001. (The encoder connected to the servo drive is an absolute encoder.)
  - Set P2-15 to 16#0000. (Remove the negative limit switch which is connected.) (Users can set P2-15 by themselves.)
  - Set P2-16 to 16#0000. (Remove the positive limit switch which is connected.) (Users can set P2-16 by themselves.)
  - Set P2-17 to 16#0000. (Remove the function of stopping the servo drive in an emergency.) (Users can set P2-17 by themselves.)
  - Set P3-00. (Set the station address of the servo drive.) The value of P3-00 must be in the range of 16#01 to 16#0C.

    * There must be a servo drive whose station address is 16#01 on a DMCNET.
  - Set P3-12 to 16#100. (Enable the function of memorizing the values of the parameters in the servo drive.)
  - Turn off the servo drive, and then turn on the servo drive.
  - Set P0-02 to 16#120. If the servo drive is connected successfully, the value shown on the display of the servo drive will be 16#80. If the value shown on the display is 16#06, users have to check whether there is a servo drive whose station address is 16#01.
- Writing a program

  Write T_DMCControlInit in a program for AH20MC-5A.




In this example, after the PLC used begins to operate, the servo drive whose station address is 1 will be initialized first. After the initialization of the servo drive whose station address is 1 is complete, the value shown on the display of the servo drive will be 16#111, the servo drive will be operable, and the servo drive will be ON.

* When the motion control function block T_DMCControlInit is used, only one servo drive can be initialized at a time.

## 14.8 Troubleshooting

| Problem | Remedy |
|---|---|
| The DMCNET connection LED indicator on an AH500 series motion control module is not ON. | Check whether a networking cable is connected to the AH500 series motion control module correctly, and check whether a terminal resistor is connected correctly. |
| The value of bit 0~bit 3 in SR1073 (SR1173, SR1273…) is 0. | Users have to instruct the servo drive used to reset NMT by means of bit 0~bit 3 in SR1072 (SR1172, SR1272…). If the servo drive is still not connected, the users have to check whether the cables connected to the hardware used are loose, and check whether a terminal resistor is connected. |
| The value of a parameter in a servo drive is incorrect. | Check whether the parameter can be set when the servo is ON/OFF, and check whether the data type set is correct. |
| After a value is written into a servo drive, no response is received in a specified amount of time. | Check whether the servo drive can be connected correctly. |
| After P0-02 in a servo drive is set to 16#120, the value shown on the display of the servo drive is 16#06. | 1. Check whether there is a servo drive whose node ID is 1 on the DMCNET created.<br>2. Check whether the cables connected to the hardware used are loose. |

**MEMO**

# Chapter 15 Setting USB Communication in PMSoft

## Table of Contents

## 15.1 Functions

An AH500 series motion control module is equipped with a mini USB port. The mini USB port on an AH500 series motion control module can be connected to PMSoft. The functions that PMSoft can perform are listed below.

- A program can be uploaded/downloaded and monitored.
- The values in registers can be monitored and changed.

## 15.2 Specifications

- Communication

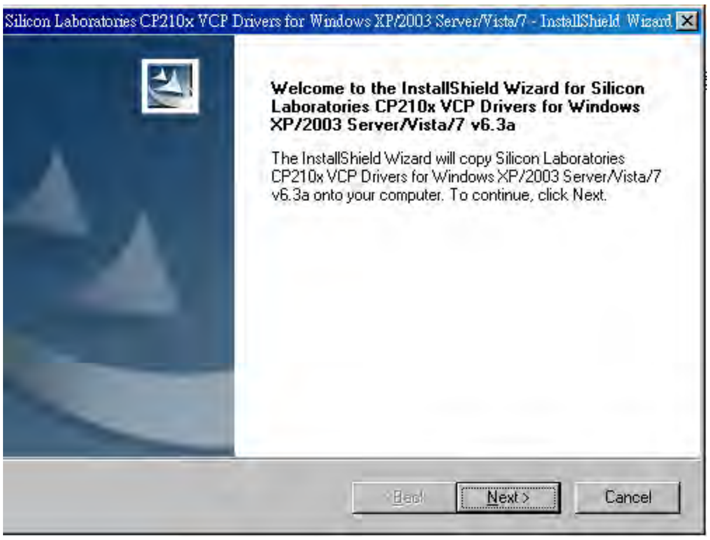
| Item | Specifications |
|---|---|
| Data type | Modbus ASCII |
| Serial transmission speed | 9600, 19200, 38400, 57600, 115200 bit/s |
| Maximum transmission distance | 5 meters |

## 15.3 Communicating with PMSoft

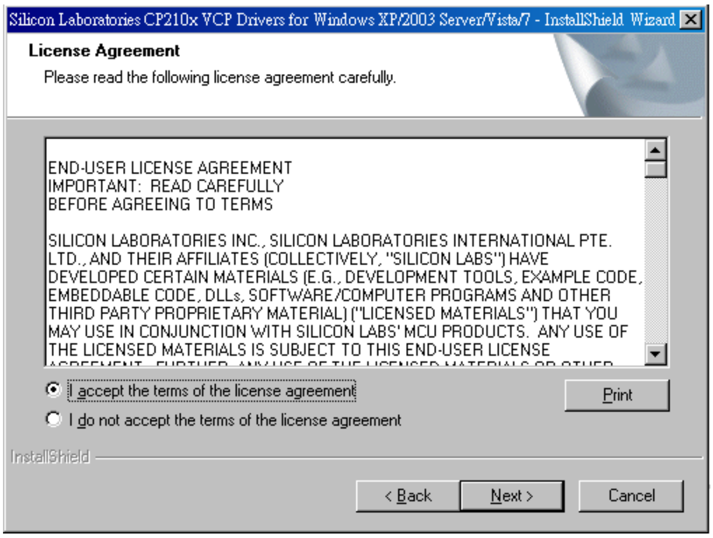
- Setting a connection environment

  Before users use the mini USB port on an AH500 series motion control module, they have to install a USB driver on a computer. The default path which denotes the folder in which a USB driver is saved is C:\Program Files\Delta Industrial Automation\PMSoft x.xx\drivers\CP210x_VCP_Win_XP_S2K3_Vista_7. x.xx is the version of PMSoft. If Silicon Laboratories CP210x VCP Drivers have been installed, they do not need to be installed again. The steps of installing Silicon Laboratories CP210x VCP Drivers are as follows.

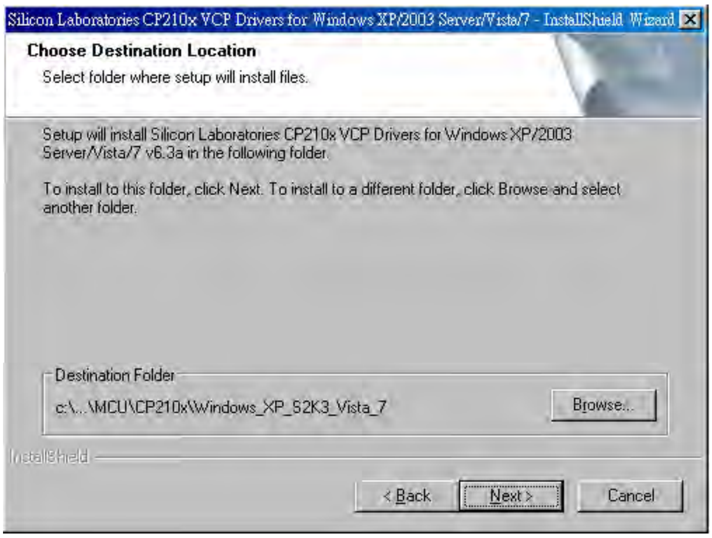
  1. Double-click the USB driver in the **PMSoft x.xx** folder.
  2. Click **Next** in the **Silicon Laboratories CP210x VCP Drivers for Windows XP/2003 Server/Vista/7-InstallShield Wizard** window.



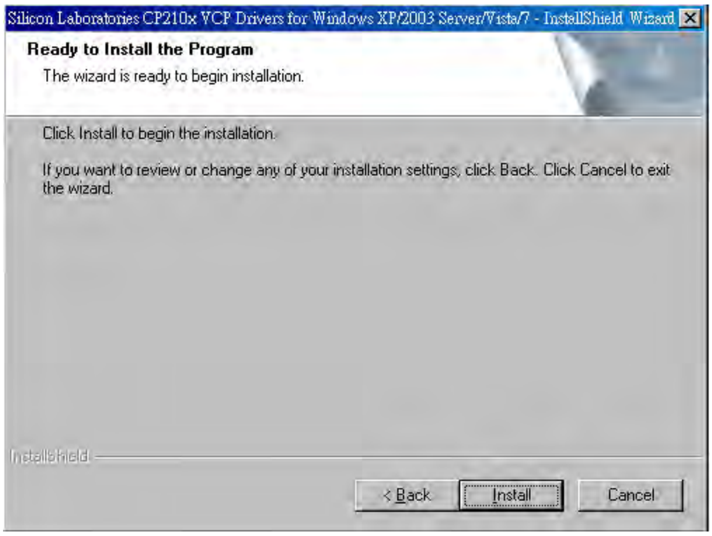
3. Select the **I accept the terms of the license agreement** option button, and then click **Next**.



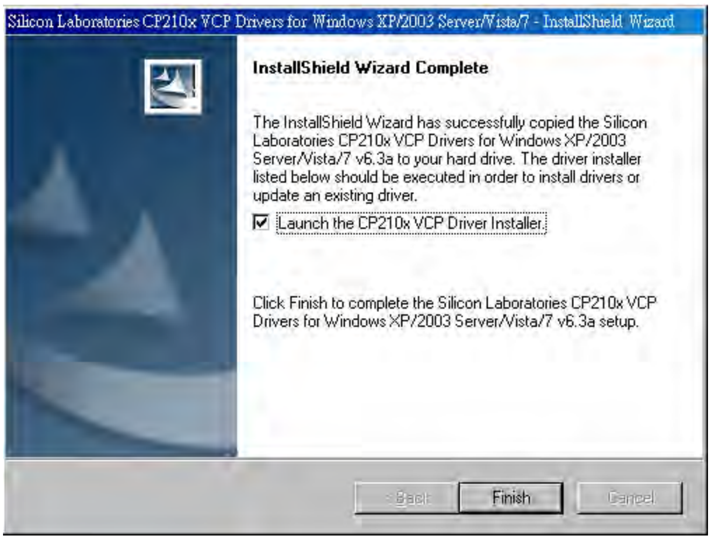

4. Users can select an installation path in the window which appears. If they do not want to change the installation path in the window, they can click **Next**.



5. After the users click **Install**, the installation of the USB driver will begin.

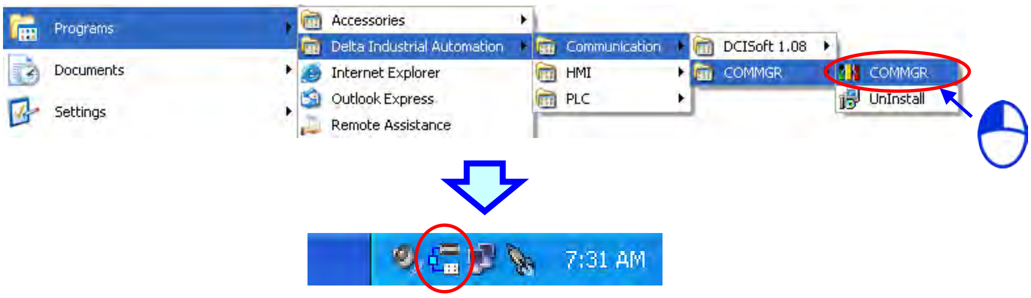



6. After the installation of the USB driver is complete, the users have to click **Finish**.

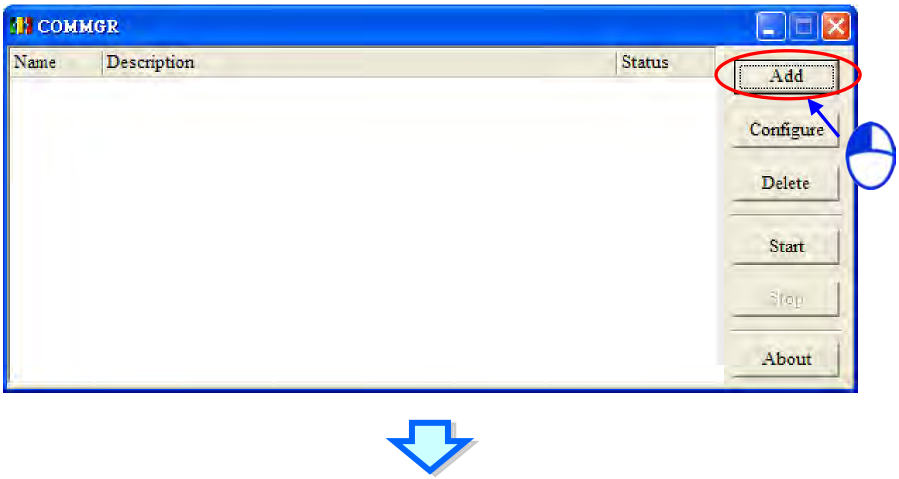


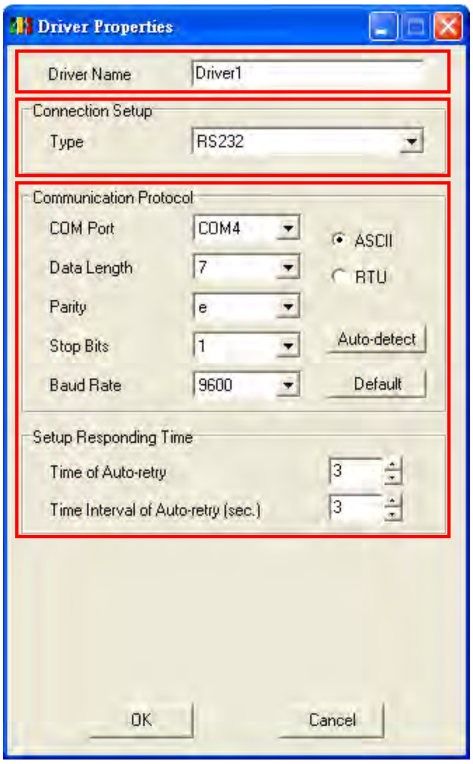

15

- Setting a connection by means of PMSoft
  1. Start COMMGR. If the icon representing COMMGR is not displayed on the system tray, the users can start COMMGR by clicking the shortcut on the **Start** menu (**Start**>**Programs**>**Delta Industrial Automation**>**Communication**>**COMMGR)**.

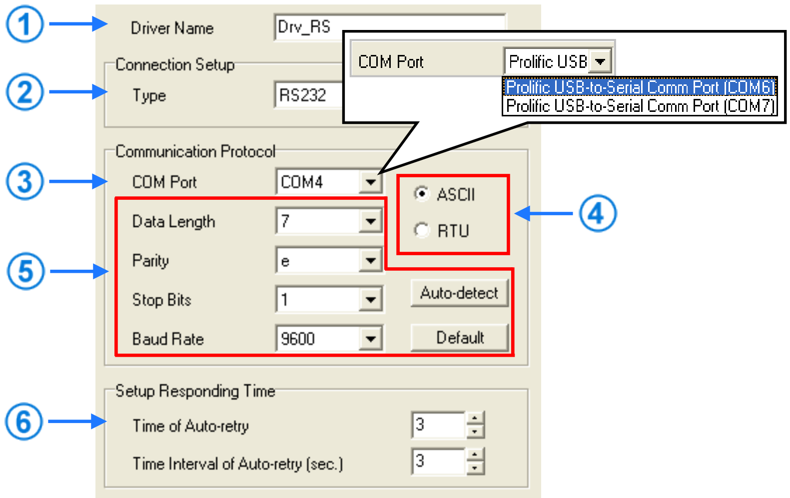



  2. Creating a driver in COMMGR
     After users click **Add** in the **COMMGR** window, the **Driver Properties** window will appear.



◆ Setting the parameters of an RS-232 driver




① Users can type a driver name in the **Driver Name** box.

② Select **RS232** in the **Type** drop-down list box in the **Connection Setup** section.

③ Select an RS-232 communication port in the **COM Port** drop-down list box. Each item in the **COM Port** drop-down list box is composed of a device name and a communication port number. The communication ports in the **COM Port** drop-down list box are the same as the communication ports in the **Device Manager** window.

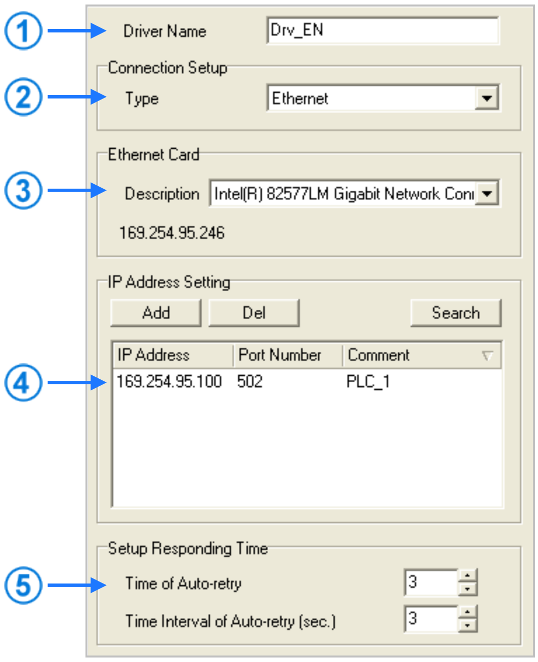
④ The communication format used can be **ASCII** or **RTU**.

⑤ The communication protocol for exchanging data through a communication port selected must be the same as the communication protocol for exchanging data through a communication port on a device connected. If users click **Default**, the values of all communication parameters will return to their default values.

If users do not know the communication protocol for exchanging data through a communication port on a device connected, the users can connect the device to an RS-232 communication port selected with an RS-232 cable, and click **Auto-detect** to automatically detect the communication protocol. If the communication protocol is detected successfully, the related communication parameters in the **Driver Properties** window are set. However, when the communication protocol is detected automatically, the **COM Port** parameter and the **ASCII/RTU** parameter are not detected. As a result, the users have to set the **COM Port** parameter and the **ASCII/RTU** parameter before clicking **Auto-detect**.

⑥ Users can select the number of times the sending of a command is retried if a connection error occurs in the **Time of Auto-retry** box, and select an interval of retrying the sending of a command in the **Time Interval of Auto-retry** box.

◆ Setting the parameters of an Ethernet driver




① Users can type a driver name in the **Driver Name** box.

② Select **Ethernet** in the **Type** drop-down list box in the **Connection Setup** section.

③ Select a network interface card in the **Description** drop-down list box. An IP address assigned to a network interface card selected is displayed in the lower left corner of the **Ethernet Card** section.

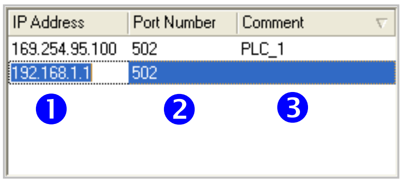
④ Owing to the characteristics of Ethernet, a computer can communicate with all devices on a network. Users can create the IP addresses of devices connected to this driver in the **IP Address Setting** section.

➢ After users click **Add** to add a new IP address to the list of IP addresses in the **IP Address Setting** section, they can type related information in the **IP Address** section, the **Port Number** column, and the **Comment** column.

❶ Users can type the IP address of a device connected in this cell.

❷ Users can type the communication port number specified.

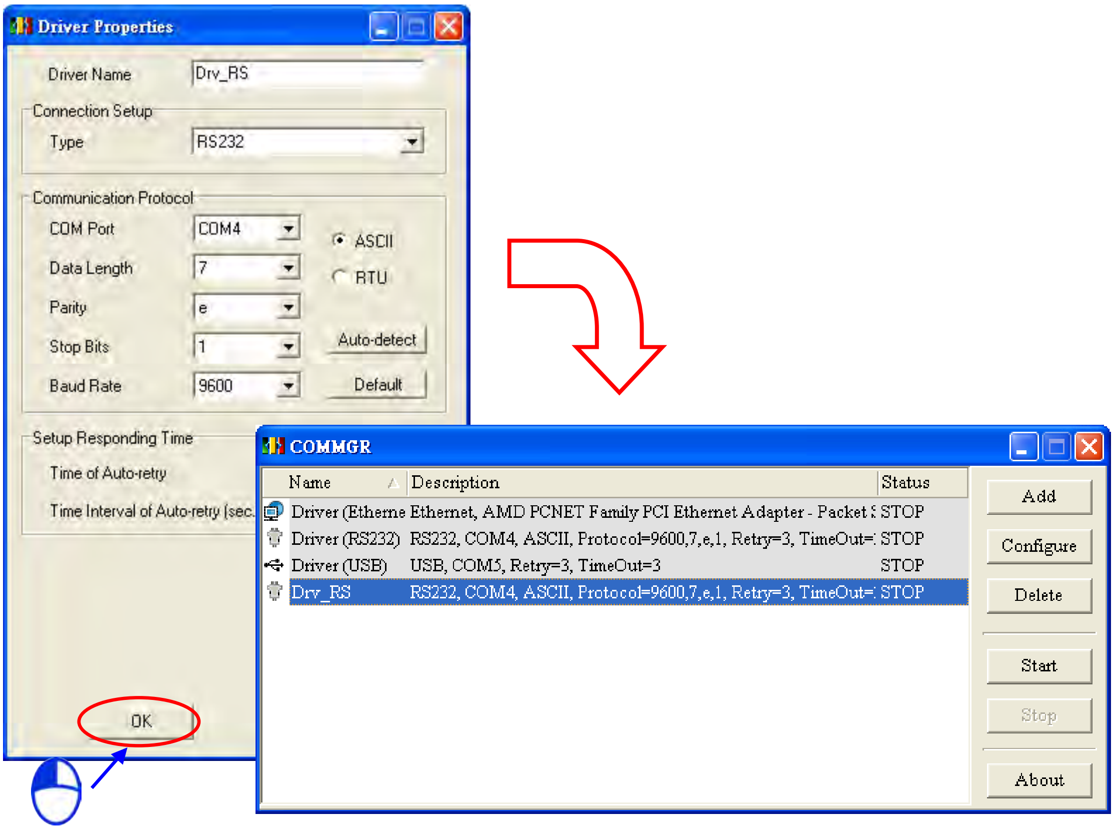
❸ Users can type a comment in this cell.




➢ After users select an IP address, they can click **Del** or press DEL on the keyboard to delete the IP address from the list.

⑤ Users can select the number of times the sending of a command is retried if a connection error occurs in the **Time of Auto-retry** box, and select an interval of retrying the sending of a command in the **Time Interval of Auto-retry** box.

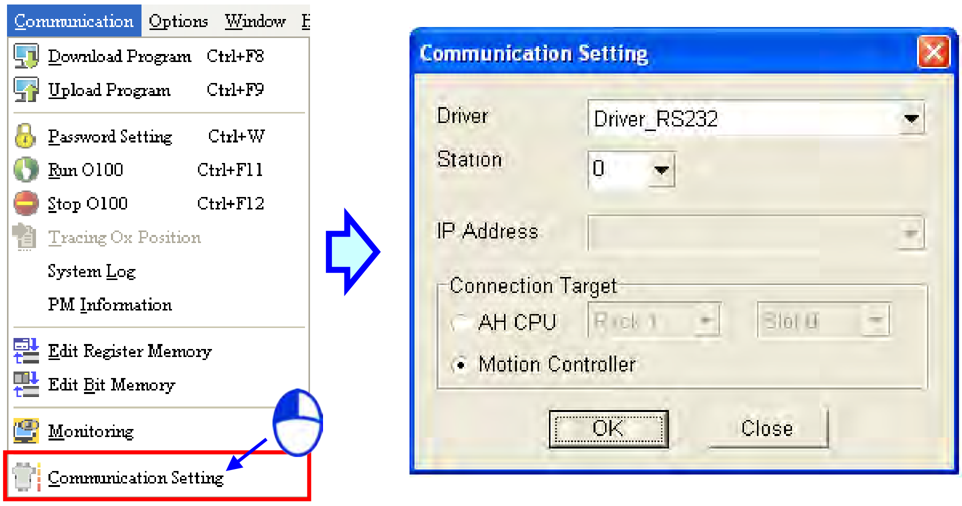
After the users set the parameters of a driver, and click **OK** in the **Driver Properties** window, the parameters related to the driver will be displayed in the **COMMGR** window. The creation of a driver is equivalent to the creation of a connection. The users can start or stop the driver according to their needs.



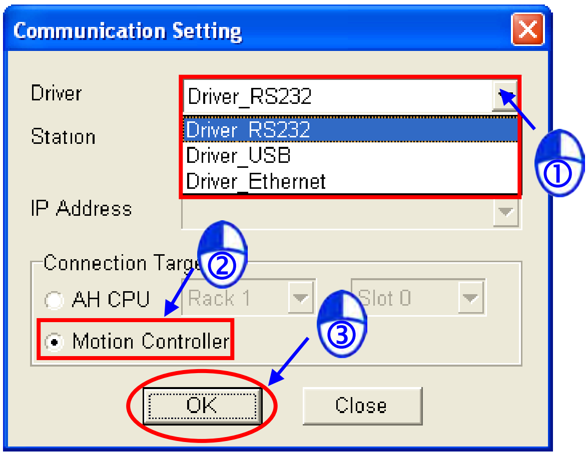

3. Using PMSoft

◆ Connecting to PMSoft directly

(1) Start PMSoft, and click **Communication Setting** on the **Communication** menu.

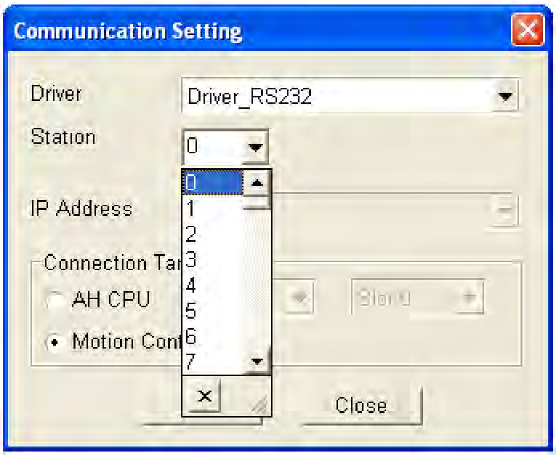


(2) Select a driver in the **Driver** drop-down list box. Before users create a connection between PMSoft and an AH500 series motion control module, they have to make sure that the driver is started in COMMGR. Select the **Motion Controller** option button, and click **OK**. The communication setting varies with the driver selected.




➢ **RS232 and USB**

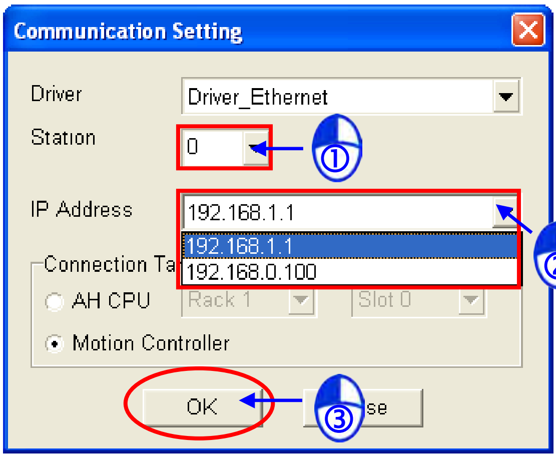
Users have to select the station address of the AH500 series motion control module connected to the computer in the **Station** drop-down list box. If the station address selected is 0, a broadcast communication will be carried out. If the AH500 series motion control module used can not communicate with PMSoft, or the users do not know the station address of the AH500 series motion control module connected to the computer, they can select 0 in the **Station** drop-down list box.



15

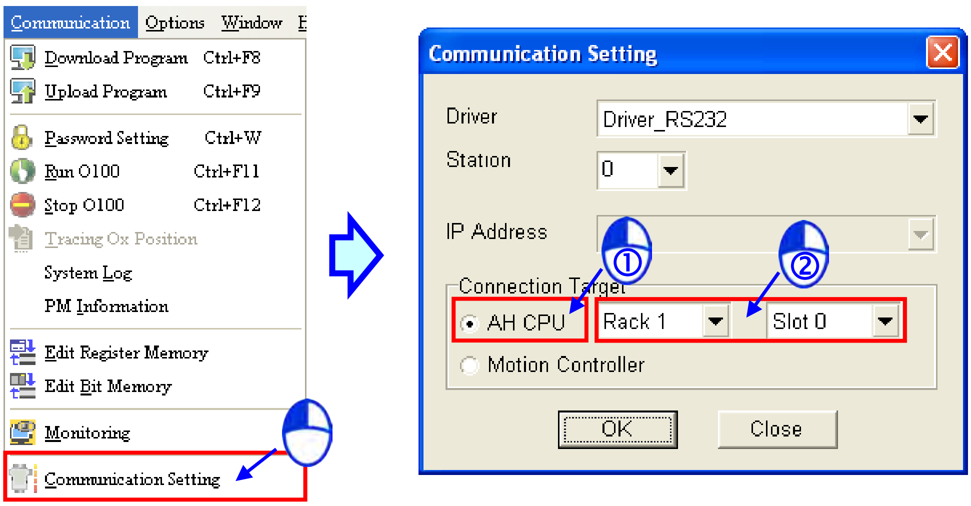
➢ **Ethernet**

Users have to select the station address of the AH500 series motion control module connected to the computer in the **Station** drop-down list box. If the station address selected is 0, a broadcast communication will be carried out. The users also have to select the IP address created in COMMGR in the **IP Address** drop-down list box.




◆ Using ISPSoft and PMSoft (indirect connection)

After users click **Communication Setting** on the **Communication** menu in PMSoft, the **Communication Setting** window will appear. The users have to select the **AH CPU** option button, select the correct rack number, and select the correct slot number.




Please refer to PMSoft User Manual for more information.

# Appendix A Error Code Tables

## Table of Contents

# A.1 Error Code Tables

After a program is written into an AH500 series motion control module, the ERROR LED indicator will blink and an error flag will be ON if an error occurs in O100 or an Ox motion subroutine. The reason for the error occurring in O100 or an Ox motion subroutine may be that the use of operands (devices) is incorrect, syntax is incorrect, or the setting of motion parameters is incorrect. Users can know the reasons for the errors occurring in an AH500 series motion control module by means of the error codes (hexadecimal codes) stored in error registers.

◆ Error message table

| **Error type** | **Program error** | | **Motion error** |
|---|---|---|---|
| | **Program block** | | **mn=10~25** |
| | **O100** | **Ox** | **(10: 1st axis; 25: 16th axis)** |
| **Error flag (Special auxiliary relay)** | SM953 | SM1049 | SMmn49 |
| **Error register (Special data register)** | SR802 | SR1041 | SRmn41 |
| **Step number** | SR803 | SR1053 | |

◆ Program error codes and motion error codes (hexadecimal codes)

| **Error code** | **Description** | **Error code** | **Description** |
|---|---|---|---|
| 0002 | The subroutine used has no data. | 0031 | The positive pulses generated by motion are inhibited. |
| 0003 | CJ, CJN, and JMP have no matching pointers. | 0032 | The negative pulses generated by motion are inhibited. |
| 0004 | There is a subroutine pointer in the main program. | 0033 | The motor used comes into contact with the left/right limit switch set. |
| 0005 | Lack of a subroutine | 0040 | A device exceeds the device range available. |
| 0006 | A pointer is used repeatedly in the same program. | 0044 | An error occurs when a device is modified by a 16-bit index register/32-bit index register. |
| 0007 | A subroutine pointer is used repeatedly. | 0045 | The conversion into a floating-point number is incorrect. |
| 0008 | The pointer used in JMP is used repeatedly in different subroutines. | 0047 | An error occurs when the Ox motion subroutine numbers in an SD card are read. |
| 0009 | The pointer used in JMP is the same as the pointer used in CALL. | 0E18 | The conversion into a binary-coded decimal number is incorrect. |
| 000A | A pointer is the same as a subroutine pointer. | 0E19 | Incorrect division operation (The divisor is 0.) |
| 0011 | Target position (I) is incorrect. | C401 | General program error |
| 0012 | Target position (II) is incorrect. | C402 | LD/LDI has been used more than nine times. |
| 0021 | Velocity (I) is incorrect. | C404 | There is more than one nested program structure supported by RPT/RPE. |
| 0022 | Velocity (II) is incorrect. | C405 | SRET is used between RPT and RPE. |
| 0023 | The velocity ($V_{RT}$) of returning home is incorrect. | C4EE | There is no M102 in the main program, or there is no M2 in a motion subroutine. |

| Error code | Description | Error code | Description |
|---|---|---|---|
| 0024 | The velocity ($V_{CR}$) to which the velocity of the axis specified decreases when the axis returns home is incorrect. | C4FF | A wrong instruction is used, or a device used exceeds the range available. |
| 0025 | The JOG speed set is incorrect. | 8000 | A DMCNET servo drive is disconnected. (AH20MC-5A supports DMCNET communication.) |

- If a servo drive is connected successfully, the error codes stored in the error registers in AH20MC-5A will include servo drive error codes. The servo drive error code stored in an error register in AH20MC -5A is composed of an AL code and 16#8000.

  Please refer to Chapter 10 in ASDA-A2 Series User Manual for more information about the servo drive error codes which are not described in the table below.

| Error code | Description | Error code | Description |
|---|---|---|---|
| 8001 | Overcurrent | 8014 | Reverse limit switch error |
| 8002 | Overvoltage | 8015 | Forward limit switch error |
| 8003 | Undervoltage | 8016 | An IGBT overheats. |
| 8004 | Motor error | 8018 | Encoder output error |
| 8005 | Regeneration error | 8030 | Motor protection error |
| 8006 | Overload | 8301 | DMCNET synchronization fails. |
| 8007 | Overspeed | 8302 | DMCNET synchronization signal error |
| 8009 | Excessive deviation | 8303 | DMCNET synchronization timeout |
| 8011 | Encoder error | | |
| 8012 | Adjustment error | | |
| 8013 | A servo drive makes an emergency stop. | | |

**MEMO**