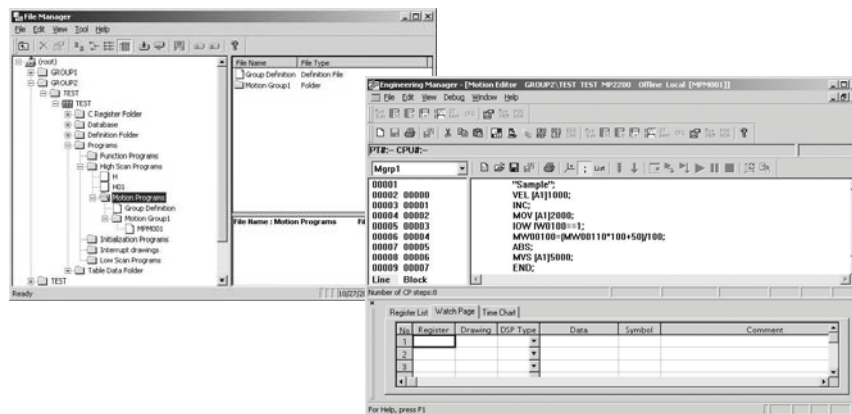




# Machine Controller MP900/MP2000 Series USER'S MANUAL MOTION PROGRAMMING





## Safety Information

The following conventions are used to indicate precautions in this manual. Failure to heed precautions provided in this manual can result in serious or possibly even fatal injury or damage to the products or to related equipment and systems.

 **WARNING** Indicates precautions that, if not heeded, could possibly result in loss of life or serious injury.

 **Caution** Indicates precautions that, if not heeded, could result in relatively serious or minor injury, damage to the product, or faulty operation.

The warning symbols for ISO and JIS standards are different, as shown below.

ISO	JIS
	

The ISO symbol is used in this manual.

Both of these symbols appear on warning labels on Yaskawa products. Please abide by these warning labels regardless of which symbol is used.

©Yaskawa, 1998

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of Yaskawa. No patent liability is assumed with respect to the use of the information contained herein. Moreover, because Yaskawa is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, Yaskawa assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

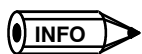
---

## Visual Aids

The following aids are used to indicate certain types of information for easier reference.



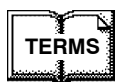
Indicates application examples.



Indicates supplemental information.



Indicates important information that should be memorized.



Describes technical terms that are difficult to understand, or appear in the text without an explanation being given.

---

---

# Contents

Safety Information .....	iii
Visual Aids .....	iv
Overview .....	ix
Using This Manual .....	xi
Safety Precautions .....	xii
Warranty .....	xiv

## 1 Overview of Motion Programs

<b>1.1 Motion Programming .....</b>	<b>1-2</b>
1.1.1 Motion Programming Capabilities .....	1-2
1.1.2 Function Performance .....	1-5
1.1.3 Starting Motion Programs .....	1-14
1.1.4 Parallel Program Execution .....	1-21
1.1.5 Editing Programs .....	1-22
<b>1.2 Programming Methods .....</b>	<b>1-23</b>
1.2.1 Programming Format .....	1-23
1.2.2 Controlled Axes .....	1-28
1.2.3 Feed Speeds .....	1-35
1.2.4 Motion Commands .....	1-39

## 2 Motion Commands

<b>2.1 Axis Move Commands .....</b>	<b>2-2</b>
2.1.1 POSITIONING (MOV) .....	2-2
2.1.2 LINEAR INTERPOLATION (MVS) .....	2-5
2.1.3 CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION (MCW, MCC) .....	2-9
2.1.4 CLOCKWISE/COUNTERCLOCKWISE HELICAL INTERPOLATION (MCW, MCC) .....	2-15
2.1.5 ZERO POINT RETURN (ZRN) .....	2-18
2.1.6 SKIP FUNCTION (SKP) .....	2-23
2.1.7 SET TIME POSITIONING (MVT) .....	2-24
2.1.8 EXTERNAL POSITIONING (EXM) .....	2-27
<b>2.2 Control Commands .....</b>	<b>2-28</b>
2.2.1 ABSOLUTE MODE (ABS) .....	2-28
2.2.2 INCREMENTAL MODE (INC) .....	2-30
2.2.3 CURRENT POSITION SET (POS) .....	2-31
2.2.4 COORDINATE PLANE SETTING (PLN) .....	2-33
2.2.5 MOVE ON MACHINE COORDINATES (MVM) .....	2-34
2.2.6 PROGRAM CURRENT POSITION UPDATE (PLD) .....	2-35
2.2.7 DWELL TIME (TIM) .....	2-36
2.2.8 PROGRAM END (END) .....	2-37

---

## 3 Advanced Programming

<b>3.1 High-Level Control Commands</b> .....	<b>3 -2</b>
3.1.1 SECOND IN-POSITION CHECK (PFN) .....	3 -2
3.1.2 SET SECOND IN-POSITION RANGE (INP) .....	3 -4
3.1.3 IGNORE SINGLE-BLOCK SIGNAL (SNG), SINGLE-BLOCK SIGNAL DISABLED/ ENABLED (SNGD/SNGE) .....	3 -6
3.1.4 USER FUNCTION CALL (UFC) .....	3 -8
3.1.5 I/O VARIABLE WAIT (IOW) .....	3 -16
3.1.6 SUBROUTINE CALL (MSEE) .....	3 -17
3.1.7 SUBROUTINE RETURN (RET) .....	3 -18
3.1.8 ONE SCAN WAIT (EOX) .....	3 -19
3.1.9 Branching Commands: IF...ELSE...IEND .....	3 -20
3.1.10 Repeat Commands: WHILE...WEND .....	3 -21
3.1.11 Parallel Execution Commands (PFORK, JOINTO, PJOINT) .....	3 -23
3.1.12 Selective Execution Commands (SFORK, JOINTO, SJOINT) .....	3 -27
<b>3.2 Speed and Acceleration/Deceleration Commands</b> .....	<b>3 -29</b>
3.2.1 ACCELERATION TIME CHANGE (ACC) .....	3 -29
3.2.2 DECELERATION TIME CHANGE (DCC) .....	3 -31
3.2.3 S-CURVE TIME CONSTANT CHANGE (SCC) .....	3 -32
3.2.4 SET SPEED (VEL) .....	3 -35
3.2.5 INTERPOLATION FEED SPEED RATIO SETTING (IFP) .....	3 -36
3.2.6 MAXIMUM INTERPOLATION FEED SPEED (FMX) .....	3 -38
3.2.7 INTERPOLATION ACCELERATION TIME CHANGE (IAC) .....	3 -40
3.2.8 INTERPOLATION DECELERATION TIME CHANGE (IDC) .....	3 -42

## 4 Sequence Commands

<b>4.1 Overview of Sequence Commands</b> .....	<b>4 -3</b>
4.1.1 Math Commands .....	4 -3
4.1.2 Combinations of Math Operations .....	4 -4
4.1.3 Combinations of Logic Operations .....	4 -4
<b>4.2 Arithmetic Operations</b> .....	<b>4 -6</b>
4.2.1 SUBSTITUTE (=) .....	4 -6
4.2.2 ADD (+) .....	4 -7
4.2.3 SUBTRACT (-) .....	4 -8
4.2.4 MULTIPLY (*) .....	4 -8
4.2.5 DIVIDE (/) .....	4 -9
4.2.6 REMAINDER (MOD) .....	4 -10
<b>4.3 Logic Operations</b> .....	<b>4 -11</b>
4.3.1 OR ( ) .....	4 -11
4.3.2 AND (&) .....	4 -12
4.3.3 XOR (^) .....	4 -13
4.3.4 NOT (!) .....	4 -14
<b>4.4 Data Comparisons</b> .....	<b>4 -16</b>
4.4.1 Data Comparison Commands (=, <, >, <=, >=, <=) .....	4 -16

<b>4.5 Data Operations</b> .....	<b>4 -18</b>
4.5.1 BIT RIGHT SHIFT (SFR) .....	4 -18
4.5.2 BIT LEFT SHIFT (SFL) .....	4 -19
4.5.3 BLOCK MOVE (BLK) .....	4 -20
4.5.4 CLEAR (CLR) .....	4 -22
4.5.5 ASCII CONVERSION 1 (ASCII) .....	4 -23
<b>4.6 Basic Functions</b> .....	<b>4 -25</b>
4.6.1 SINE (SIN) .....	4 -25
4.6.2 COSINE (COS) .....	4 -26
4.6.3 TANGENT (TAN) .....	4 -28
4.6.4 ARC SINE (ASN) .....	4 -29
4.6.5 ARC COSINE (ACS) .....	4 -30
4.6.6 ARC TANGENT (ATN) .....	4 -31
4.6.7 SQUARE ROOT (SQT) .....	4 -32
4.6.8 BCD-TO-BINARY (BIN) .....	4 -34
4.6.9 BINARY-TO-BCD (BCD) .....	4 -35
4.6.10 SET BIT (S{ }) .....	4 -36
4.6.11 RESET BIT (R{ }) .....	4 -37
<b>5 Variables (Registers)</b>	
<b>5.1 Overview</b> .....	<b>5 -2</b>
5.1.1 Overview of Variables .....	5 -2
5.1.2 Global Variables and Local Variables .....	5 -4
<b>5.2 Using Variables</b> .....	<b>5 -7</b>
5.2.1 System Variables (S Registers) .....	5 -7
5.2.2 Data Variables (M Registers) .....	5 -8
5.2.3 Input Variables (I Registers) .....	5 -9
5.2.4 Output Variables (O Registers) .....	5 -12
5.2.5 C Variables (C Registers) .....	5 -15
5.2.6 D Variables (D Registers) .....	5 -15
<b>6 MP2300S/MP2400 Exclusive-use Commands</b>	
<b>6.1 MP2300S/MP2400 Exclusive-use Commands</b> .....	<b>6 -2</b>
6.1.1 RISING PULSE (PON) .....	6 -2
6.1.2 FALLING PULSE (NON) .....	6 -4
6.1.3 ON-DELAY TIMER (TON): Counting unit: 0.01 second .....	6 -6
6.1.4 OFF-DELAY TIMER (TOF): Counting unit: 0.01 second .....	6 -7
6.1.5 USER FUNCTION CALL (FUNC) .....	6 -9
6.1.6 SEQUENCE SUBROUTINE CALL (SSEE) .....	6 -10

---

**A Motion Command List**

**A.1 Motion Command List ..... A - 2**

**B Motion Program Alarm List**

**B.1 Motion Program Alarm Storage Location ..... B -2**

**B.2 Motion Program Alarm List ..... B -4**

**Revision History**

# Overview

## ■ About this Manual

This manual provides information on motion commands for the MP900 and MP2000 series (herein-after referred to as MP series) Machine Controllers.

This manual describes the following items.

- Overview and specifications of motion programs and programming methods
- Basic programming
- Advanced programming
- Sequence commands
- Variables

Read this manual carefully to ensure the proper use of the MP series Machine Controller. Also, keep this manual in a safe place so that it can be referred to whenever necessary.

## ■ Related Manuals

The MP900-series of Machine Controllers consists of four models: the MP910, the MP920, the MP930, and the MP940.

The MP2000-series consists of six: the MP2100, the MP2100M, the MP2200, the MP2300, the MP2300S, and the MP2400.

The manuals have been written for these products.

Refer to the following related manuals as required.

Manual Name	Manual Number	Contents
MP930 Machine Controller User's Manual: Design and Maintenance	SIEZ-C887-1.1	Describes the functions, specifications, setup procedures, and operating methods of the MP930.
MP920 Machine Controller User's Manual: Design and Maintenance	SIEZ-C887-2.1	Describes the design and maintenance for the MP920 Machine Controllers.
MP910 Machine Controller User's Manual: Design and Maintenance	SIEZ-C887-3.1	Describes the functions, specifications, setup procedures, and operating methods of the MP910.
MP940 Machine Controller User's Manual: Design and Maintenance	SIEZ-C887-4.1	Describes the functions, specifications, setup procedures, and operating methods of the MP940.
MP2100 Machine Controller User's Manual: Design and Maintenance	SIEPC8807001	Describes the functions, specifications, setup procedures, and operating methods of the MP2100.



<b>Manual Name</b>	<b>Manual Number</b>	<b>Contents</b>
MP920 Machine Controller User's Manual: Motion Module	SIEZ-C887-2.5	Describes the functions, specifications, and usage of the MP920 Motion Modules (SVB-01, PO-01).
MP920 Machine Controller User's Manual: Communications Module	SIEZ-C887-2.6	Describes the functions, specifications, and operation methods of the MP920 Communications Module (217IF, 215IF, 218IF) in details.
MP2100/MP2100M Machine Controller User's Manual	SIEPC88070001	Describes the functions, specifications, setup procedures, and operating methods of the MP2100/MP2100M.
MP2200 Machine Controller User's Manual	SIEPC88070014	Describes the functions, specifications, setup procedures, and operating methods of the MP2200.
MP2300 Machine Controller Basic Module User's Manual	SIEPC88070003	Describes the functions, specifications, setup procedures, and operating methods of the MP2300.
MP2300 Machine Controller Communication Module User's Manual	SIEPC88070004	Describes the functions, specifications, and application methods of the MP2300 Communication Modules.
MP2300S Machine Controller Basic Module User's Manual	SIEPC88073200	Describes the functions, specifications, setup procedures, and operating methods of the MP2300S.
MP2400 Machine Controller User's Manual	SIEPC88074200	Describes the functions, specifications, setup procedures, and operating methods of the MP2400.
MP900/MP2000 Series Machine Controller User's Manual: Ladder Programming	SIEZ-C887-1.2	Describes the processing instructions used in MP-series Machine Controller ladder programs.
MP900/MP2000 Series Machine Controller MPE720 Software for Programming Device User's Manual	SIEPC88070005	Describes the installation and operation of the programming software MPE720 for MP series.
Engineering Tool for MP2000 Series Machine Controller MPE720 Version 6 User's Manual	SIEPC88070030	Describes the installation and operation of the programming software MPE720 for MP2000 series.
MP2200/MP2300 Machine Controller Motion Module User's Manual	SIEPC88070016	Describes the functions, specifications, and usage of the MP2200 and MP2300 Motion Modules.
MP2500/MP2500M User's Manual	SIEPC88075200	Describes the functions, specifications, setup procedures, and operating methods of the MP2500/MP2500M.

## Using This Manual

### ■ Intended Audience

This manual is intended for the following users.

- Those responsible for designing the MP series Machine Controller system
- Those responsible for writing MP series Machine Controller motion programs

### ■ Description of Technical Terms, Abbreviations, and Symbols

- MPE720 = MPE720 software
- Programming Device = Personal computer where the MPE720 is installed.
- MP900: Model series including MP910, MP920, MP930, MP940
- MP2000: Model series including MP2100, MP2100M, MP2200, MP2300, MP2300S, MP2400

# Safety Precautions

This section describes important precautions that apply to motion programming. Before programming, always read this manual and all other attached documents to ensure correct programming. Before using the equipment, familiarize yourself with equipment details, safety information, and all other precautions.

## ■ Application Precautions



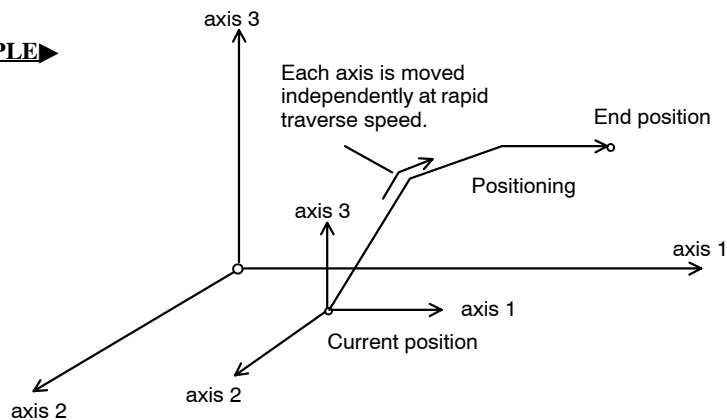
### Caution

When programming the following axis move commands, check the path to make sure that there are no tools or other obstacles in the way of the workpiece.

The axis move commands that must be checked are as follows:

- POSITIONING (MOV)
- LINEAR INTERPOLATION (MVS)
- CIRCULAR INTERPOLATION (MCC, MCW)
- HELICAL INTERPOLATION (MCC, MCW)
- SET TIME POSITIONING (MVT)
- SKIP (SKP)
- ZERO POINT RETURN (ZRN)
- EXTERNAL POSITIONING (EXM)

#### ◀EXAMPLE▶



Example of Basic Path for POSITIONING (MOV)

Failure to carry out the above checks may result in damage to equipment, serious personal injury, or even death.



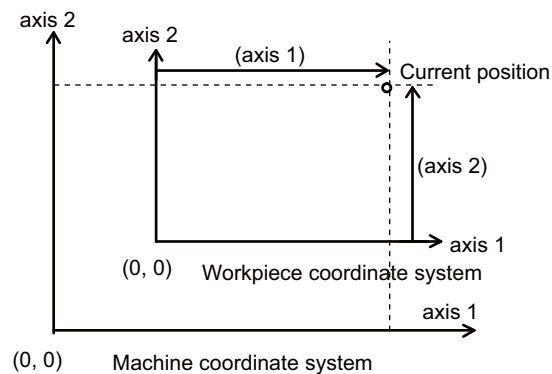
## Caution

- If the following coordinate commands are designated incorrectly, the subsequent move operations will be entirely different than those expected. Before starting operations, be sure to check that the settings are designated correctly.

The coordinate commands that must be checked are as follows:

- ABSOLUTE PROGRAMMING MODE (ABS)
- INCREMENTAL PROGRAMMING MODE (INC)
- CURRENT POSITION SET (POS)
- MOVE ON MACHINE COORDINATES (MVM)

### ◀ EXAMPLE ▶



**Example of Workpiece Coordinate System Created with CURRENT POSITION SET (POS)**

Failure to carry out the above checks may result in damage to equipment, serious personal injury, or even death.

### ■ General Precautions

#### Observe the following general precautions to ensure safe application.

- The products shown in illustrations in this manual are sometimes shown without covers or protective guards. Always replace the cover or protective guard as specified first, and then operate the products in accordance with the manual.
- The drawings presented in this manual are typical examples and may not match the product you received.
- If the manual must be ordered due to loss or damage, inform your nearest Yaskawa representative or one of the offices listed on the back of this manual.

---

# Warranty

## ■ Details of Warranty

### Warranty Period

The warranty period for a product that was purchased (hereinafter called “delivered product”) is one year from the time of delivery to the location specified by the customer or 18 months from the time of shipment from the Yaskawa factory, whichever is sooner.

### Warranty Scope

Yaskawa shall replace or repair a defective product free of charge if a defect attributable to Yaskawa occurs during the warranty period above. This warranty does not cover defects caused by the delivered product reaching the end of its service life and replacement of parts that require replacement or that have a limited service life.

This warranty does not cover failures that result from any of the following causes.

1. Improper handling, abuse, or use in unsuitable conditions or in environments not described in product catalogs or manuals, or in any separately agreed-upon specifications
2. Causes not attributable to the delivered product itself
3. Modifications or repairs not performed by Yaskawa
4. Abuse of the delivered product in a manner in which it was not originally intended
5. Causes that were not foreseeable with the scientific and technological understanding at the time of shipment from Yaskawa
6. Events for which Yaskawa is not responsible, such as natural or human-made disasters

## ■ Limitations of Liability

1. Yaskawa shall in no event be responsible for any damage or loss of opportunity to the customer that arises due to failure of the delivered product.
2. Yaskawa shall not be responsible for any programs (including parameter settings) or the results of program execution of the programs provided by the user or by a third party for use with programmable Yaskawa products.
3. The information described in product catalogs or manuals is provided for the purpose of the customer purchasing the appropriate product for the intended application. The use thereof does not guarantee that there are no infringements of intellectual property rights or other proprietary rights of Yaskawa or third parties, nor does it construe a license.
4. Yaskawa shall not be responsible for any damage arising from infringements of intellectual property rights or other proprietary rights of third parties as a result of using the information described in catalogs or manuals.

## ■ Suitability for Use

1. It is the customer's responsibility to confirm conformity with any standards, codes, or regulations that apply if the Yaskawa product is used in combination with any other products.
2. The customer must confirm that the Yaskawa product is suitable for the systems, machines, and equipment used by the customer.
3. Consult with Yaskawa to determine whether use in the following applications is acceptable. If use in the application is acceptable, use the product with extra allowance in ratings and specifications, and provide safety measures to minimize hazards in the event of failure.
  - Outdoor use, use involving potential chemical contamination or electrical interference, or use in conditions or environments not described in product catalogs or manuals
  - Nuclear energy control systems, combustion systems, railroad systems, aviation systems, vehicle systems, medical equipment, amusement machines, and installations subject to separate industry or government regulations
  - Systems, machines, and equipment that may present a risk to life or property
  - Systems that require a high degree of reliability, such as systems that supply gas, water, or electricity, or systems that operate continuously 24 hours a day
  - Other systems that require a similar high degree of safety
4. Never use the product for an application involving serious risk to life or property without first ensuring that the system is designed to secure the required level of safety with risk warnings and redundancy, and that the Yaskawa product is properly rated and installed.
5. The circuit examples and other application examples described in product catalogs and manuals are for reference. Check the functionality and safety of the actual devices and equipment to be used before using the product.
6. Read and understand all use prohibitions and precautions, and operate the Yaskawa product correctly to prevent accidental harm to third parties.

## ■ Specifications Change

The names, specifications, appearance, and accessories of products in product catalogs and manuals may be changed at any time based on improvements and other reasons. The next editions of the revised catalogs or manuals will be published with updated code numbers. Consult with your Yaskawa representative to confirm the actual specifications before purchasing a product.

---

## Overview of Motion Programs

This chapter explains how to create motion programs. Motion programs are input with the Programming Device, and then transferred to the MP series Machine Controller for execution.

<b>1.1 Motion Programming</b> .....	<b>1 -2</b>
1.1.1 Motion Programming Capabilities .....	1 -2
1.1.2 Function Performance .....	1 -5
1.1.3 Starting Motion Programs .....	1 -14
1.1.4 Parallel Program Execution .....	1 -21
1.1.5 Editing Programs .....	1 -22
<b>1.2 Programming Methods</b> .....	<b>1 -23</b>
1.2.1 Programming Format .....	1 -23
1.2.2 Controlled Axes .....	1 -28
1.2.3 Feed Speeds .....	1 -35
1.2.4 Motion Commands .....	1 -39

# 1.1 Motion Programming

This section gives an overview of motion programming. Be sure to read through this section before starting to program.

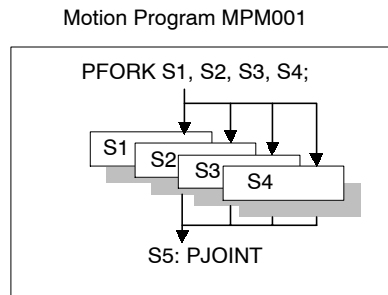
## 1.1.1 Motion Programming Capabilities

The MP series Machine Controller can be programmed to perform motions required for industrial machines.

For reference, the following examples illustrate the main features of motion programs.

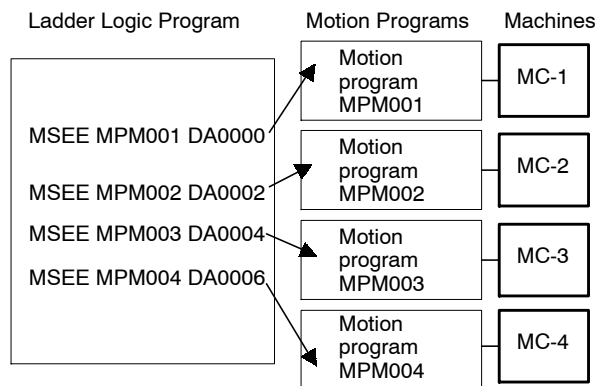
### ■ Parallel Execution

- Up to four program blocks can be executed in parallel within one motion program application.
- Up to four program blocks can be executed in parallel using the PFORK command.
- A total of four parallel axes can be freely controlled.



### ■ Simultaneous Control of Multiple Machines

Multiple machines can be controlled simultaneously by one MP series Machine Controller.

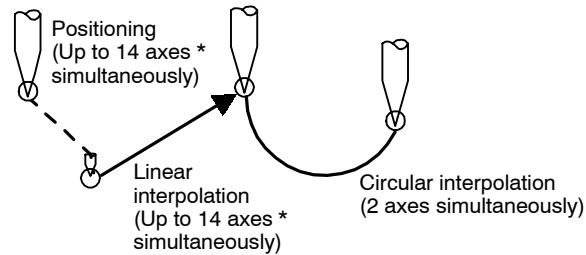




## Wide Choice of Move Commands

- POSITIONING: Up to 14 axes \*
- LINEAR INTERPOLATION: Up to 14 axes \*
- CIRCULAR INTERPOLATION: Two axes
- HELICAL INTERPOLATION: Three axes

The basic commands can be easily specified.



\* Indicates the maximum number of axes that the MP930 can control simultaneously.

Machine Controller	Max. Number of Controlled Axes	Max. Number of Controller Axes for Linear Interpolation
MP910	28	14
MP920	60	16
MP2100	16	16
MP2100M	32	16
MP2200	256	16
MP2300	48	16
MP2500	16	16
MP2500M	32	16

## Very Flexible Calculations with Many Math Commands

- Integer addition, subtraction, multiplication, and division
- Real number addition, subtraction, multiplication, and division
- Logic operations
- Trigonometric functions
- Exponents
- Logarithms

Example of math commands

```
MW81D = AB01H*SIN (100) + 50;
MW5 = BCD (MW0001);
IF MF467 < > MF897;
IF MW0001 > 7;
IF OB0 = = 1;
OB0 =(IB0|IB2|IB3) & 1B1;
MW6 = SQT(MW809);
```

1

## ■ Control Commands

- CONDITIONAL BRANCH
- REPEAT
- TIMER (TIM)
- SUBROUTINE (MSEE)
- PARALLEL FORK (PFORK)
- SELECTIVE FORK (SFORK)
- Etc.

### CONDITIONAL BRANCH

IF            <Conditional expression>

- 
- Processing when condition is satisfied
- 
- 

ELSE

- 
- Processing when condition is not satisfied
- 

IEND

### REPEAT

WHITE      <Conditional expression>

- 
- Processing
- 

WEND

## 1.1.2 Function Performance

### ■ MP910 Motion Control Function Specifications

Table 1.1 lists the motion control function specifications for the MP910.

**Table 1.1 MP910 Motion Control Function Specifications**

Item		Specification
<b>Number of Controlled Axes</b>		1 to 28 axes (14 axes/port, two ports max.)
<b>Control Specifications</b>	<b>PTP Control</b>	Linear, rotary, and infinite-length axes
	<b>Interpolation</b>	Up to 14 axes for linear interpolation, 2 axis for circular interpolation, 3 axis for helical interpolation
	<b>Speed Control</b>	None
	<b>Torque Limit</b>	Yes (According to parameter settings only)
<b>Reference Unit</b>		mm, inch, deg, pulse
<b>Reference Unit Minimum Setting</b>		1, 0.1, 0.01, 0.001, 0.0001, 0.00001
<b>Maximum Programmable Value</b>		-2147483648 to +2147483647 (signed 32-bit value)
<b>Speed Reference Unit</b>		mm/min, inch/min, deg/min, pulse/min
<b>Acceleration/Deceleration Type</b>		Linear, asymmetric, S-curve Asymmetric acceleration/deceleration is not possible with POSITIONING (MOV).
<b>Override Function</b>		Positioning: 0.01% to 327.67% by axis Interpolation: 0.01% to 327.67% by group
<b>Coordinate System</b>		Rectangular coordinates
<b>Zero Point Return</b>		Four types Dog + Phase C, zero point limit switch, dog + Zero point limit switch, Phase C  Zero point setting function provided.
<b>Programs</b>	<b>Language</b>	Special motion language
	<b>Number of Tasks</b>	Up to four programs can be executed in parallel.
	<b>Number of Programs</b>	Up to 256
	<b>Program Capacity</b>	Equivalent to 80 Kbytes (characters) (Can be increased or decreased according to the size of ladder logic program used; maximum of 100 Kbytes.)

1

Item	Specification
<b>Command Words</b>	Axis Move Commands: 8 commands MOV, MVS, MCW, MCC, ZRN, SKP, MVT, EXM  Basic Control Commands: 6 commands ABS, INC, POS, PLN, MVM, PLD  Speed and Acceleration/Deceleration Commands: 7 commands ACC, SCC, VEL, IAC, IDC, IFP, FMX  High-level Control Commands: 4 commands PFN, INP, SNG, UFC  Control Commands: 10 commands MSEE, TIM, IOW, END, RET, EOX, IF ELSE IEND, WHILE WEND, PFORK JOINTO PJOINT, SFORK JOIN- TO SJOINT  Math and Sequence Control Commands: 36 commands =, +, -, *, /, MOD,  , ^, &, !, (), S{ }, R{ }, SIN, COS, TAN, ASN, ACS, ATN, SQRT, BIN, BCD, ==, <>, >, <, >=, <=, TON, TOF, SFR, SFL, PON, NON, BLK, CLR

■ **MP920 Motion Control Function Specifications**

Table 1.2 lists the motion control function specifications for the MP920.

**Table 1.2 MP920 Motion Control Function Specifications**

Item	Specification	
<b>Number of Controlled Axes</b>	1 to 60 axes (when SVA-01 Module is used.)	
<b>Control Specifications</b>	<b>PTP Control</b>	Linear, rotary, and infinite-length axes
	<b>Interpolation</b>	Up to 16 axes for linear interpolation, 2 axis for circular interpolation, 3 axis for helical interpolation
	<b>Speed Reference Output</b>	Available with SVA-01 and SVA02 Modules.
	<b>Torque Reference Output</b>	Available with SVA02 Module.
	<b>Position Control</b>	Positioning, external positioning, zero point return, interpolation, interpolation with position detection function, fixed speed feed, fixed length feed
	<b>Phase Control</b>	Available with SVA-01 and SVA02 Modules.
<b>Reference Unit</b>	mm, inch, deg, pulse	
<b>Reference Unit Minimum Setting</b>	1, 0.1, 0.01, 0.001, 0.0001, 0.00001	
<b>Maximum Programmable Value</b>	-2147483648 to +2147483647 (signed 32-bit value)	
<b>Speed Reference Unit</b>	mm/min, inch/min, deg/min, pulse/min	
<b>Acceleration/Deceleration Type</b>	Linear, asymmetric, S-curve	
<b>Override Function</b>	Positioning: 0.01% to 327.67% by axis Interpolation: 0.01% to 327.67% by group	

Item		Specification
<b>Coordinate System</b>		Rectangular coordinates
<b>Zero Point Return</b>		Eight types 1. DEC1 + Phase C                    5. DEC1 + ZERO 2. DEC2 + Phase C                    6. DEC2 + ZERO 3. DEC1 + LMT                         7. DEC1 + LMT + ZERO 4. Phase C                               8. ZERO
<b>Programs</b>	<b>Language</b>	Special motion language, ladder language
	<b>Number of Tasks</b>	Up to eight programs can be executed in parallel.
	<b>Number of Programs</b>	Up to 256
	<b>Program Capacity</b>	80 Kbytes
<b>Applicable Servopack</b>		<ul style="list-style-type: none"> <li>• Analog SGDA-□/SGDB-□□□/SGDM-□□</li> <li>• Network SGD-□N/SGDB-□N</li> </ul>
<b>Encoder</b>		Incremental or absolute
<b>Command Words</b>		Axis Move Commands: 8 commands MOV, MVS, MCW, MCC, ZRN, SKP, MVT, EXM  Basic Control Commands: 6 commands ABS, INC, POS, PLN, MVM, PLD  Speed and Acceleration/Deceleration Commands: 8 commands ACC, SCC, VEL, IAC, IDC, IFP, FMX  High-level Control Commands: 4 commands PFN, INP, SNG, UFC  Control Commands: 10 commands MSEE, TIM, IOW, END, RET, EOX, IF ELSE IEND, WHILE WEND, PFORK JOINTO PJOINT, SFORK JOIN- TO SJOINT  Math and Sequence Control Commands: 36 commands =, +, -, *, /, MOD,  , ^, &, !, (), S{ }, R{ }, SIN, COS, TAN, ASN, ACS, ATN, SQRT, BIN, BCD, ==, <>, >, <, >=, <=, TON, TOF, SFR, SFL, PON, NON, BLK, CLR

■ **MP930 Motion Control Function Specifications**

Table 1.3 lists the motion control function specifications for the MP930.

**Table 1.3 MP930 Motion Control Function Specifications**

Item		Specification
<b>Number of Controlled Axes</b>		1 to 14 axes
<b>Control Specifications</b>	<b>PTP Control</b>	Linear, rotary, and infinite-length axes
	<b>Interpolation</b>	Up to 14 axes for linear interpolation, 2 axes for circular interpolation, and 3 axes for helical interpolation
	<b>Speed Control</b>	None
	<b>Torque Limit</b>	Yes (According to parameter setting only)
<b>Reference Unit</b>		mm, inch, deg, pulse
<b>Reference Unit Minimum Setting</b>		1, 0.1, 0.01, 0.001, 0.0001, 0.00001
<b>Maximum Programmable Value</b>		-2147483648 to +2147483647 (signed 32-bit)
<b>Speed Reference Unit</b>		mm/min, inch/min, deg/min, pulse/min
<b>Acceleration/Deceleration Type</b>		Linear, asymmetric, S-curve Asymmetric acceleration/deceleration is not possible with POSITIONING (MOV).
<b>Override Function</b>		Positioning: 0.00% to 327.67% per axis Interpolation: 0.00% to 327.67% per group
<b>Coordinate System</b>		Rectangular coordinates
<b>Zero Point Return</b>		Four types Dog + Phase C, zero point limit switch, dog + home position limit switch, Phase C  Zero point setting function provided.
<b>Programs</b>	<b>Language</b>	Special motion language
	<b>Number of Tasks</b>	Up to four programs can be executed in parallel.
	<b>Number of Programs</b>	Up to 256
	<b>Program Capacity</b>	Equivalent to 80 K bytes (characters) (Can be increased or decreased according to the size of ladder logic program used, to up to a maximum of 100 KB.)

1

Item	Specification
<b>Command Words</b>	<p>Axis Move Commands: 8 commands MOV, MVS, MCW, MCC, ZRN, SKP, MVT, EXM</p> <p>Basic Control Commands: 6 commands ABS, INC, POS, PLN, MVM, PLD</p> <p>Speed and Acceleration/Deceleration Commands: 8 commands ACC, DCC, SCC, VEL, IAC, IDC, IFP, FMX</p> <p>High-level Control Commands: 4 commands PFN, INP, SNG, UFC</p> <p>Control Commands: 10 commands MSEE, TIM, IOW, END, RET, EQX, IF ELSE IEND, WHILE WEND, PFORK JOINTO PJOINT, SFORK JOIN- TO SJOINT</p> <p>Math and Sequence Control Commands: 32 commands =, +, -, *, /, MOD,  , ^, &amp;, !, (), S{ }, R{ }, SIN, COS, TAN, ASN, ACS, ATN, SQRT, BIN, BCD, ==, &lt;&gt;, &gt;, &lt;, &gt;=, &lt;=, SFR, SFL, BLK, CLR</p>

## ■ MP940 Motion Control Function Specifications

Table 1.4 lists the motion control function specifications for the MP940.

**Table 1.4 MP940 Motion Control Function Specifications**

Item	Specification	
<b>Number of Controlled Axes</b>	1 axis	
<b>Control Specifications</b>	<b>PTP Control</b>	Linear, rotary, and infinite-length axes
	<b>Interpolation</b>	Linear
	<b>Speed Reference Output</b>	Yes
	<b>Torque Reference Output</b>	Yes
	<b>Position Control</b>	Positioning, external positioning, zero point return, interpolation, interpolation with position detection function, fixed speed feed, fixed length feed
	<b>Phase Control</b>	Yes

# Overview of Motion Programs

## 1.1.2 Function Performance

1

Item		Specification
<b>Position Control</b>	<b>Reference Unit</b>	mm, inch, deg, pulse
	<b>Reference Unit Minimum Setting</b>	1, 0.1, 0.01, 0.001, 0.0001, 0.00001
	<b>Maximum Programmable Value</b>	-2147483648 to +2147483647 (signed 32-bit value)
	<b>Speed Reference Unit</b>	mm/min, inch/min, deg/min, pulse/min
	<b>Acceleration/Deceleration Type</b>	Linear, asymmetric, S-curve
	<b>Override Function</b>	0 to 327.67% (0.01% scale)
<b>Coordinate System</b>		Rectangular coordinates
<b>Zero Point Return</b>		Eight types <ul style="list-style-type: none"> <li>• DEC1 + Phase C</li> <li>• DEC2 + Phase C</li> <li>• DEC1 + LMT</li> <li>• Phase C</li> <li>• DEC1 + ZERO</li> <li>• DEC2 + ZERO</li> <li>• DEC1 + LMT + ZERO</li> <li>• ZERO</li> </ul>
<b>Programs</b>	<b>Language</b>	Special motion language, ladder language
	<b>Number of Tasks</b>	Up to 8 axes can be executed in parallel.
	<b>Number of Programs</b>	Up to 32
	<b>Program Capacity</b>	80 Kbytes
<b>Speed Control</b>	<b>Speed Reference</b>	-327.68 to +327.67%/rated speed With torque limit function
	<b>Acceleration/Deceleration Type</b>	Linear, asymmetric, S-curve (averaging movement)
<b>Torque Control</b>	<b>Torque Reference</b>	-327.68 to +327.67%/rated torque With speed limit function
<b>Phase Control</b>	<b>Speed Reference Unit</b>	-327.68 to +327.67%/rated speed
	<b>Speed Correction</b>	-327.68 to +327.67%/rated speed
	<b>Position Correction</b>	-2147483648 to +2147483647 pulses



Item	Specification
<b>Command Words</b>	<p>Axis Move Commands: 5 commands MOV, MVS, ZRN, SKP, EXM</p> <p>Basic Control Commands: 5 commands ABS, INC, POS, MVM, PLD</p> <p>Speed and Acceleration/Deceleration Commands: 8 commands ACC, DCC, SCC, VEL, IAC, IDC, IFP, FMX</p> <p>High-level Control Commands: 4 commands PFN, INP, SNG, UFC</p> <p>Control Commands: 10 commands MSEE, TIM, IOW, END, RET, EOX, IF ELSE IEND, WHILE WEND, PFORK JOINTO PJOINT, SFORK JOIN- TO SJOINT</p> <p>Math and Sequence Control Commands: 32 commands =, +, -, *, /, MOD,  , ^, &amp;, !, (), S{ }, R{ }, SIN, COS, TAN, ASN, ACS, ATN, SQRT, BIN, BCD, ==, &lt;&gt;, &gt;, &lt;, &gt;=, &lt;=, SFR, SFL, BLK, CLR</p>

## ■ MP2100 Motion Control Function Specifications

Table 1.5 lists the motion control function specifications for the MP2100.

**Table 1.5 MP2100 Motion Control Function Specifications**

Item	Specification	
<b>Number of Controlled Axes</b>	1 to 16 axes	
<b>Control Specifications</b>	<b>PTP Control</b>	Linear, rotary, and infinite-length axes
	<b>Interpolation</b>	Up to 16 axes for linear interpolation, to 2 axes for circular interpolation, to 3 axes for helical interpolation
	<b>Speed Reference Output</b>	Yes
	<b>Torque Reference Output</b>	Yes
	<b>Position Control</b>	Positioning, external positioning, zero point return, interpolation, interpolation with position detection function, fixed speed feed, fixed length feed
	<b>Phase Control</b>	Yes
<b>Reference Unit</b>	mm, inch, deg, pulse	
<b>Reference Unit Minimum Setting</b>	1, 0.1, 0.01, 0.001, 0.0001, 0.00001	
<b>Maximum Programmable Value</b>	-2147483648 to +2147483647 (signed 32-bit value)	
<b>Speed Reference Unit</b>	mm/sec, inch/sec, deg/sec, pulse/sec, pulse/sec, mm/min, inch/min, deg/min, pulse/min, %	
<b>Acceleration/Deceleration Type</b>	Linear, asymmetric, S-curve	
<b>Override Function</b>	Positioning: 0.01 to 327.67% per axis Interpolation: 0.01 to 327.67% per MSEE command	

# Overview of Motion Programs

## 1.1.2 Function Performance

Item		Specification
<b>Coordinate System</b>		Rectangular coordinates
<b>Zero Point Return</b>		Thirteen types <ul style="list-style-type: none"> <li>• DEC1 + Phase C                      • HOME LS + C Pulse</li> <li>• Phase C                                • HOME LS</li> <li>• DEC1 + ZERO                        • NOT + C Pulse</li> <li>• ZERO                                    • NOT</li> <li>• C Pulse                                • Input + C Pulse</li> <li>• POT + C Pulse                        • Input</li> <li>• POT</li> </ul>
<b>Programs</b>	<b>Language</b>	Special motion language, ladder language
	<b>Number of Tasks</b>	Up to 16 axes can be executed in parallel.
	<b>Number of Programs</b>	Up to 256
	<b>Program Capacity</b>	1200 Kbytes
<b>Command Words</b>		Axis Move Commands: 12 commands MOV, MVS, MCW, MCC, ZRN, SKP, MVT, EXM, VCS, VCR, TCS, TCR  Basic Control Commands: 6 commands ABS, INC, POS, PLN, MVM, PLD  Speed and Acceleration/Deceleration Commands: 8 commands ACC, DCC, SCC, VEL, IAC, IDC, IFP, FMX  High-level Control Commands: 4 commands PFN, INP, SNGD, SNGE, UFC  Control Commands: 11 commands MSEE, TIM, IOW, END, RET, EOX, IF ELSE IEND, WHILE WEND, PFORK JOINTO, PJOINT, SFORK, JOINTO, SJOINT  Math and Sequence Control Commands: 32 commands =, +, -, *, /, MOD,  , ^, &, !, (), S{ }, R{ }, SIN, COS, TAN, ASN, ACS, ATN, SQRT, BIN, BCD, ==, <>, >, <, >=, <=, SFR, SFL, BLK, CLR

## ■ MP2300 Motion Control Function Specifications

Table 1.6 lists the motion control function specifications for the MP2300.

**Table 1.6 MP2300 Motion Control Function Specifications**

Item		Specification
<b>Number of Controlled Axes</b>		1 to 16 axes
<b>Control Specifications</b>	<b>PTP Control</b>	Linear, rotary, and infinite-length axes
	<b>Interpolation</b>	Up to 16 axes for linear interpolation, to 2 axes for circular interpolation, to 3 axes for helical interpolation
	<b>Speed Reference Output</b>	Yes
	<b>Torque Reference Output</b>	Yes
	<b>Position Control</b>	Positioning, external positioning, zero point return, interpolation, interpolation with position detection function, fixed speed feed, fixed length feed
	<b>Phase Control</b>	Yes
<b>Reference Unit</b>		mm, inch, deg, pulse
<b>Reference Unit Minimum Setting</b>		1, 0.1, 0.01, 0.001, 0.0001, 0.00001
<b>Maximum Programmable Value</b>		-2147483648 to +2147483647 (signed 32-bit value)
<b>Speed Reference Unit</b>		mm/sec, inch/sec, deg/sec, pulse/sec, pulse/sec, mm/min, inch/min, deg/min, pulse/min, %
<b>Acceleration/Deceleration Type</b>		Linear, asymmetric, S-curve
<b>Override Function</b>		Positioning: 0.01 to 327.67% per axis Interpolation: 0.01 to 327.67% per MSEE command
<b>Coordinate System</b>		Rectangular coordinates
<b>Zero Point Return</b>		Thirteen types <ul style="list-style-type: none"> <li>• DEC1 + Phase C</li> <li>• Phase C</li> <li>• DEC1 + ZERO</li> <li>• ZERO</li> <li>• C Pulse</li> <li>• POT + C Pulse</li> <li>• POT</li> <li>• HOME LS + C Pulse</li> <li>• HOME LS</li> <li>• NOT + C Pulse</li> <li>• NOT</li> <li>• Input + C Pulse</li> <li>• Input</li> </ul>
<b>Programs</b>	<b>Language</b>	Special motion language, ladder language
	<b>Number of Tasks</b>	Up to 16 axes can be executed in parallel.
	<b>Number of Programs</b>	Up to 256
	<b>Program Capacity</b>	1200 Kbytes

1

Item	Specification
<b>Command Words</b>	Axis Move Commands: 12 commands MOV, MVS, MCW, MCC, ZNR, SKP, MVT, EXM, VCS, VCR, TCS, TCR  Basic Control Commands: 6 commands ABS, INC, POS, PLN, MVM, PLD  Speed and Acceleration/Deceleration Commands: 8 commands ACC, DCC, SCC, VEL, LAC, IDC, IFP, FMX  High-level Control Commands: 4 commands PFN, INP, SNG, UFC  Control Commands: 11 commands MSEE, TIM, JOW, END, RET, BOX, IF ELSE IEND, WHILE WEND, PFORK JOINTO PJOINT, SFORK JOINTO SJOINT, GOTO  Math and Sequence Control Commands: 32 commands =, +, -, *, /, MOD,  , ^, &, !, (), S{ }, R{ }, SIN, COS, TAN, ASN, ACS, ATN, SQRT, BIN, BCD, ==, <>, >, <, >=, <=, SFR, SFL, BLK, CLR  Table Data Operation Commands: 2 commands TBLBR, TBLBW  Character String Operation Commands: 3 commands ASCII, BINASC, ASCBIN

### 1.1.3 Starting Motion Programs

Motion programming is a textual motion programming language. Motion programming can be used to create 256 programs separate from the ladder drawings.

Two types of motion program are provided: Main programs (MPM□□□) that can be called from DWG.H, and subroutines (MPS□□□) that can be called from the main programs.

**Table 1.7 Types of Motion Program**

Classification	Designation Method	Feature	Number of Programs
<b>Main programs</b>	MPM□□□ 1 to 256	Can be called from DWG.H.	A total of up to 256 main programs and subroutines can be created.
<b>Subroutines</b>	MPS□□□ 1 to 256	Can be called from the main programs.	

**IMPORTANT**

The same MPM or MPS number cannot be used more than once.

There are two methods of designating a motion program: Direct designation of the program number, and indirect designation of the number of the register in which the program number is stored.

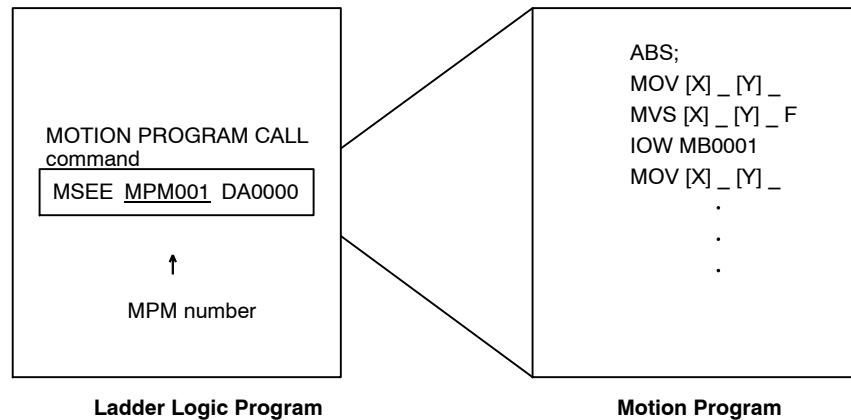


Figure 1.1 Starting a Motion Program by Direct Designation

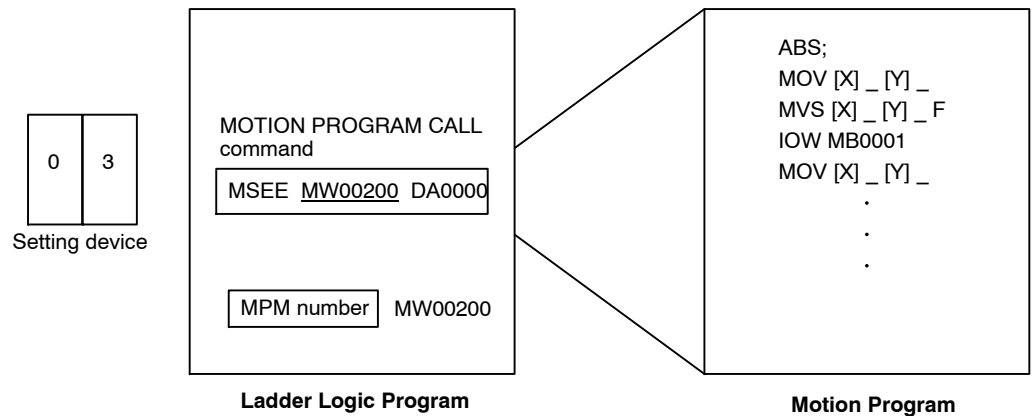
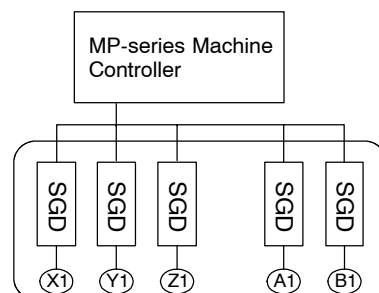


Figure 1.2 Starting a Motion Program by Indirect Designation

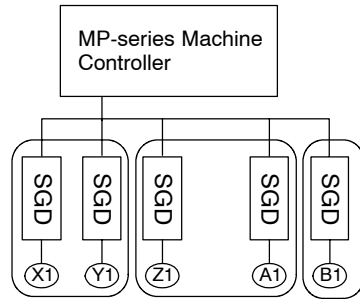
## ■ Group

With the MP-series Machine Controllers, the axes can be grouped by operation so that one Machine Controller can independently control several machines. As a result, each axis group can be separately programmed. The axes to be included in a group are defined in the Group Definition settings. Two types of group operations are available: Single Group and Multiple Groups. Refer to *MP900/MP2000 Series MPE Software for Programming Device User's Manual (SIEPC88070005)* for more information on Group Definition settings.

### Single Group Operation

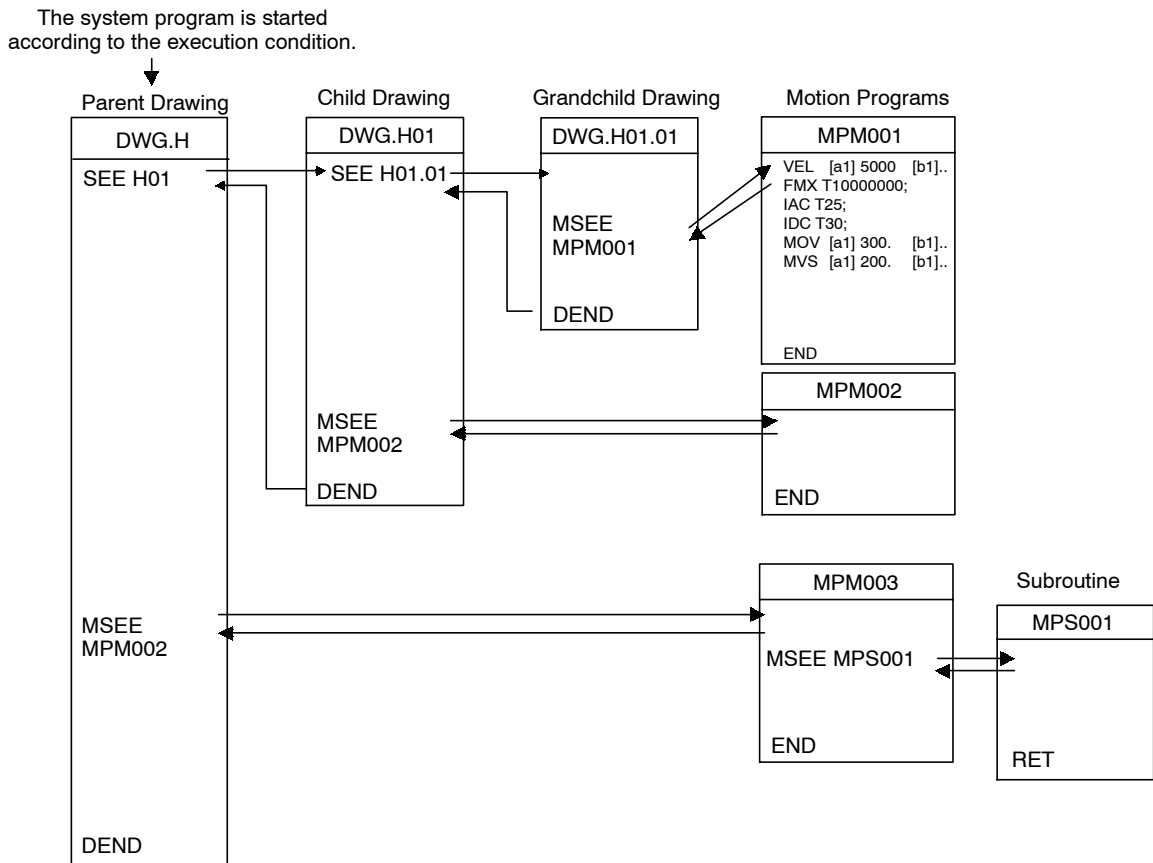


### Multiple Groups Operation



### ■ Motion Program Execution Processing Method

Motion programs must be called from DWG.H using the MSEE command. H drawings can be called from parent, child, or grandchild drawings.



- In each high-speed scanning cycle, the ladder logic commands for DWG. H are executed in the hierarchical order: Parent drawing→Child drawing→Grandchild drawing.
- Motion programs are called in the scanning cycle, but unlike ladder logic programs, all programs cannot be executed in a single scan. So, the motion-management functions of the system control the execution of the motion program.

**IMPORTANT**

1. Commands not completely processed in one scan  
MOV, MVS, MCW/MCC, ZRN, SKP, MVT, EXM, ACC, DCC, SCC, PFN, VEL, INP, TIM, IOW
2. Commands completely processed in one scan  
ABS, INC, IFP, PLN, IAC, IDC, FMX, POS, PLD, MSEE, END, RET, IF, WHILE, PFORK, SFORK,  
all sequence commands



## Restrictions on the Motion Program

1. More than one motion program with the same number cannot be called with an MSEE command.
2. An MPS□□□ subroutine cannot be called with a ladder-logic MSEE command. The subroutine can be called only from the motion program (MPM□□□ or MPS□□□).
3. One same subroutine cannot be called from two different locations at the same time.

## ■ Work Registers

The work registers are required to set or monitor the status of the motion program. The configuration of motion program work register is shown in the following table.

<b>1st word</b>	Motion-program status flag
<b>2nd word</b>	Motion-program control signal
<b>3rd word</b>	Interpolation override
<b>4th word</b>	System work number

**Note** The third and fourth words are only for MP2000-series Machine Controllers.

## ■ Motion Program Control Signals

A motion-program control signal, such as “Program run start request” or “Program stop request”, must be input to execute the motion program that was called from DWG. H with the MSEE command. The second word in the MSEE work register contains the control signal for the motion program. The following types of signals are used.

Bit No.	Signal Name	Signal Type
<b>0</b>	Program run start request	Differential or N.O (Normally Open) contact input
<b>1</b>	Program pause request	N.O contact
<b>2</b>	Program stop request	N.O contact
<b>3</b>	Program single-block mode selection	N.O contact
<b>4</b>	Request for program single-block start	Differential or N.O contact input
<b>5</b>	Alarm reset request	N.O contact
<b>6</b>	Request for start of program continuous run	Differential or N.O contact input (only for MP2000-series Machine Controllers)
<b>8</b>	Skip 1 information	N.O contact
<b>9</b>	Skip 2 information	N.O contact
<b>D</b>	System work number setting <sup>*1</sup>	N.O contact (only for MP2000-series Machine Controllers)
<b>E</b>	Interpolation override selection <sup>*2</sup>	N.O contact (only for MP2000-series Machine Controllers)

- \* 1. System work number setting  
 OFF: The work that the system automatically fetched is used. The system work number may change every time.  
 ON: The work of the preset system work number is used.
- \* 2. Interpolation override selection  
 OFF: The interpolation override is fixed at 100%.  
 ON: According to the preset interpolation override

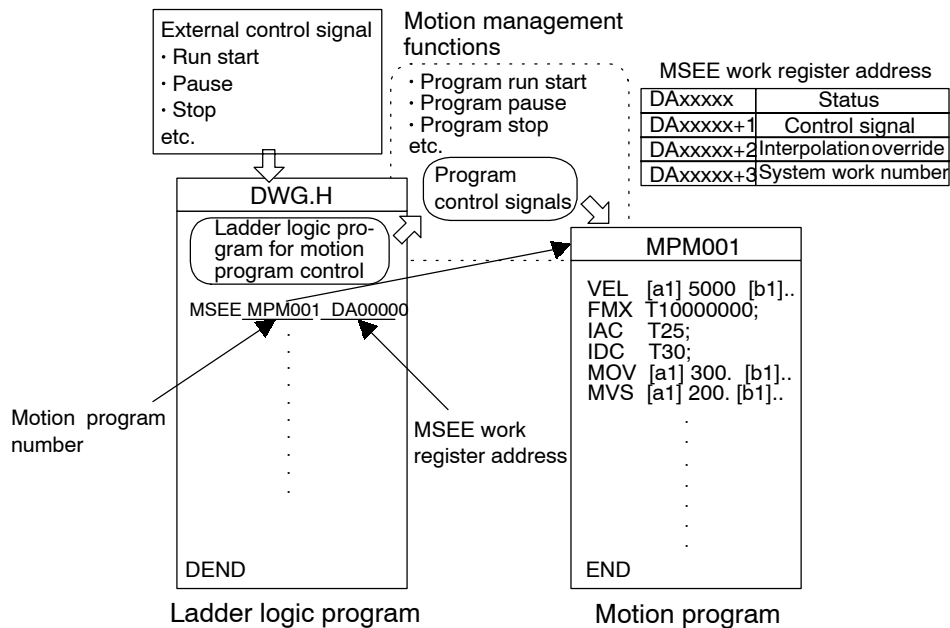
When the previously listed signals are sent to the work register designated by the MSEE command + 1 with the ladder logic program, the motion-program operations such as run start, stop, and pause can be performed by using the motion-management functions of the system.

Use the correct type of signal to input a ladder logic program.

**IMPORTANT**

The program starts running if the start signal is ON when the power supply is turned ON.

The process for the execution of a motion program is illustrated in the following diagram.





## ■ Motion Program Status Flag

The first word of the MSEE work register contains the motion-program status flag. The status flag can be read to monitor the status of the motion program being executed. The following table shows the meaning of each status flag.

Bit No.	Status
0	Program is running.
1	Program is temporarily stopped.
2	Program is stopped after receiving a request to stop the program (Used by the system.)
3	(Used by the system.)
4	Program single block operation has been stopped.
8	Program alarm is occurring.
9	Program is stopped at break point.
B	In debugging mode (EWS debug operation).
D	Start request signal history (only for MP2000-series).
E	MP900-series: Error (main program overlapped). MP2000-series: Error (No system work).
F	Main program number out of range.

## ■ Interpolation Override

In the third word of the MSEE work register, set the override for the execution of interpolation move command.

Setting Unit: 1 = 0.01%

This setting is valid only when Bit E (Interpolation Override Selection) of the control signal for the motion program is set to ON.

## ■ System Work Number

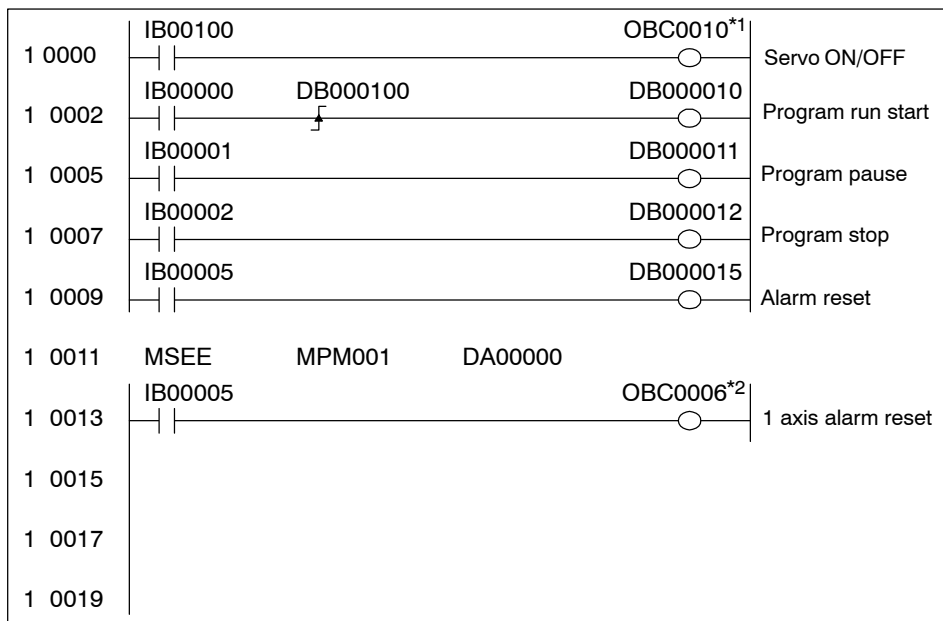
Set the system work number to be used for execution of the motion program in the fourth word of the MSEE work register.

Setting Range: 1 to 16

The system work number that has been set is valid only when Bit D (System Work Number Setting) of the control signal for the motion program is ON. If this work number is out of the setting range or the system work for the number that was set is already being used, the status flag Bit E “Error: No system work” turns ON.

### ■ Example of Ladder Program for Motion-program Control

The following examples show the minimum amount of code that is required in a ladder logic program to control a motion program.



\* 1. OB80000 for MP2000-series Machine Controllers

\* 2. OB8000F for MP2000-series Machine Controllers

The contents of the above ladder program are as follows.

Step No.	Program Contents
1 to 7	The signals connected to the MP-series Machine Controller external input signals are stored as the motion program control signal IW0000 (external input signal)→DW00001 (2nd word of MSEE work register) Program run start Program pause Program stop Alarm reset
8	Calls the motion program MPM001 MSEE <u>MPM001</u> <u>DA00000</u> 1          2 1: Motion program number 2: MSEE work register address
11 to 15	The alarm reset signal (IB00005) sets the operation mode of the setting parameter for each axis and resets the alarm to clear the axis alarm.

If input signals (IB00000 to IB00007) are sent with the ladder logic program shown here to DW00001 (second word of the MSEE work register) as control signals for the motion program from an external device connected to a machine controller in the MP series, motion program operations such as run start, stop, and pause can be done with the motion-management functions of the system.

**EXAMPLE**

The following table shows an example of the external input signals required to create the minimum amount of code that is required in a ladder logic program for running motion programs on the MP-series Machine Controller.

**Table 1.8 External Input Signals and Motion Program Control Signals**

External Signal Address	External Signal Name	BIT	Motion Program Control Signal
IB00000:	Program run start	B0:	Program run start request
IB00001:	Program pause	B1:	Program pause request
IB00002:	Program stop	B2:	Program stop request
IB00003:	Program debugging mode	B3:	Program debugging mode selection
IB00004:	Program debugging start	B4:	Program debugging start request
IB00005:	Alarm reset	B5:	Alarm reset request

### 1.1.4 Parallel Program Execution

In the MP-series Machine Controller, complex control of Modules can be performed by parallel execution, thereby allowing flexible programming for various machine operations. The following two methods are provided for the parallel execution of motion programming:

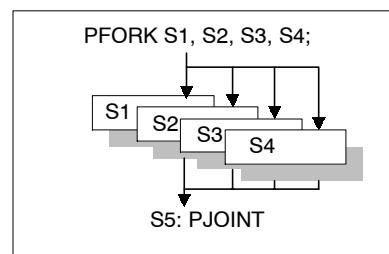
#### ■ Parallel Execution With PFORK Command

Four program blocks can be executed in parallel in one motion program with a PFORK command.

##### Parallel Execution 1

- Four program blocks can be executed in parallel in one motion program.
- Up to four program blocks can be executed in parallel with the PFORK command.

Motion Program MPM001

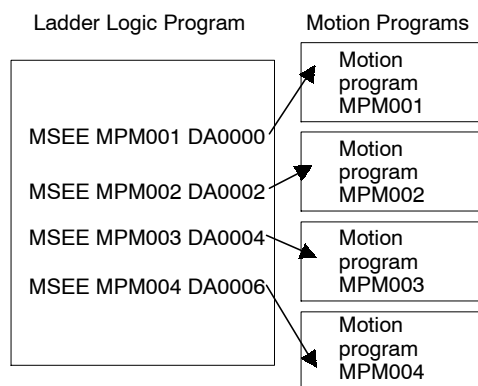


#### ■ Parallel Execution With Ladder Logic MSEE Command

Multiple motion programs can be executed in parallel with a ladder logic MSEE command. When programs are generated automatically in the Group Definition window of the MPE720, four programs can be executed in parallel.

##### Parallel Execution 2

- Multiple motion programs can be executed in parallel.
- Multiple motion programs can be executed in parallel using the ladder logic MSEE command.



### 1.1.5 Editing Programs

Motion programs are edited on the Motion Editor window of MPE720. The Motion Editor window has the following functions:

- It has the same edit functions as a standard text editor, such as cut & paste, search, replace, and jump.
- It also provides special functions, such as debugging operations, program monitoring, and position teaching.
- It has an import function to read files created by a standard text editor as MP-series Machine Controller programs.
- It has an export function for creating text files from MP-series Machine Controller motion program files.

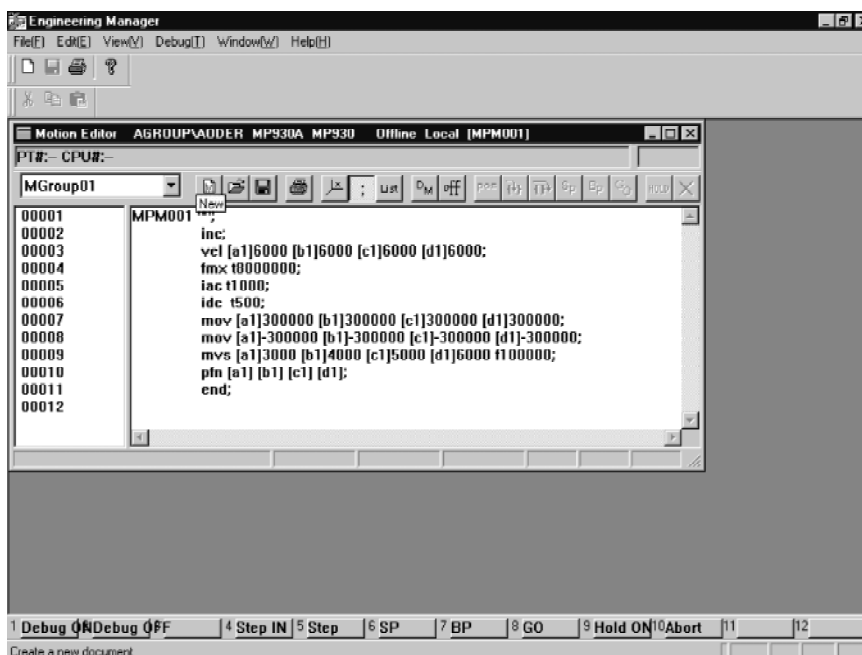


Figure 1.3 MPE720 Motion Editor Window



For details of the edit functions for motion programs, refer to the *MPE720 Software for Programming Device User's Manual (SIEPC88070005)*.

## 1.2 Programming Methods

This section describes the basic rules for creating motion programs. Be sure to read this section through before starting to write a program.

### 1.2.1 Programming Format

#### ■ Sample Motion Programs

Table 1.9 Sample Program

Line No.	Program	Remarks
00001	MPM001 "sample"	Program number and comment
00002	FMX T1000000;	Maximum interpolation feed speed setting
00003	IAC T100;	Interpolation feed acceleration time setting
00004	IDC T100;	Interpolation feed deceleration time setting
00005	VEL [X1] 10000 [Y1] 2000 [Z1] 3000;	Feed speed setting
00006	INC;	Incremental mode setting
00007	MOV [X1] 100. [Y1] 150. [Z1] 200.;	Positioning
00008	MVS [X1] 100. [Y1] 50. F500000;	Linear interpolation
00009	IOW IW0011 = = 1;	
00010	MW0100 = (MW0110*100+50)/100;	
00011	MW0200 = (MW0210*100+50)/100;	
00012	ABS;	
00013	MOV [X1] MW0100 [Y1] MW0200;	
00014	POS [X1] 0 [Y1] 0	
00015	PFORK LA01 LA02 LA03 LA04;	PARALLEL FORK command
00016	LA01: INC;	Label
00017	MOV [X1] 1000.;	
00018	JOINTO LA05;	
00019	LA02: INC;	
00020	MOV [Y1] 2000.;	
00021	JOINTO LA05;	
00022	LA03: ABS;	
00023	MVS [Z1] 1500 F500000;	
00024	MW1000 = 12345;	
00025	JOINTO LA05;	
00026	LA04: MW1100 = 1000;	
00027	IOW IB101 = = 1;	
00028	JOINTO LA05;	
00029	LA05: PJOINT	End of parallel execution
00030	END;	Program end

## ■ Programming Format

The variable-length programming format is as shown in *Table 1.10*.

**Table 1.10 Variable-Length Programming Format**

Item	Programming Format
<b>Program Number</b>	MPM□□□      □□□ = 1 to 256
<b>Labels</b>	Up to eight characters
<b>Motion Commands</b>	Three letters (Some commands have more or less than three letters.)
<b>Coordinate Words</b>	[abcd] ± 123.456 A   B    C A: Axis name B: Sign (+, -) designation possible C: Coordinate value (For details, see 1.2.2 <i>Controlled Axes</i> .)
<b>Interpolation Feed Speeds</b>	F3000000 Changes according to the setting of the number of digits after the decimal point (fixed parameter). Number of digits after the decimal point = 3: 3000.000 mm/min Number of digits after the decimal point = 5: 30.00000 mm/min
<b>Dwell Times</b>	TIM T1000      Unit: 10 msec (no decimal point)
<b>Subroutine Numbers</b>	MPS□□□      □□□ = 1 to 256
<b>P Designations</b>	P100;            Interpolation feed speed ratio setting: 1 to 100
<b>End of Block</b>	;

## ■ Leading Zeroes

Leading zeroes can be omitted from numeric values following characters, including program numbers and register (variable) numbers.

◀EXAMPLE▶

[X1] 00123            ⇒ [X1] 123  
 [X1] MW00010        ⇒ [X1] MW10  
 MPS002              ⇒ MPS2

## ■ Signs

Of the numeric value signs, positive (+) signs can be omitted, but negative (-) signs cannot.

◀EXAMPLE▶

[X1] +123            ⇒ [X1] 123  
 [X1] -123            ⇒ [X1] -123



A decimal point cannot be used in an interpolation feed speed (Fxxxx) references. F30000.000 cannot be designated. Designate F30000000. (Number of digits after the decimal point = 3)

## ■ Usable Characters

Table 1.11 shows the characters that can be used, together with their meanings.

**Table 1.11 Usable Characters**

Character	Meaning	Character	Meaning
<b>C</b>	C register	<b>P</b>	Interpolation feed speed override
<b>D</b>	D register	<b>R</b>	Radius of circle
	External positioning travel distance	<b>SS</b>	Skip signal number
<b>I</b>	I register	<b>T</b>	Timer value, number of circle turns FMX, IAC, IDC
<b>M</b>	M register		
<b>O</b>	O register	<b>U</b>	Circle center point coordinate 1 (Axis X)
<b>S</b>	S register	<b>V</b>	Circle center point coordinate 2 (Axis Y)
<b>F</b>	Interpolation feed speed	<b>MPS</b>	Subroutine number

## ■ Function Characters

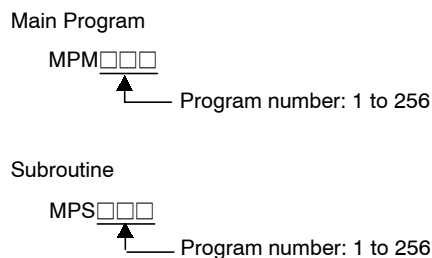
Table 1.12 shows the function characters that can be used, together with their meanings.

**Table 1.12 Function Characters**

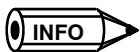
Character	Meaning	Character	Meaning
<b>SP</b>	Space (delimiter)	<b>&amp;</b>	Operator
<b>TAB</b>	Tab (delimiter)	<b>!</b>	Operator
<b>;</b>	End of block	<b>=</b>	Operator
<b>[Enter]</b>	Line feed	<b>( )</b>	Operator
<b>0 to 9</b>	Number	<b>= =</b>	Operator
<b>A to Z</b>	Alphabetic character	<b>&gt;</b>	Operator
<b>.</b>	Decimal point	<b>&lt;</b>	Operator
<b>+</b>	Operator	<b>&lt; &gt;</b>	Operator
<b>-</b>	Operator	<b>&gt; =</b>	Operator
<b>*</b>	Operator	<b>&lt; =</b>	Operator
<b>/</b>	Operator	<b>S { }</b>	Operator
<b> </b>	Operator	<b>R { }</b>	Operator
<b>^</b>	Operator		

## ■ Handling of Program Numbers

Program numbers are used to identify programs. There are two types of program number: Main program numbers and subroutine numbers. From 1 to 256 numbers can be attached to the programs or subroutines.



**Figure 1.4 Motion Program File Names**



1. The same program number cannot be designated for both a main program and a subroutine.
2. A total of up to 256 main programs and subroutines can be created.

## ■ Coding Comments

Comments can be coded in programs for use in motion program maintenance. The following two coding methods are provided:

- The comment statement is enclosed in double quotation marks.

“ Character string ”

### ◀EXAMPLE▶

ZRN [*axis1*] 0 [*axis2*] 0 [*axis3*] 0; “Zero point return of all axes”

MVS [*axis1*] 100.0 [*axis2*] 200. [*axis3*] 300.0; “3-axis linear interpolation”

- If the comment is not enclosed in double quotation marks, all the remaining characters on the same line as the double quotation marks (until Enter is pressed) will be treated as a comment.

“ Character string [Enter]

### ◀EXAMPLE▶

“After zero point return for all axes, move to queue position by linear interpolation

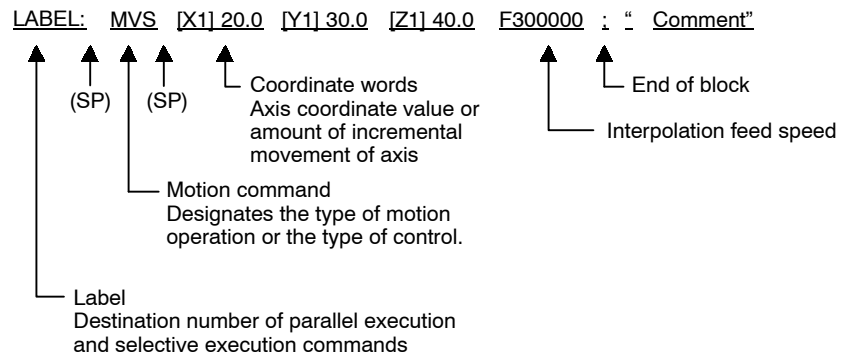
ZRN [*axis1*] 0 [*axis2*] 0 [*axis3*] 0;

MVS [*axis1*] 100.0 [*axis2*] 200. [*axis3*] 300.0;



## ■ Creating a One-block Command

1. Create a one-block command according to the programming format. A typical one-block example is shown in the following illustration.



2. At least one space [SP] must be inserted between the motion command and the coordinate words.
3. There is no limit on the number of characters in one line of one block. However, it is recommended that the number of characters be limited to the range that can be displayed on the screen, so that the program can be easily seen.
4. Insert CR or LF after a semi-colon at the end of a block, and press Enter.
5. The label serves as the destination number of the PARALLEL FORK command (PFORK) and the SELECTIVE FORK command (SFORK).

## ■ Labels

Labels must be provided as the destination numbers of the PARALLEL FORK command (PFORK) and the SELECTIVE FORK command (SFORK). Insert a colon [:] at the end of a character string containing from one to eight alphanumeric characters or symbols to create a label.

The following table shows the characters that can be used in labels.

Character	Numbers	0 to 9
	Letters	A to Z, a to z
	Symbols	%, ¥, @, -, _

### ◀EXAMPLE▶

```

PFORK LAB1 LAB2;

LAB1:  ZRN [axis1] 0 [axis2] 0 [axis3] 0;
        JOINTO LAB3;

LAB2:  MVS [axis1] 100.0 [axis2] 200. [axis3] 300.0;
        JOINTO LAB3;

LAB3:  PJOINT;
  
```

**IMPORTANT**

1. If there is more than one identical label in a program, an error will result (“Duplicate labels are defined”).
2. If the number of PFORK branches and the number of labels are different, an error will result.
3. Use a space or a comma (,) to divide a label of a PFORK command. A space and a comma can be used together.  
(Example) PFORK LABEL1 LABEL2, LABEL3 LABEL4;

1

## 1.2.2 Controlled Axes

### ■ Axis Names

With the MP-series Machine Controller, any axis name can be set using a maximum of eight characters. The axis names are set on the Group Definition window of the MPE720. The characters that can be used in axis names and the default axis names are shown in the following table.

<b>Usable Characters</b>	0 to 9, A to Z, a to z
<b>Example of Axis Names</b>	[AXIS1] [X1] [CONV1]
<b>Default Axis Names</b>	With 4-axis designation [A1] [B1] [C1] [D1]  With 8-axis designation [A1] [B1] [C1] [D1] [E1] [F1] [G1] [H1]



1. An axis name must contain from one to eight alphanumeric characters, and must be enclosed in [ ].
2. The same axis name cannot be set for more than 1 axis.
3. If the axis names set on the Group Definition window are different from those designated in the motion program, an error will result (“Axis name error”).

### ■ Coordinate Words

In this manual, the combination of an axis name and the signed travel distance or coordinate value that follows it is called a “coordinate word.”

The meanings of the coordinate words used in this system are shown in the following table.

Designation Method for Coordinate Words		Meaning
<b>Axis Name</b>	[X1], [Y1] [AXIS]	Designation of the name of the axis to be moved
<b>Travel Distance or Coordinate Value</b>	Direct designation: 123.456 Indirect designation: MW0100	Coordinate value of the designated axis or incremental move distance

Designation Method for Coordinate Words		Meaning
Auxiliary Data for Circular or Helical Interpolation	R	Circular interpolation radius value (fixed to incremental value)
	U	Center point coordinates for circular interpolation (Axis X)
	V	Center point coordinates for circular interpolation (Axis Y)
External Positioning Travel Distance	D	Distance travelled from input of external signal (fixed to incremental value)

**IMPORTANT**

- The “long” data type is used when a travel distance or coordinate value is designated as a variable.  
**Example:** ML0100, DL0300
- With a decimal point value, include a 0 for each place below the decimal point.  
**Example:** 300.000... If the number of digits after the decimal point = 3, store 300000 in the variable.  
In I and O registers, the servo parameter area cannot be used as variables for travel distances and coordinate values.

## ■ Number of Simultaneously Controlled Axes

The number of axes that can be controlled simultaneously from the motion programs is as shown in *Table 1.13*.

**Table 1.13 Number of Simultaneously Controlled Axes**

Command	Number of Simultaneously Controlled Axes
<b>POSITIONING (MOV)</b>	Up to 14 axes *
<b>LINEAR INTERPOLATION (MVS)</b>	Up to 14 axes *
<b>CIRCULAR INTERPOLATION (MCW/MCC)</b>	Two axes
<b>HELICAL INTERPOLATION (MCW/MCC)</b>	Three axes
<b>SKIP (SKP)</b>	Up to 14 axes *
<b>EXTERNAL POSITIONING (EXM)</b>	One axis
<b>SET TIME POSITIONING (MVT)</b>	Up to 14 axes *

\* Number of axes with simultaneous control for the MP930.

The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to *1.1.2 Function Performance* for details.

## ■ Reference Units

In the MP-series Machine Controller, the programmable reference unit can be set in the parameter for each axis.

**Table 1.14 Reference Units**

Parameter Setting Value	Reference Unit: pulse	Reference Unit: mm	Reference Unit: degree	Reference Unit: inch
Number of digits after the decimal point = 0	1 pulse	1 mm	1°	1"
Number of digits after the decimal point = 1	1 pulse	0.1 mm	0.1°	0.1"
Number of digits after the decimal point = 2	1 pulse	0.01 mm	0.01°	0.01"
Number of digits after the decimal point = 3	1 pulse	0.001 mm	0.001°	0.001"
Number of digits after the decimal point = 4	1 pulse	0.0001 mm	0.0001°	0.0001"
Number of digits after the decimal point = 5	1 pulse	0.00001 mm	0.00001°	0.00001"



If the reference unit is the pulse, the number of digits after the decimal point will be invalid. Likewise, the decimal point in the motion program input and the position monitor display have no significance.

## ■ Maximum Programmable Values

The maximum values for a single move command are as shown in the following table.

	Number of Digits After Decimal Point	Reference Unit: pulse	Reference Unit: mm	Reference Unit: degree	Reference Unit: inch
<b>Finite Length</b>	<b>0</b>	-2147483648 to 2147483647	-2147483648 to 2147483647	0 to 35999999	-2147483648 to 2147483647
	<b>1</b>	-2147483648 to 2147483647	-214748364.8 to 214748364.7	0 to 3599999.9	-214748364.8 to 214748364.7
	<b>2</b>	-2147483648 to 2147483647	-21474836.48 to 21474836.47	0 to 359999.99	-21474836.48 to 21474836.47
	<b>3</b>	-2147483648 to 2147483647	-2147483.648 to 2147483.647	0 to 35999.999	-2147483.648 to 2147483.647
	<b>4</b>	-2147483648 to 2147483647	-214748.3648 to 214748.3647	0 to 3599.9999	-214748.3648 to 214748.3647
	<b>5</b>	-2147483648 to 2147483647	-21474.83648 to 21474.83647	0 to 359.99999	-21474.83648 to 21474.83647

**Note** When reference unit = pulse

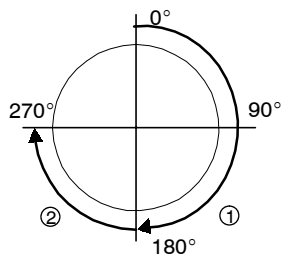
## ■ Designating Absolute Mode for Infinite-Length Axes

With an absolute reference (mode for designating in a range of 0 to 359.999°) for an infinite-length axis, the designated sign indicates the direction of rotation and the designated angle indicates the absolute position, as shown in the following illustration.

### ◀EXAMPLE▶

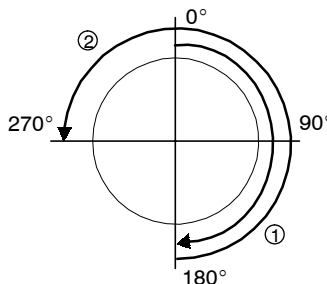
- In this example, the position is specified from a current value of 180°.

**Designation Example 1:** Absolute programming mode for an infinite-length axis

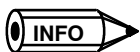


```
ZRN [X1] 0;
① INC MOV [X1] 180.0;
② ABS MOV [X1] 270.0; Moved clockwise
(in the forward direction) to 270°
```

**Designation example 2:** Absolute programming mode for an infinite-length axis



```
ZRN [X1] 0;
① INC MOV [X1] 180.0;
② ABS MOV [X1] -270.0; Moved counterclockwise
(in the reverse direction) to 270°
```



In the designation of the absolute programming mode for an infinite-length axis, a counterclockwise designation cannot result in +0.0 with a movement to the 0° position. In this case, be sure to designate +360.0.

■ **Coding Numbers and Variables**

The numbers that can be used in motion programs are classified into constants and variables. The method of setting numbers is shown in *Table 1.15*.

**Constants**

**Table 1.15** Specifiable Numeric Notation

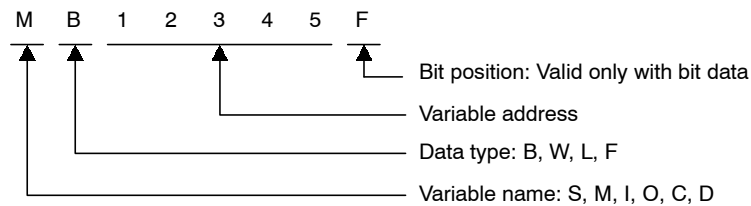
Classification	Range	Coding Example
<b>Decimal Integers</b>	-2147483648 to 2147483647	0, 734, +823, -2493
<b>Decimal Fractions</b>	-214748.364 to 2147483.647 Changes according to the setting of the number of digits after the decimal point.	763., +824.2, -234.56 -321.12345
<b>Hexadecimal Integers</b>	0 to FFFFFFFFH	FFFABCDEH, 2345H, FH
<b>Real Numbers</b>	±(1.175E-38 to 3.402E+38), 0	123.45, -765.4321

## Variables

The variables that can be used in motion programs are listed in *Table 1.16*. Use them as required.

**Table 1.16 Types of Variable and Notation Method**

Classification	Variable Type	Data Type			
		Bit	Word	Long	Floating Point
Global Variables	S Registers	SB	SW	SL	SF
	M Registers	MB	MW	ML	MF
	I Registers	IB	IW	IL	IF
	O Registers	OB	OW	OL	OF
	C Registers	CB	CW	CL	CF
Local Variables	D Registers	DB	DW	DL	DF



### EXAMPLE

```
MB001001 = 1;
MW00100 = 1234;
ML00100 = 12345678;
MF00100 = 1234.5678
```

## Math Operations and Functions

Global variables, local variables, and constants can be used to perform ordinary math operations in combination with operators and functions, and the results can be substituted in variables. The commands listed in the following table are provided for the math operations and functions.

Classification	Command	Name	Programming Format
Arithmetic Operations	=	SUBSTITUTE	MW- =MW-;
	+	ADD	MW- =MW- +MW-;
	-	SUBTRACT	MW- =MW- -MW-;
	*	MULTIPLY	MW- =MW- *MW-;
	/	DIVIDE	MW- =MW- /MW-;
	MOD	REMAINDER	MW- =MOD;

1

Classification	Command	Name	Programming Format
<b>Logic Operations</b>		OR (logical OR)	MB- =MB-  MB-;
	^	XOR (exclusive logical OR)	MB- =MB- ^MB-;
	&	AND (logical AND)	MB- =MB- &MB-;
	!	NOT (inversion)	MB- =MB- !MB-;
<b>Data Comparison</b>	= =	MATCH	IF MW- = =MW-;
	< >	MISMATCH	IF MW- < >MW-;
	>	GREATER THAN	IF MW- >MW-;
	<	LESS THAN	IF MW- <MW-;
	> =	GREATER THAN OR EQUAL TO	IF MW- > =MW-;
	< =	LESS THAN OR EQUAL TO	IF MW- < =MW-;
<b>Data Operations</b>	<b>SFR</b>	SHIFT RIGHT	SFR MB-N-W-;
	<b>SFL</b>	SHIFT LEFT	SFL MB-N-W-;
	<b>BLK</b>	MOVE BLOCK	BLK MW-MW-W-;
	<b>CLR</b>	CLEAR	CLR MB-W-;
<b>Basic Functions</b>	<b>SIN</b>	SINE	SIN(MW-);
	<b>COS</b>	COSINE	COS(MW-);
	<b>TAN</b>	TANGENT	TAN(MF-);
	<b>ASN</b>	ARC SINE	ASN(MF-);
	<b>ACS</b>	ARC COSINE	ACS(MF-);
	<b>ATN</b>	ARC TANGENT	ATN(MW-);
	<b>SQT</b>	SQUARE ROOT	SQT(MW-);
	<b>BIN</b>	BCD-TO-BINARY	BIN(MW-);
	<b>BCD</b>	BINARY-TO-BCD	BCD(MW-);
	<b>S{ }</b>	SET BIT	S{MB-}=MB- &MB-;
	<b>R{ }</b>	RESET BIT	R{MB-}=MB- &MB-;



## 1.2.3 Feed Speeds

### ■ Rapid Traverse Speed

The rapid traverse speed is used with the following axis movements:

- POSITIONING (MOV)
- JOG (JOG) operations
- STEP (STEP) operations

Set the rapid traverse speed values for each axis in parameter OLxx22/OLxx10 (Rapid Traverse Speed), or set SET VELOCITY (VEL) in the motion program.

- Rapid Traverse Speed Parameter

Machine Controller	Parameter Number	Name	Setting Range	Unit
MP900-series	OLxx22	Rapid Traverse Speed	0 to $2^{31}-1$	According to the reference unit
MP2000-series	OLxx10	Speed Reference Setting	0 to $2^{31}-1$	According to the reference unit

- Setting the Feed Speed in a Motion Program

- |   |
|---|
| <ul style="list-style-type: none"> <li>• Direct Designation in the Rapid Traverse Speed Parameter<br/>           OLC022 = 6000; Axis 1 Rapid Traverse Speed (OL8010)<br/>           OLC062 = 5000; Axis 2 Rapid Traverse Speed (OL8090)<br/>           OLC0A2 = 7000; Axis 3 Rapid Traverse Speed (OL8110)</li> </ul> |
| <ul style="list-style-type: none"> <li>• Setting with SET VELOCITY (VEL)<br/>           VEL [X1] 6000 [Y2] 5000 [Z1] 7000;</li> </ul>   |

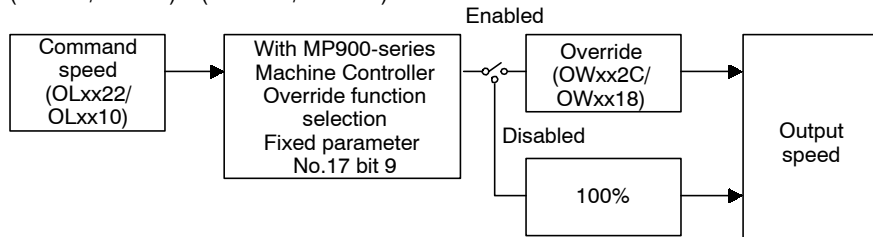
An override in the range of 0% to 327.67% can be selected for the Rapid Traverse Speed. Set Override in the setting parameter for each axis.

Machine Controller	Parameter Number	Setting Range	Unit
MP900-series	OWxx2C	0 to 32767	1 = 0.01%
MP2000-series	OWxx18	0 to 32767	1 = 0.01%

There are three override setting methods: Motion program, ladder logic program, or the Parameter Setting Screen.

$$\text{Command speed} \times \text{override} = \text{output speed}$$

(OLxx22/OLxx10) (OWxx2C/OWxx18)



**Note** With the 2000-series Machine Controller, the Override value is always effective.



1. The Override value is always effective during operation. It can be changed in the ladder logic program, motion program, and parameter settings while an axis is travelling.
2. If the output speed is out of range after adjustment for the override setting, a parameter setting error will result.

## ■ Interpolation Feed Speed

- The feed speeds for interpolation commands are designated by numbers following the letter F. They are sometimes referred to as F designations.
- The F designation for linear and circular interpolation designates the tangential speed.

### ◀ EXAMPLE ▶

INC MVS [X]1200 [Y]900 F500;

$$F = 500 = \sqrt{400^2 + 300^2} \text{ [reference unit/min]}$$

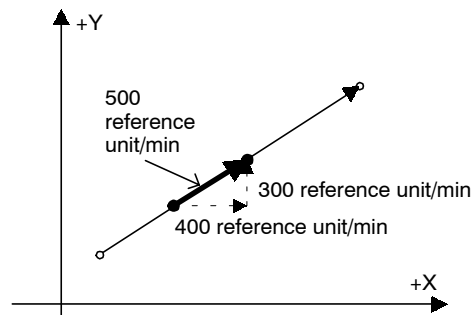


Figure 1.5 Tangential Speed for 2-axis Linear Interpolation



#### ◆ Override

Override often means “to invalidate.” In this manual, however, it should be taken to mean “change” the set value.

MCC [X]— [Y]— I— J— F200;  
 $F = 200 = \sqrt{V_x^2 + V_y^2}$  [reference unit/min]

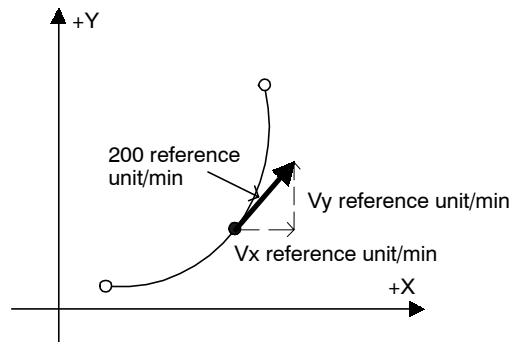


Figure 1.6 Tangential Speed for Circular Interpolation

◀EXAMPLE▶

INC MVS [X] 100 [Y] 100 [Z] 100 F500;  
 $F = 400 = \sqrt{V_x^2 + V_y^2 + V_z^2}$  [reference unit/min]

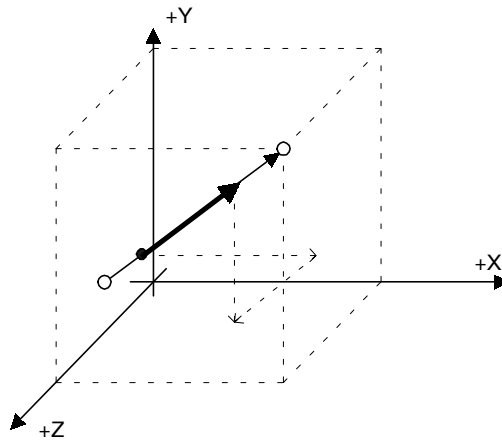
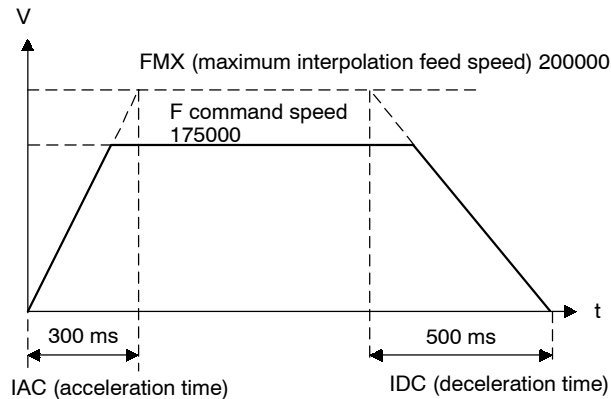


Figure 1.7 Tangential Speed for 2-axis Linear Interpolation

INC MVS [X]— [Y]— [Z]— [S]— F600;  
 $F = 600 = \sqrt{V_x^2 + V_y^2 + V_z^2 + V_s^2}$  [reference unit/min]

Figure 1.8 Tangential Speed for 4-axis Linear Interpolation

- The upper limit of the feed speed is controlled by the capacity of the machine system and the servo system. The upper limit of this feed speed is set in the following motion command.



```
(Programming example)
FMX T200000;
IAC T300;
IDC T500;
MVS [X] 200, [Y] 250, F175000;
```

**Figure 1.9 Setting Command for Maximum Interpolation Feed Speed**

If an F designation is made that exceeds the maximum interpolation feed speed, an alarm will be generated.

**IMPORTANT**

If an interpolation command is used in the motion program, the FMX command must be executed at the beginning of the program.

- F Designation Unit

A decimal point cannot be used in an F designation.

**EXAMPLE**

F2000000

- An override in the range of 0% to 327.67% can be selected for the interpolation feed speed.

The following registers are used for the Override setting.

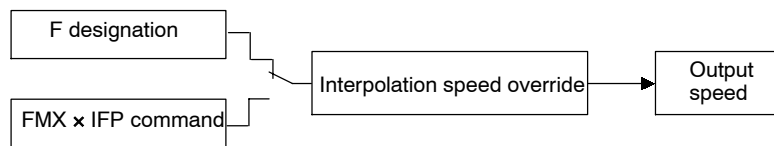
- For MP900-series Machine Controllers

The register set in the Group Definition window of the MPE720 (Default = MW00001)

- For MP2000-series Machine Controllers  
The register of the third word of MSEE command work register  
However, the validity of this interpolation override depends on the setting of Bit14 (Override Selection Enabled) in the register of the second word of the MSEE command work register: Override Selection Enabled to OFF: Override fixed at 100%  
Override Selection Enabled to ON: Override set in the register of the third word of the MSEE command work register

$$\begin{array}{l} \text{F designation} \\ \text{FMX} \times \text{IFP command} \end{array} \times \text{Interpolation speed override} = \text{Output speed}$$

(100 % = 10000)



The motion commands relating to the interpolation feed speed are shown below.

- F designation F designation in interpolation command
- IFP command INTERPOLATION FEED SPEED RATIO SETTING
- FMX command MAXIMUM INTERPOLATION FEED SPEED SETTING
- IAC command INTERPOLATION ACCELERATION TIME CHANGE
- IDC command INTERPOLATION DECELERATION TIME CHANGE
- SCC command S-CURVE TIME CONSTANT CHANGE



1. The Override setting is always effective during operation.
2. When the output speed is out of range after adjustment the override setting data, it will be clamped at the FMX value.

#### IMPORTANT

1. If a rotary axis is included as an axis for an interpolation command, the mechanical travel speed will not be the tangential speed for the F designation.
2. If “FO” is specified in the F designation, a program error will result.
3. Do not specify an F designation with a minus sign (F-□□□). Doing so will cause an alarm to be generated.

## 1.2.4 Motion Commands

Refer to *Appendix A Motion Command List* for the motion commands.

# 2

## Motion Commands

This chapter describes programming axis move commands and control commands, which are two types of motion commands.

<b>2.1 Axis Move Commands</b> .....	<b>2 -2</b>
2.1.1 POSITIONING (MOV) .....	2 -2
2.1.2 LINEAR INTERPOLATION (MVS) .....	2 -5
2.1.3 CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION (MCW, MCC) .....	2 -9
2.1.4 CLOCKWISE/COUNTERCLOCKWISE HELICAL INTERPOLATION (MCW, MCC) .....	2 -15
2.1.5 ZERO POINT RETURN (ZRN) .....	2 -18
2.1.6 SKIP FUNCTION (SKP) .....	2 -23
2.1.7 SET TIME POSITIONING (MVT) .....	2 -24
2.1.8 EXTERNAL POSITIONING (EXM) .....	2 -27
<b>2.2 Control Commands</b> .....	<b>2 -28</b>
2.2.1 ABSOLUTE MODE (ABS) .....	2 -28
2.2.2 INCREMENTAL MODE (INC) .....	2 -30
2.2.3 CURRENT POSITION SET (POS) .....	2 -31
2.2.4 COORDINATE PLANE SETTING (PLN) .....	2 -33
2.2.5 MOVE ON MACHINE COORDINATES (MVM) ...	2 -34
2.2.6 PROGRAM CURRENT POSITION UPDATE (PLD) .....	2 -35
2.2.7 DWELL TIME (TIM) .....	2 -36
2.2.8 PROGRAM END (END) .....	2 -37

## 2.1 Axis Move Commands

This section describes the methods of designating axis move commands and provides some programming examples.

### 2.1.1 POSITIONING (MOV)



## Caution

- The path of movement with the POSITIONING (MOV) command is not always a straight line. When programming, be sure to check the path to make sure that there are no tools or other obstacles in the way of the workpiece.  
Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

#### ■ Overview

The POSITIONING (MOV) command independently moves each axis from the current position to the end position at rapid traverse speed (the speed set in each axis parameter). Up to 14 axes\* can be moved simultaneously. Any axis not specified in the command will not be moved.

The path of movement with the MOV command is different from the linear travel described in 2.1.2 LINEAR INTERPOLATION (MVS).

\* Number of axes with simultaneous control for the MP930.

The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to 1.1.2 Function Performance for details.

#### ■ Description

The MOV command is designated as follows:

MOV [axis1] - [axis2] - ...;  
Reference position

The following illustration shows the path of movement with the MOV command.

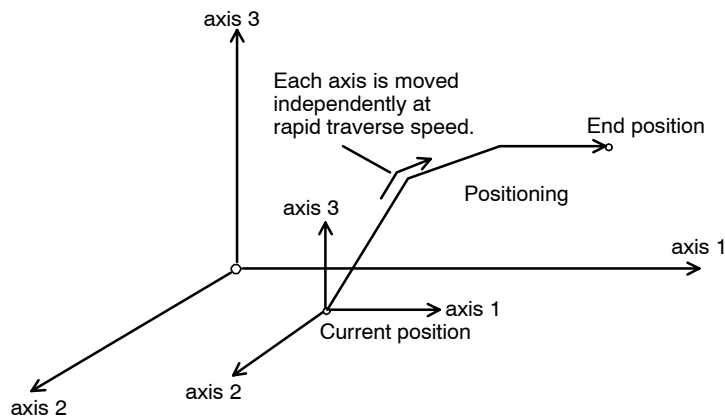


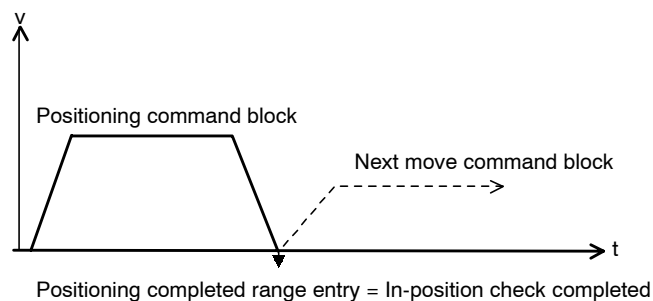
Figure 2.1 Path of Movement with MOV

- The reference position is set as follows, depending on the ABS or INC mode designated in advance:

**Absolute (ABS) mode:** A mode where the coordinate language of a command to move axes is regarded as an absolute value on the work coordinate.

**Increment (INC) mode:** A mode where the coordinate language of a command to move axes is regarded as an incremental value from the current position on the work coordinate.

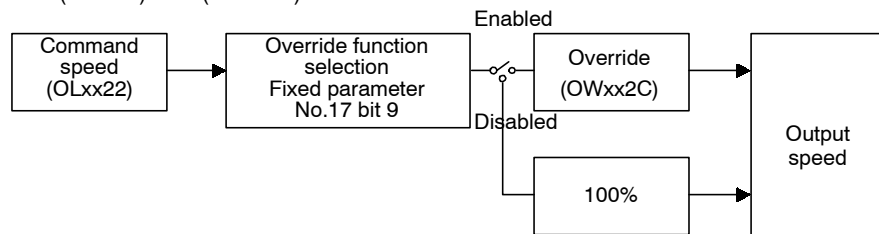
- An in-position check is performed for the axis movement with the MOV command to check that the positioning completed range has been entered. After the in-position check has been completed, the next move command block will be executed. The following illustration shows the in-position check operation.



**Figure 2.2 In-position Check Operation**

- The rapid traverse speed is the speed set in setting parameter 30 (Rapid Traverse Speeds (OLxx22)\*) for each axis. Overrides in the range of 0% to 327.67% can be selected for the rapid traverse speeds. Set the Overrides (OWxx2C) in setting parameter 35 for each axis.

$$\begin{array}{ccc} \text{Command speed} & \times & \text{Override} = \text{Output speed} \\ \text{(OLxx22)} & & \text{(OWxx2C)} \end{array}$$



\* The parameter used for the MP900-series Machine Controller.

Refer to 1.2.3 Feed Speeds for the parameter for the MP2000-series Machine Controller.

- Single-step linear acceleration/deceleration or S-curve acceleration/deceleration can be set in the parameter for automatic acceleration/deceleration control of movement using POSITIONING (MOV).



◆ ABS and INC commands

These commands designate whether a coordinate word is to be treated as an absolute value or an incremental value. They are modal group commands, so once one has been executed, it remains in effect until the other one is executed.



- The parameters and motion commands relating to automatic acceleration/deceleration for positioning are as follows:
- Parameters




	<b>MP900-series Machine Controller</b>	<b>MP2000-series Machine Controller</b>
<b>Linear Acceleration/Deceleration Time</b>	OWxx0C / OWxx0D	OWxx36 / OWxx38
<b>Filter Time Constant</b>	OWxx14	OWxx3A
<b>Motion Command Control Flags Filter Type Selection</b>	OWxx21 bit 4 to bit 7	OWxx03 bit 8 to bit 11

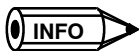
- Motion commands

ACCELERATION TIME CHANGE (ACC)  
 DECELERATION TIME CHANGE (DCC)  
 S-CURVE TIME CONSTANT CHANGE (SCC)

The acceleration/deceleration patterns can be set by combining the above parameters and motion commands.

The acceleration/deceleration pattern example of MP900-series Machine Controller is shown below.

<b>NO.</b>	<b>Automatic Acceleration/ Deceleration Type</b>	<b>Related Parameter Setting</b>	<b>Remarks</b>
<b>1</b>	No acceleration/deceleration	OWxx0C = 0 OWxx21 bit 4 to bit 7 = 0	
<b>2</b>	Single-step linear acceleration/deceleration	OWxx0C ≠ 0 OWxx21 bit 4 to bit 7 = 0	
<b>3</b>	S-curve acceleration/deceleration	OWxx0C ≠ 0 OWxx21 bit 4 to bit 7 = 2 OWxx14 = 0	



Asymmetric acceleration/deceleration cannot be designated for the MOV command.

## ■ Programming Examples

The following illustration shows a programming example for the MOV command in ABS mode.

### ◀ EXAMPLE ▶

```
ABS;
MOV [axis1] 4000 [axis2] 3000 [axis3] 2000;
Where
Current position: axis 1 = axis 2 = axis 3 = 0
```

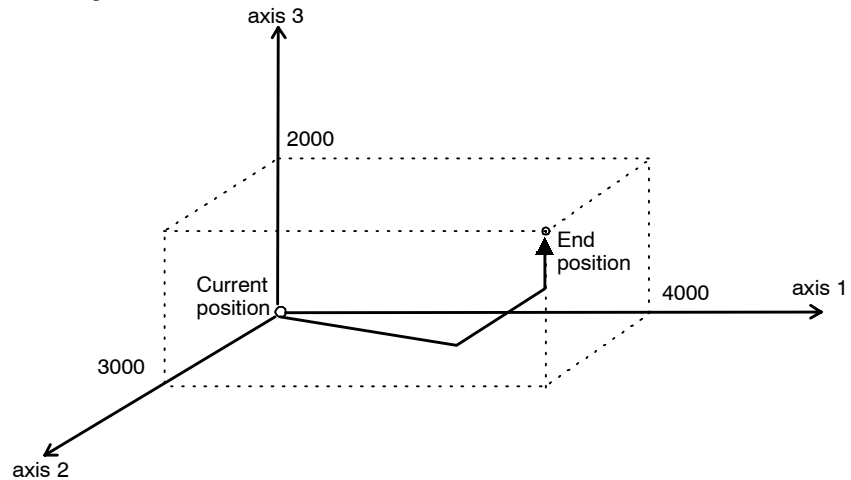


Figure 2.3 Programming Example for MOV

## 2.1.2 LINEAR INTERPOLATION (MVS)



### Caution

- LINEAR INTERPOLATION (MVS) can be executed for either linear axes or rotary axes. If rotary axes are included, however, the linear interpolation path will not be in straight line. When programming, be sure to check the path to make sure that there are no tools or other obstacles in the way of the workpiece.

Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

## ■ Overview

The LINEAR INTERPOLATION (MVS) command moves each axis linearly from the current position to the end position at interpolation feed speed. Up to 14 axes\* can be moved simultaneously. Any axis not specified in the command will not be moved.

\* Number of axes with simultaneous control for the MP930.

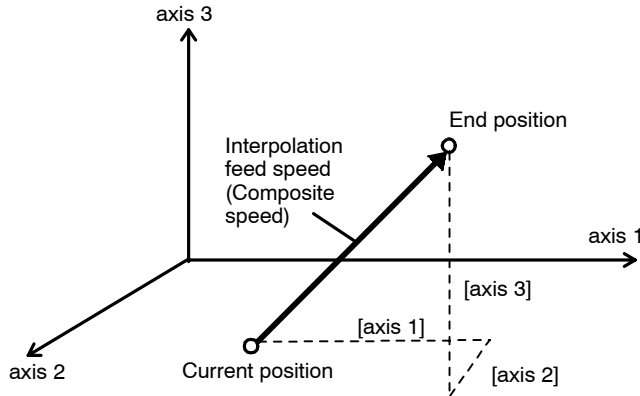
The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to 1.1.2 *Function Performance* for details.

## ■ Description

The MVS command is designated as follows:

```
MVS [axis1] - [axis2] - ... F=;
      Reference position  Interpolation feed speed
```

The following illustration shows the path of movement.



**Figure 2.4 Path of Movement with MVS**

- The reference position depends on the ABS or INC mode designated in advance.
- The interpolation feed speed is also referred to as the F designation. It is specified in the speed designation (F) or in the INTERPOLATION FEED SPEED RATIO SETTING command (IFP). The last F designation or IFP command specified in the previous block remains in effect. A LINEAR INTERPOLATION command for which no F designation or IFP command has been executed since the power was turned ON will generate an alarm.
- The composite speed of all the designated axes will be equivalent to the F designation. If an F designation exceeds the limit set in MAXIMUM INTERPOLATION FEED SPEED (FMX), an alarm will be generated.

- Two Axes (axis 1 and axis 2)

$$F = \sqrt{V_{axis1}^2 + V_{axis2}^2}$$

- Three Axes (axis 1, axis 2, and axis 3)

$$F = \sqrt{V_{axis1}^2 + V_{axis2}^2 + V_{axis3}^2}$$

- Four Axes (axis 1, axis 2, axis 3, and axis 4)

$$F = \sqrt{V_{axis1}^2 + V_{axis2}^2 + V_{axis3}^2 + V_{axis4}^2}$$

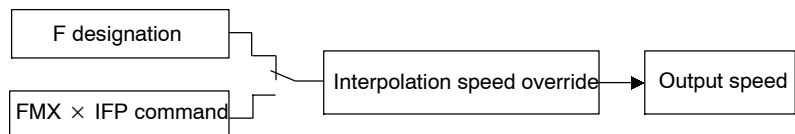
**IMPORTANT**

When creating a motion program that uses interpolation commands, designate MAXIMUM INTERPOLATION FEED SPEED (FMX) at the beginning of the program. If this is not designated, an alarm will be generated.

- In actual programming operations, an override of 0% to 327.67% can be applied to the F designation. The override will take effect immediately. Refer to 1.2.3 Feed Speeds for the override setting methods.

$$F \text{ designation} \times \text{Interpolation speed override} = \text{Output speed}$$

$FMX \times IFP \text{ command} \quad (100\% = 10000)$



- An in-position check is not executed for axis movement with LINEAR INTERPOLATION (MVS). The next block will be executed when pulse distribution for the designated block has been completed. To execute the next block after the in-position check, designate IN-POSITION CHECK (PFN) in the same block or the next block.
- The following types of acceleration/deceleration can be set for automatic acceleration/deceleration control using parameter settings and the IAC and IDC commands:
  - Single-step linear acceleration/deceleration
  - Asymmetric acceleration/deceleration
  - S-curve acceleration/deceleration
- The parameters and motion commands relating to automatic acceleration/deceleration of the interpolation feed speed are as follows:
- Parameters

	<b>MP900-series Machine Controller</b>	<b>MP2000-series Machine Controller</b>
<b>Filter Time Constant</b>	OWxx14	OWxx3A
<b>Motion Command Control Flags Filter Type Selection</b>	OWxx21 bit 4 to bit 7	OWxx03 bit 8 to bit 11

- Motion commands

MAXIMUM INTERPOLATION FEED SPEED SETTING (FMX)

F designation in interpolation command: INTERPOLATION FEED SPEED

INTERPOLATION FEED SPEED RATIO SETTING (IFP)

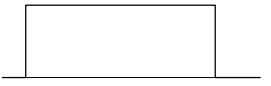
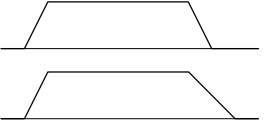
INTERPOLATION ACCELERATION TIME CHANGE (IAC)

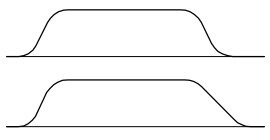
INTERPOLATION DECELERATION TIME CHANGE (DCC)

S-CURVE TIME CONSTANT CHANGE (SCC)

The acceleration/deceleration patterns can be set by combining the above parameters and motion commands.

The acceleration/deceleration pattern example of MP900-series Machine Controller is shown below.

<b>NO.</b>	<b>Automatic Acceleration/ Deceleration Type</b>	<b>Related Parameter and Command</b>	<b>Remarks</b>
<b>1</b>	No acceleration/deceleration	Interpolation acceleration time change (IAC) = 0  Interpolation deceleration time change (IDC) = 0  OWxx21 bit 4 to bit 7 = 0	
<b>2</b>	Single-step linear acceleration/ deceleration	Interpolation acceleration time change (IAC) ≠ 0  Interpolation deceleration time change (IDC) ≠ 0  OWxx21 bit 4 to bit 7 = 0	

NO.	Automatic Acceleration/Deceleration Type	Related Parameter and Command	Remarks
3	S-curve acceleration/deceleration	Interpolation acceleration time change (IAC) $\neq 0$ Interpolation deceleration time change (IDC) $\neq 0$ OWxx21 bit 4 to bit 7 = 2 OWxx14 $\neq 0$	

- The acceleration time and deceleration time for the automatic acceleration/deceleration control of movement with interpolation commands can be set in the IAC and IDC commands.

2

### ■ Programming Examples

The following illustration shows a programming example for the MVS command in ABS mode.

◀EXAMPLE▶

```

FMX T30000000;
MW00001 = 10000; (*)
ABS;
MVS [axis1] 4000 [axis2] 3000 [axis3] 2000 F1000;
Where
Current position: axis 1 = axis 2 = axis 3 = 0
    
```

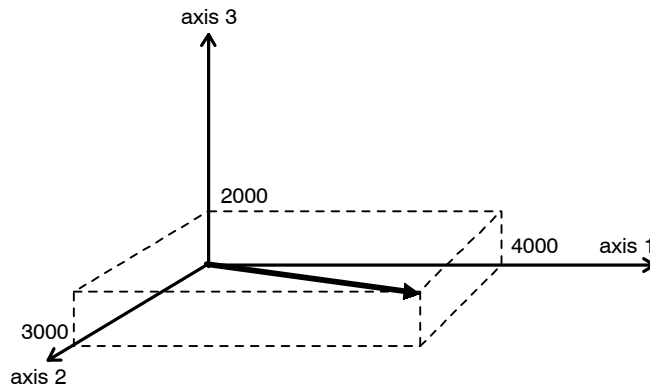
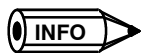


Figure 2.5 Programming Example for MVS

\* Example of MP930  
The interpolation override setting differs depending on the Machine Controller model. Refer to 1.2.3 *Feed Speeds* for details.



1. The speed designation (F) must be specified in the same block as the interpolation command.
2. The INTERPOLATION FEED SPEED RATIO SETTING (IFP) is an isolated command, and cannot be specified in the same block as the interpolation command.  
For details on the IFP command, see 3.2.5 *INTERPOLATION FEED SPEED RATIO SETTING (IFP)*.
3. If the override applied to the F designation results in a speed exceeding the maximum interpolation feed speed (FMX), the speed will be clamped to the FMX speed.

## 2.1.3 CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION (MCW, MCC)

### ■ Overview

The CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION (MCW, MCC) command is used to move two axes simultaneously from the current position to the end position on the designated plane at interpolation feed speed (F) on the circle determined by the center point position (U\_V\_) or the radius value (R).

### ■ Description

The CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION commands are designated as follows:

```
MCW [axis1] - [axis2] - U-V- T- F-;
           A           B C D
```

A: End position  
B: Center point position  
C: Number of turns  
D: Interpolation feed speed

or

```
MCC [axis1] - [axis2] - R- F-;
           A           B C
```

A: End point  
B: Radius value  
C: Interpolation feed speed

**Note** With center point position designation, multiple turns can be specified. The number of turns can also be omitted.

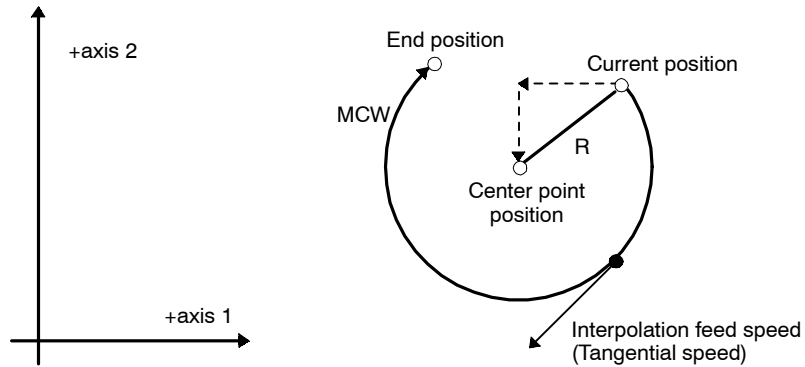
The direction of rotation for the CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION commands are as follows:

MCW: Clockwise (CW)  
MCC: Counterclockwise (CCW)

### IMPORTANT

1. Before designating a CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION command, be sure to designate the circular interpolation plane in COORDINATE PLANE SETTING (PLN). Be sure to designate the circular direction of rotation as MCW or MCC in the CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION command. Designate axes 1 and 2 for the end position and center point position in the same order as the axes are specified in the PLN command.
2. Designate the axes for the end position and center point position in the same order as the axes are specified in the PLN command.

The following illustration shows the method of designating the CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION commands.



**Figure 2.6 Programming Method for CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION (MCW, MCC)**

- The end position and center point positions are designated according to the ABS or INC mode that was last designated.

## ■ Programming Examples

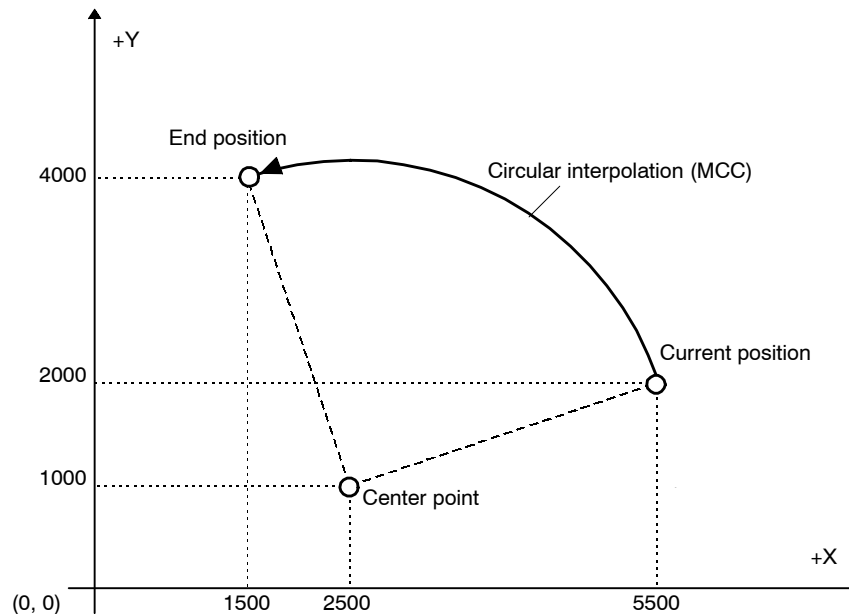
### Programming Example in ABS Mode

The following illustration shows a programming example for the MCC command in ABS mode.

**EXAMPLE**

```

ABS;
FMX T30000000;
MW 00001 = 10000; (*)
PLN [X] [Y] ;
MCC [X] 1500 [Y] 4000 U2500 V1000 F150;
    
```



**Figure 2.7 Programming Example in ABS Mode**

\* Example of MP930

The interpolation override setting differs depending on the Machine Controller model. Refer to 1.2.3 Feed Speeds for details.

### Programming Example in INC Mode

The following illustration shows a programming example for COUNTERCLOCKWISE CIRCULAR INTERPOLATION (MCC) in INC mode.

#### ◀EXAMPLE▶

```

INC;
FMX T30000000;
MW00001 = 10000; (*)
PLN [X] [Y] ;
MCC [X] -4000 [Y] 2000 U-3000 V-1000 F150;

```

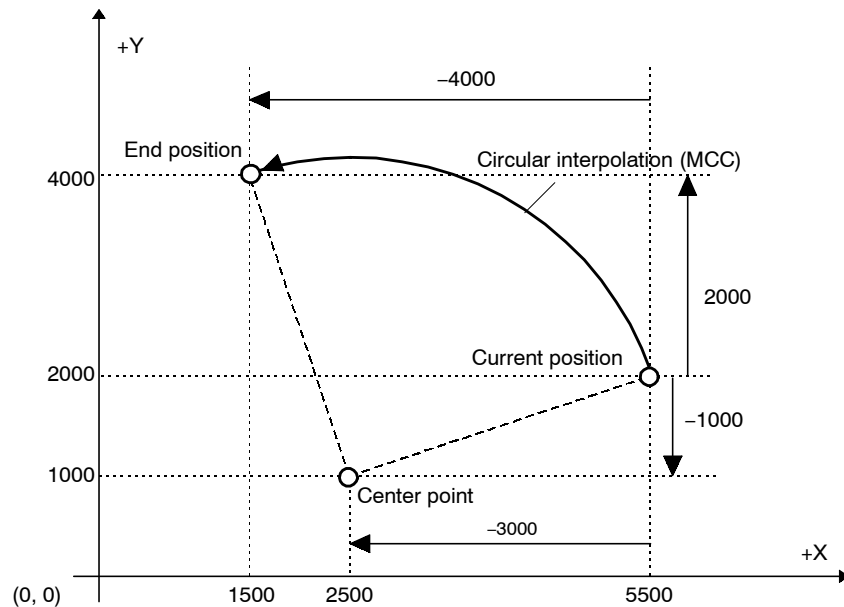


Figure 2.8 Programming Example in INC Mode

\* Example of MP930

The interpolation override setting differs depending on the Machine Controller model. Refer to 1.2.3 Feed Speeds for details.

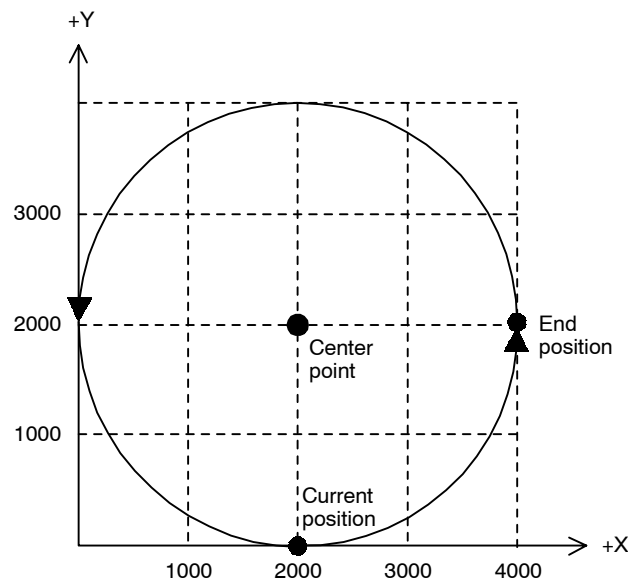


## Programming Example for Multiple Circles

The following illustration shows a programming example for multiple circles with the center point position designation.

### EXAMPLE

```
ABS;
FMX T30000000;
MW00001 = 10000; (*)
PLN [X] [Y] ;
MCC [X]4000 [Y]2000 U2000 V2000 T2 F150;
```



In the above example, this will be 2 1/4 circles.

\* Example of MP930

The interpolation override setting differs depending on the Machine Controller model. Refer to 1.2.3 *Feed Speeds* for details.




---

```
MCC [X]2000 [Y]0 U2000 V2000 T2 F150;
```

If current position = end position, as shown in the above command, there will be three circles.

---

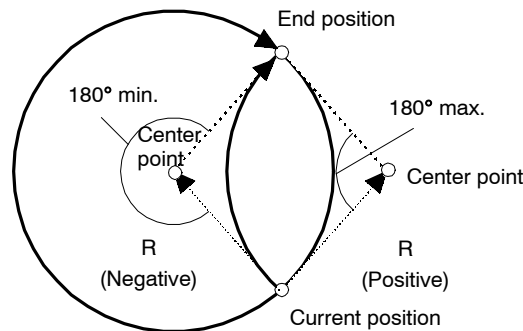
- A circle can be designated by the radius value R of the circle, instead of by the center point position. The circular interpolation in this case will be as shown in *Figure 2.9*.

MCW [axis 1] – [axis 2] – R-;

If  $R > 0$ : Circular interpolation with an arc angle of  $180^\circ$  or less

If  $R < 0$ : Circular interpolation with an arc angle of  $180^\circ$  or more

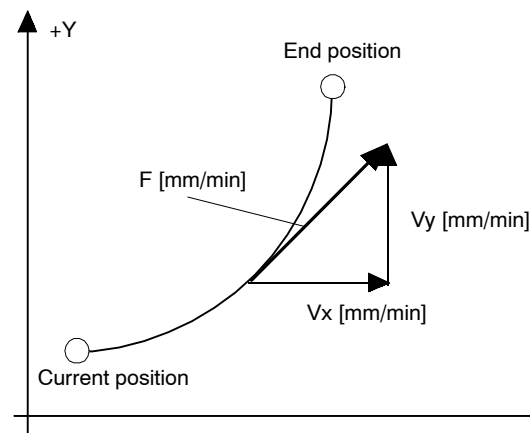
**Note** If  $R = 0$ : An alarm will be generated.



**Figure 2.9 CLOCKWISE CIRCULAR INTERPOLATION (MCC)**

- Before designating a CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION command, specify the plane in COORDINATE PLANE SETTING (PLN). If there is a previous designation, the plane that was last designated will remain in effect. If the plane is not specified at least once, an alarm will be generated.
- The interpolation feed speed is also referred to as the F designation. It is specified in the speed designation (F) or in the INTERPOLATION FEED SPEED RATIO SETTING command (IFP). The last F designation specified in the previous block remains in effect. An interpolation command for which no F designation has been specified will generate an alarm.
- The tangential speed of the designated circle will be equivalent to the F designation. If an F designation exceeds the limit set in MAXIMUM INTERPOLATION FEED SPEED (FMX), an alarm will be generated.

The following illustration shows the maximum interpolation feed speed.



**Figure 2.10 MAXIMUM INTERPOLATION FEED SPEED (FMX)**

- In actual programming operations, an override of 0% to 327.67% can be applied to the F designation. The override will take effect immediately.

**IMPORTANT**

When creating a motion program that uses interpolation commands, designate MAXIMUM INTERPOLATION FEED SPEED (FMX) at the beginning of the program. If this is not designated, an alarm will be generated.

- To execute the next block (whether or not it is a move command block) after the in-position check, designate the PFN command in the same block or the next block.
- The following types of acceleration/deceleration can be set for automatic acceleration/deceleration control using parameter settings and the IAC and IDC commands:
  - Single-step linear acceleration/deceleration
  - Asymmetric acceleration/deceleration
  - S-curve acceleration/deceleration
- With the center point position designation, a completely closed circle can be designated in one block by setting the same point for the starting point and the end point. Multiple circles can also be designated in one block using the number of turns designation (T-). A complete circle and multiple turns cannot be designated with the radius value R designation.



The methods of designating speed commands, accelerations/decelerations, overrides, and so on, are all the same as for linear interpolation. For details, see 2.1.2 LINEAR INTERPOLATION (MVS).

## ■ Programming Examples

The following illustration shows a programming example for a center point position designation in ABS mode.

**EXAMPLE**

```

ABS;
FMX T30000000;
MW00001 = 10000; (*)
MOV [X] 0 [Y] 0;
PLN [X] [Y] ;
MCW [X] 0 [Y] 0 U1000 V0;

```

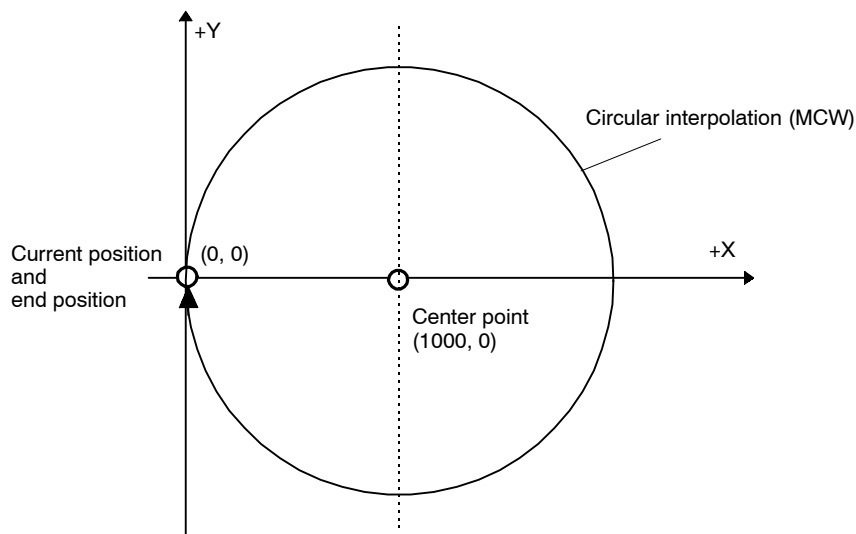


Figure 2.11 Programming Example for Center Point Position Designation

\* Example of MP930

The interpolation override setting differs depending on the Machine Controller model. Refer to 1.2.3 *Feed Speeds* for details.



1. The speed designation (F) must be specified in the same block as the interpolation command.
2. The INTERPOLATION FEED SPEED RATIO SETTING command (IFP) is an isolated command, and cannot be specified in the same block as the interpolation command.
3. If the override applied to the F designation results in a speed exceeding the maximum interpolation feed speed (FMX), the speed will be clamped to the FMX speed.

## 2.1.4 CLOCKWISE/COUNTERCLOCKWISE HELICAL INTERPOLATION (MCW, MCC)



### Caution

- The linear interpolation axis specified for a CLOCKWISE/COUNTERCLOCKWISE HELICAL INTERPOLATION (MCW, MCC) command can be either a linear axis or a rotary axis. Depending on the axis movement in the linear interpolation portion, the helical interpolation path may not be a helical shape. When programming, be sure to check the path to make sure that there are no tools or other obstacles in the way of the workpiece.

Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

### ■ Overview

The CLOCKWISE/COUNTERCLOCKWISE HELICAL INTERPOLATION (MCW, MCC) commands are extensions of the CLOCKWISE/COUNTERCLOCKWISE CIRCULAR INTERPOLATION commands. They simultaneously execute a linear interpolation movement while moving on the circle (circular interpolation) determined by the designated center point position or radius value (R).

The tangential speed for circular interpolation will be the interpolation feed speed (F designation). This movement is called “helical interpolation.” The methods for designating the circular interpolation command portion are the same as the methods used for normal circular interpolation. The description has been omitted here.

## ■ Description

The CLOCKWISE/COUNTERCLOCKWISE HELICAL INTERPOLATION commands are designated as follows:

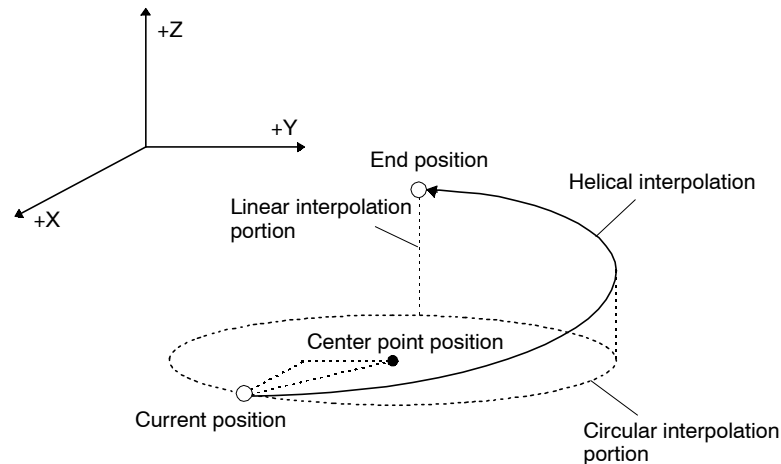
MCW  $\frac{[axis1]}{A} - \frac{[axis2]}{B} - U - \frac{[axis3]}{C} - T - \frac{E}{D}$ ;

A: End position  
 B: Center point position  
 C: Linear interpolation end position  
 D: Number of turns  
 E: Interpolation feed speed

or

MCC  $\frac{[axis1]}{A} - \frac{[axis2]}{B} - R - \frac{[axis3]}{C} - \frac{F}{D}$ ;

A: End point  
 B: Radius  
 C: Linear interpolation end position  
 D: Interpolation feed speed



**Figure 2.12 HELICAL INTERPOLATION (MCW, MCC)**

- Any axis that has not been specified in the plane designation can be designated as the linear interpolation axis. The axis need not necessarily be at right angles to the interpolation plane.
- Before specifying a CLOCKWISE/COUNTERCLOCKWISE HELICAL INTERPOLATION command, specify the plane in COORDINATE PLANE SETTING (PLN).
- Designate the axes for the end position and center point position in the same order as the axes are specified in the PLN command.
- The interpolation feed speed is also referred to as the F designation. It is specified in the speed designation (F) or in the INTERPOLATION FEED SPEED RATIO SETTING command (IFP).
- The last F designation specified in the previous block remains in effect. An interpolation command for which no F designation has been specified will generate an alarm.
- The feed speed (F) is the tangential speed of the circle on the circular plane. Therefore, the linear axis speed (F') will be  $F' = F \times (\text{Length of linear axis}) / (\text{Length of arc})$ .



The methods of designating speed commands, accelerations/decelerations, overrides, and so on, are all the same as for linear interpolation. For details, see 2.1.2 *LINEAR INTERPOLATION (MVS)*.

## ■ Programming Examples

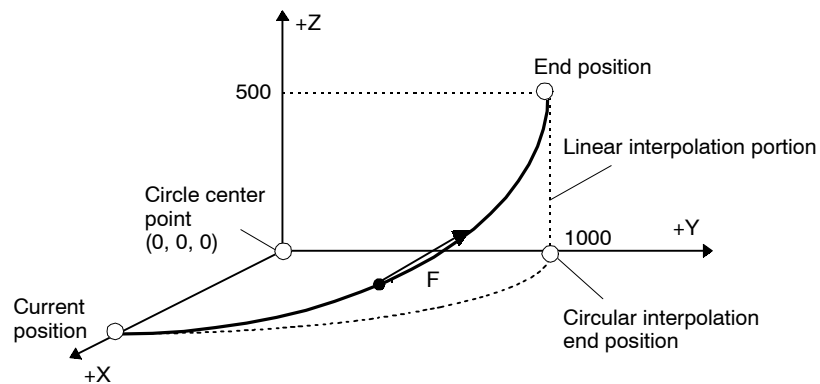
The following illustration shows a programming example for the COUNTERCLOCKWISE HELICAL INTERPOLATION command in ABS mode.

### ◀EXAMPLE▶

```

ABS;
FMX T30000000;
MW00001 = 10000; (*)
MOV [X] 1000 [Y] 0 [Z] 0;
PLN [X] [Y];
MCC [X] 0 [Y] 1000 U0 V0 Z500;

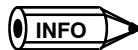
```



**Figure 2.13** Programming Example for HELICAL INTERPOLATION

\* Example of MP930

The interpolation override setting differs depending on the Machine Controller model. Refer to 1.2.3 *Feed Speeds* for details.



1. The speed designation (F) must be specified in the same block as the interpolation command.
2. The INTERPOLATION FEED SPEED RATIO SETTING command (IFP) is an isolated command, and cannot be specified in the same block as the interpolation command.
3. If the override applied to the F designation results in a speed exceeding the maximum interpolation feed speed (FMX), the speed will be clamped to the FMX speed.

## 2.1.5 ZERO POINT RETURN (ZRN)

### ■ Overview

The ZERO POINT RETURN (ZRN) command executes the zero point return operation. Up to 14 axes\* can be designated simultaneously. The resulting stop position is set as the machine coordinate origin.

The zero point return operation will be executed only for the specified axes. After the zero point return operation has been completed for all of the designated axes, program execution proceeds to the next block.

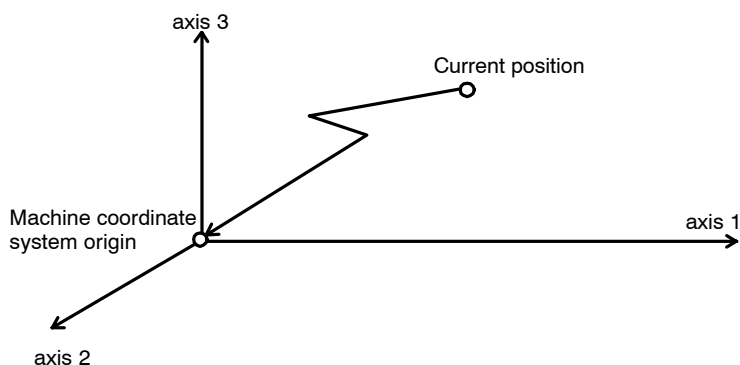
\* Number of axes with simultaneous control for the MP930.

The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to *1.1.2 Function Performance* for details.

### ■ Description

ZRN command designation and the path of movement are as follows:

ZRN [axis1] - [axis2] - ... ;  
Axis designation + 0 (position is always 0)



**Figure 2.14 Path of Movement for Zero Point Return Operation**

- When the ZRN command is executed, the returned position is set as the machine coordinate origin. At the same time, the workpiece coordinate system previously set by CURRENT POSITION SET (POS) is cancelled.

After the ZRN command has been executed, the machine coordinate system will be the same as the workpiece coordinate system. Until the next time that CURRENT POSITION SET (POS) is executed, MOVE ON MACHINE COORDINATES (MVM) will be ineffective, even if it is designated.

#### IMPORTANT

Request for temporary stop of program is disabled during ZRN command execution. To stop the operation on the way, execute request for stop of program.

- The types of zero point return operations vary in accordance with the series of machine controller.

For the MP900-series: Fixed parameter 31 (Zero Point Return Method)

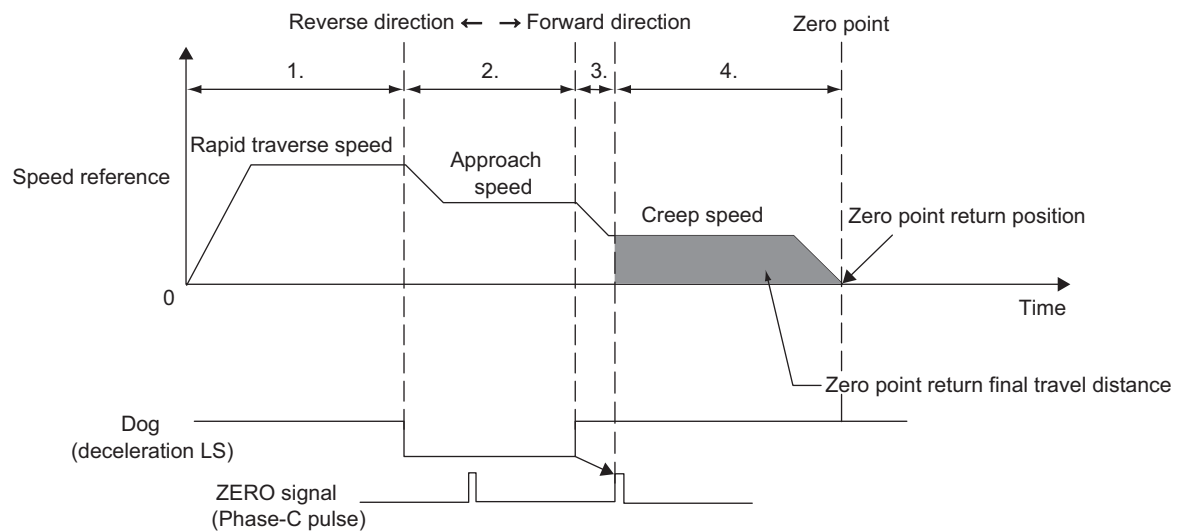
For the MP2000-series: Setting parameter OWxx3C (Zero Point Return Method)

## ■ Types of Zero Point Return Operation

The details of zero point return method are as explained below.

### DEC1 + Phase-C pulse: Parameter setting 0

The zero return operation is executed while switching the feed speed in three steps.



### ZERO signal: Parameter setting 1

The zero point return operation is executed by sending the ZERO signal instead of the phase-C pulse in the “Phase-C pulse” method of the zero point return.

### DEC1 + ZERO signal: Parameter setting 2

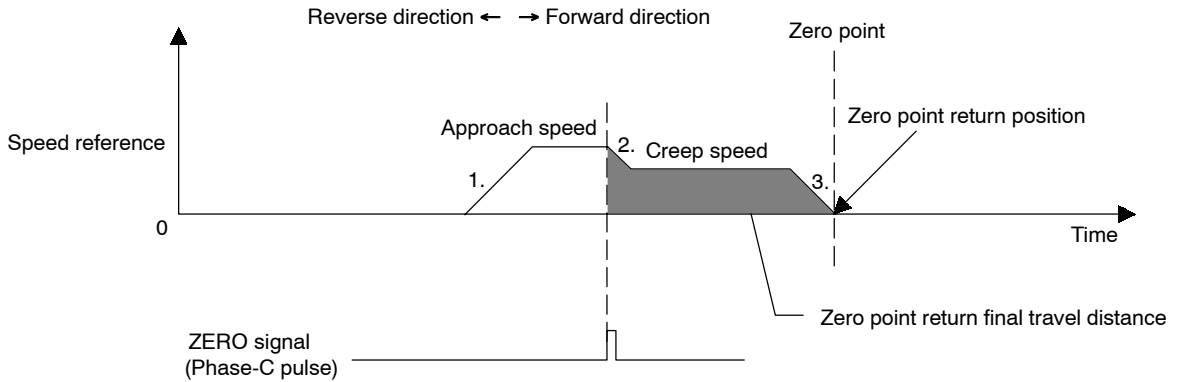
The zero point return operation is executed by sending the ZERO signal instead of the phase-C pulse in the “DEC1 + Phase-C pulse” method of the zero point return.



2

### Phase-C Pulse: Parameter setting 3

For a machine for which a limit switch (LS) such as deceleration LS can be installed, the zero point return operation is executed by sending only the phase-C pulse from the servomotor.



### DEC2 + ZERO signal: Parameter setting 4

The zero point return operation is executed by sending the ZERO signal instead of the phase-C pulse of “DEC2 + Phase-C pulse”.

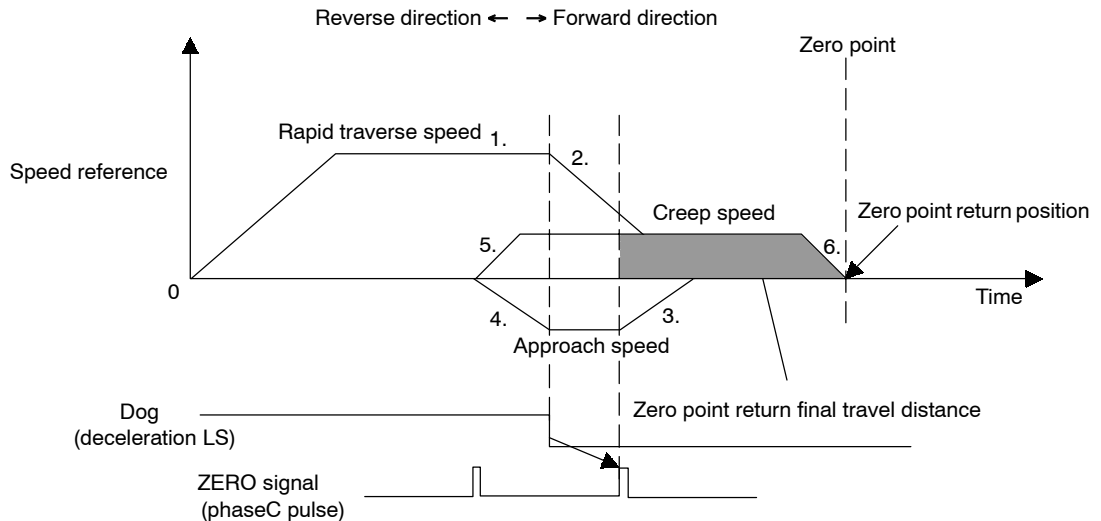
### DEC1 + LMT + ZERO signal: Parameter setting 5

The zero point return operation is executed by sending the ZERO signal instead of the phase-C pulse of “DEC1 + LMT + Phase-C pulse”.

### DEC2 + Phase-C pulse: Parameter setting 6

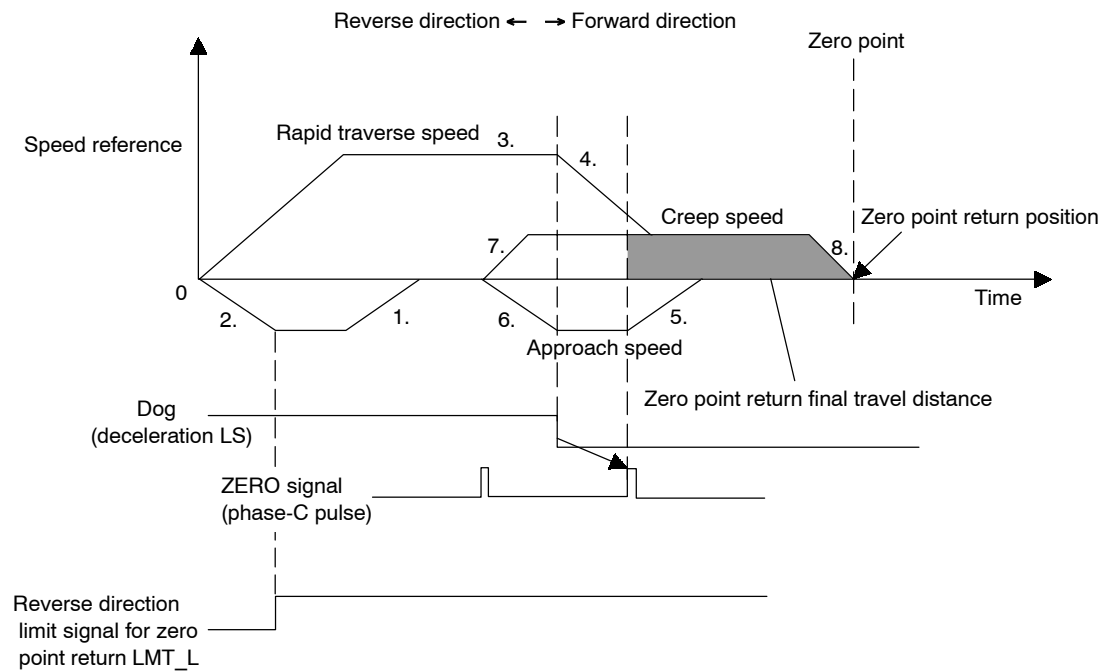
The axis travels in reverse at the approach speed, and then searches the zero point at the creep speed for the zero point return.

This method is used for machines that require repetitive accuracy.



## DEC1 + LMT + Phase-C pulse: Parameter setting 7

The axis recognizes the machine position by the forward/reverse limit (LMT) signal, and automatically performs a return operation. The zero return operation is possible from any position.

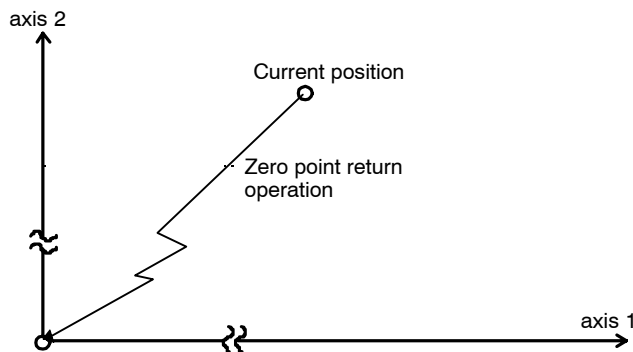


## ■ Programming Examples

A programming example for zero point return operation in ABS mode is shown in the following illustration.

### ◀EXAMPLE▶

```
ABS;
ZRN [axis1] 0 [axis2] 0;
```

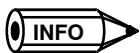


The stop position is set as the machine coordinate origin (0, 0).

**Figure 2.15 Programming Example for Zero Point Return Operation 4**

The parameters and Servopack user constants relating to zero point return are shown in the following table.

- Zero point return direction      Setting parameter Operation Mode Settings (OWxx0069 Home Position Return Direction)
- Zero point return feed speed      Setting parameter 30 Rapid Traverse Speed (OLxx22)
- Approach speed                      Cn-0022 Zero Point Return Approach Speed 1
- Creep speed                            Cn-0023 Zero Point Return Approach Speed 2
- Final travel distance                Cn-0028, Zero Point Return Final Travel Distance



Be sure to specify 0 after the axis. If 0 is omitted, an error will result.

## 2.1.6 SKIP FUNCTION (SKP)

### ■ Overview

The SKIP FUNCTION (SKP) uses linear interpolation to move up to 14 axes\* simultaneously from the current position to the end position at interpolation feed speed (F). When the skip signal turns ON during axis movement, the moving axis is decelerated to stop and the remaining travel distance is cancelled. Any axis not specified in the command will not be moved.

When the skip signal turns ON during axis movement in a SKP command block, axis movement is decelerated to stop, the remaining travel distance for that block is cancelled, and program execution proceeds to the next block. In this way, the SKP command enables the programming of motion control that can respond to external conditions.

\* Number of axes with simultaneous control for the MP930.

The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to *1.1.2 Function Performance* for details.

### ■ Description

The SKP command is designated as follows:

SKP	[axis1]	-	[axis2]	-	...	F=	SS=;
	A		B			C	

A: Reference position  
B: Interpolation feed speed  
C: Skip selection

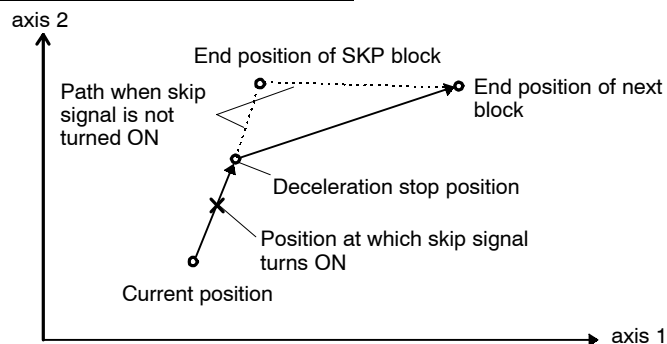


Figure 2.16 SKP Command Path

- When a ladder logic programs are created by automatic generation, the skip input signals can control up to 14 axes\* divided into four groups. Two points can be allocated for each group.
- The skip input signal is selected by writing the number 1 or 2 for the skip selection (SS). Decide which skip signal is to correspond to 1 or 2 by first allocating the control signals on the Group Definition window.
- The skip signal is input to the control signal for the MSEE command.
  - Skip input signal 1: Bit 8 of the control signal
  - Skip input signal 2: Bit 9 of the control signal

\* Number of axes with simultaneous control for the MP930.

The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to *1.1.2 Function Performance* for details.



1. The speed designation (F) must be specified in the same block as the SKP command.
2. The INTERPOLATION FEED SPEED RATIO SETTING command (IFP) is an isolated command, and cannot be specified in the same block as the interpolation command.
3. If the override applied to the F designation results in a speed greater than the maximum interpolation feed speed (FMX), the speed will be clamped to the FMX speed.
4. The moving axis decelerated to a stop. The SKP command, however, remains in effect until the positioning completion signal turns ON.

## 2.1.7 SET TIME POSITIONING (MVT)

### ■ Overview

The SET TIME POSITIONING (MVT) command uses the same operations as POSITIONING (MOV) to move each axis from the current position to the end position. To complete positioning in the specified time, the feed speed of each axis is clamped. This command does not use an interpolation operation, and there is no restriction on completing the positioning for all the designated axes simultaneously. There is a time lag with the acceleration/deceleration setting.

Up to 14 axes\* can be designated simultaneously. Any axis that is not specified in the command will not be moved.

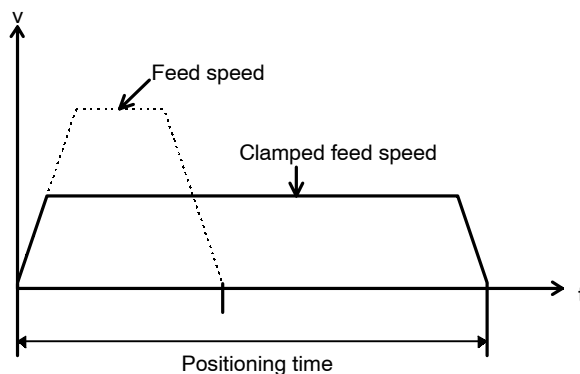
\* Number of axes with simultaneous control for the MP930.

The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to *1.1.2 Function Performance* for details.

### ■ Description

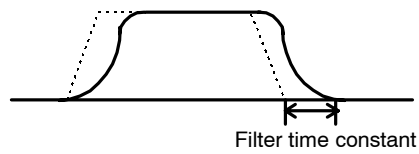
The MVT command is designated as follows:

MVT [axis1] - [axis2] - ... T-;  
 Reference position    Positioning time (ms)



**Figure 2.17 SET TIME POSITIONING (MVT)**

- If an override is used, positioning will not be completed in the specified time.
- If a filter is used, the positioning time will be delayed by the filter time constant.



**Figure 2.18 Positioning Time Delay When a Filter is Used**

**IMPORTANT**

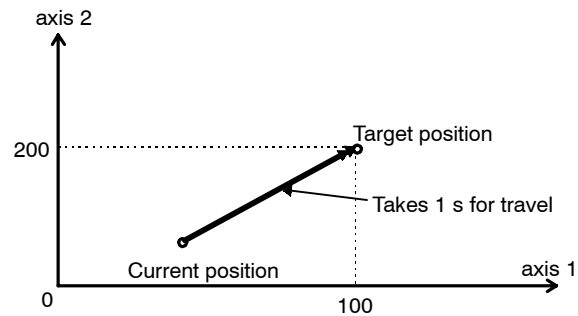
1. If the positioning time is set to 0, an alarm will occur in the motion program.
2. If the travel distance of any axis is 0, an alarm will be generated.

**■ Programming Examples**

The following illustration shows a programming example for SET TIME POSITIONING (MVT) in ABS mode.

**◀EXAMPLE▶**

```
ABS;
MVT [axis1] 100 [axis2] 200 T1000;
```

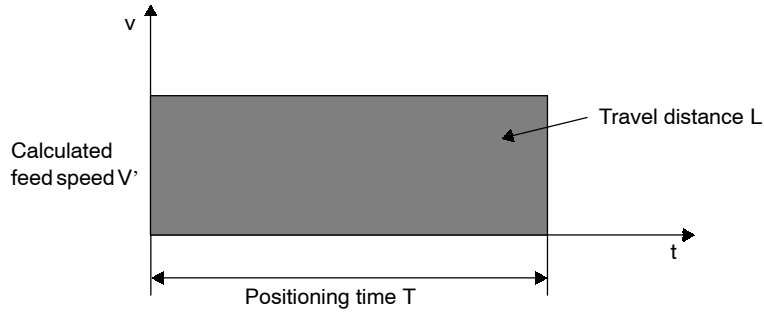


**Figure 2.19 Programming Example for SET TIME POSITIONING (MVT)**

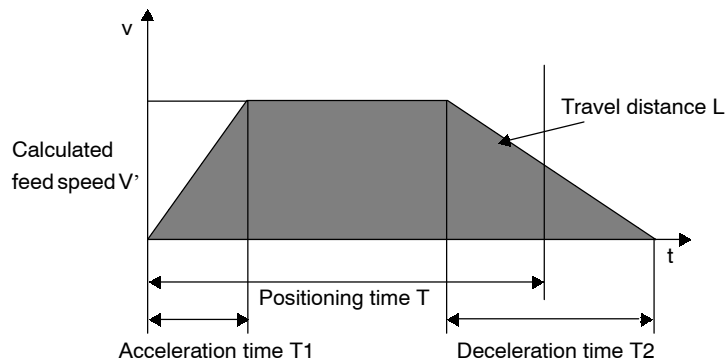


The operation specifications of the MVT command is identical for all the MP-series Machine Controller models.

Inside the MP-series Machine Controller, the feed speed during the execution of MVT command does not include acceleration/deceleration.



The actual operation when the acceleration time  $T1$  is less than the deceleration time  $T2$ , the actual operation time can be calculated by the following equation.



If the acceleration time is equal to the deceleration time ( $T1 = T2$ ), the positioning time designated by the MVT command is the time at the start of deceleration.

The setting parameter  $OL_{xx22}/OL_{xx10}$  (Rapid Traverse Speed) for each axis is changed by the MVT command. The feed speed set by the VEL command will also be changed accordingly. After executing the MVT command, reset the feed speed by using the VEL command.

The acceleration/deceleration time for SVA and MECHATROLINK differs as follows:

- SVA: Depends on the settings parameters for the acceleration/deceleration time and the selected filter.
- MECHATROLINK: Depends on the acceleration/deceleration time constant and the filter setting of SERVOPACK.

## 2.1.8 EXTERNAL POSITIONING (EXM)

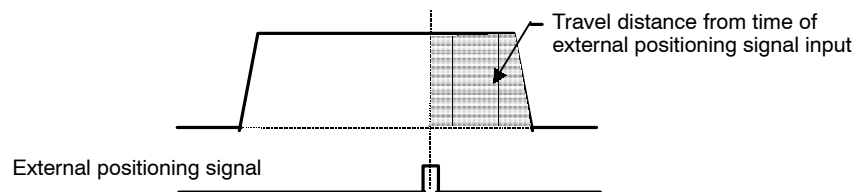
### ■ Overview

When the external positioning signal is input while positioning is being executed, the EXTERNAL POSITIONING (EXM) command completes positioning by using an incremental value to move the axis the specified travel distance, and then executes the next block.

### ■ Description

The EXM command is designated as follows:

<pre>EXM [axis] - D-;       A   B</pre> <p>A: Reference position B: Travel distance from time of external positioning signal input (incremental distance)</p>
---



**Figure 2.20 EXTERNAL POSITIONING (EXM) Signal**

When a negative value is specified for the travel distance, the axis decelerates to stop, and then moves in the negative direction.

- The EXM command is valid for only one axis. It is not a modal command.
- If a machine lock occurs, EXM ignores the signal and executes only the designated travel distance.
- If no external signal is input, the axis moves to the reference position.

#### IMPORTANT

1. Connect the external positioning signal to the Servopack 1CN 10-pin external latch (/EXT) connector.
2. The external latch (/EXT) input signal is also used by the zero point return when using a zero point limit switch.
3. The unit of D-designation (moving amount from external positioning signal input) is pulse.



## 2.2 Control Commands

The control commands control the motion of each axis. This section explains the programming methods for the basic control commands.

### 2.2.1 ABSOLUTE MODE (ABS)



## Caution

- The movement of a coordinate word designated in ABS mode is entirely different from that of the same coordinate word designated in INC mode. Before starting operations, be sure to check that the ABS or INC command is specified correctly.

Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

#### ■ Overview

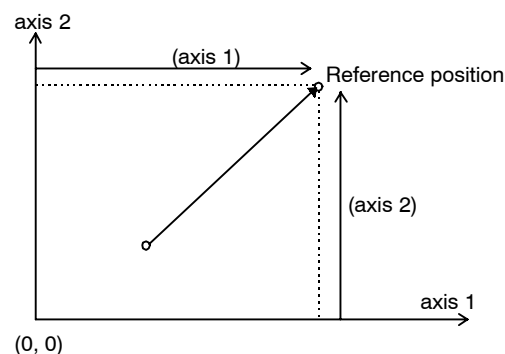
The ABSOLUTE MODE (ABS) command causes the coordinate words that control axis movement to be treated as absolute values in the workpiece coordinate system.

Once ABS mode has been executed, it remains in effect until INCREMENTAL MODE (INC) is next executed. ABS mode is the default mode when the power is turned ON.

#### ■ Description

The method of designating ABS mode is as follows:

```
ABS;
or
ABS MOV [axis I] -;
```



**Figure 2.21 Absolute Values in the Workpiece Coordinate System**

In this manual, absolute mode is sometimes abbreviated as ABS mode.

## ■ Programming Examples

The following illustration shows a programming example in ABS mode.

### ◀EXAMPLE▶

```
ABS MOV [axis1] 100 [axis2] 200;
```

```
MOV [axis1] 50 [axis2] 100;
```

```
MOV [axis1] 100;
```

```
MOV [axis2] 50;
```

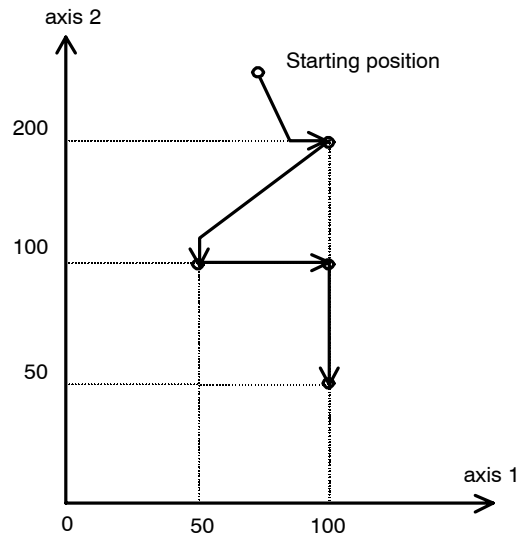


Figure 2.22 Programming Example in ABS Mode

## 2.2.2 INCREMENTAL MODE (INC)

### ■ Overview

The INCREMENTAL MODE (INC) command causes the coordinate words that control axis movement to be treated as incremental values from the current position in the workpiece coordinate system.

Once INC mode has been designated, it remains in effect until ABSOLUTE MODE (ABS) is next designated. ABS mode is the default mode when the power is turned ON.

### ■ Description

The method of designating INC mode is as follows:

```
INC;
or
INC MOV [axis1] -;
```

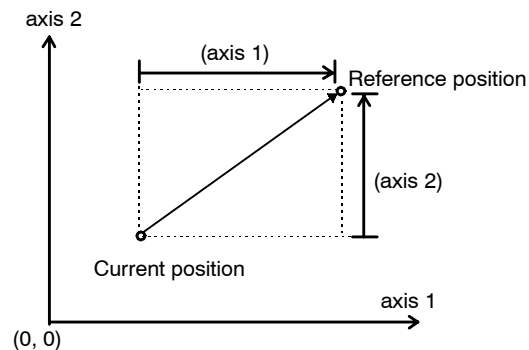


Figure 2.23 Incremental Values in the Workpiece Coordinate System

### ■ Programming Examples

The following illustration shows a programming example in INC mode.

#### ◀EXAMPLE▶

```
INC MOV [axis1] 100 [axis2] 200;
```

```
MOV [axis1] 50 [axis2] 100;
```

```
MOV [axis1] 100;
```

```
MOV [axis2] 50;
```

Where

Starting position: axis 1 = axis 2 = 0

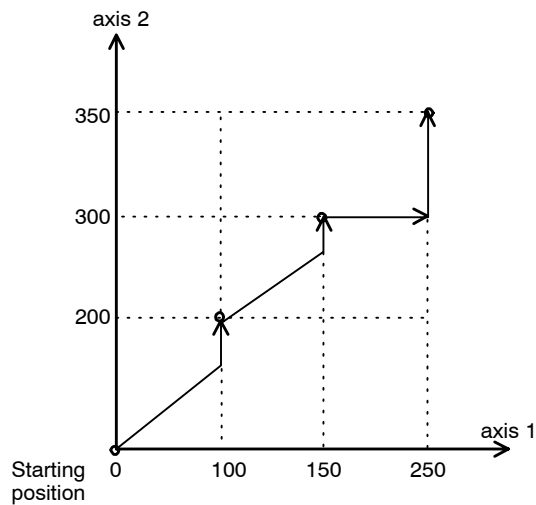


Figure 2.24 Programming Example in INC Mode

### 2.2.3 CURRENT POSITION SET (POS)



## Caution

- Care is required with the CURRENT POSITION SET (POS) command.

The CURRENT POSITION SET (POS) command is used to create new workpiece coordinate system values. If POS is specified incorrectly, subsequent move operations will be entirely different. Before starting operations, be sure to check that the workpiece coordinate system is specified correctly.

Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

#### ■ Overview

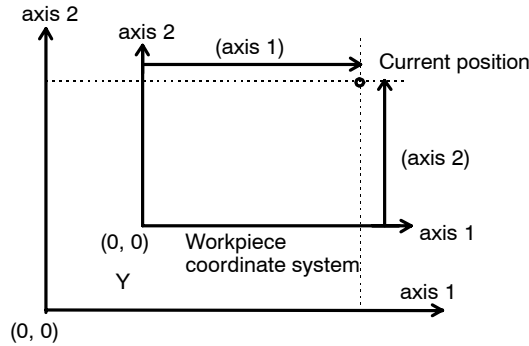
The CURRENT POSITION SET (POS) command is used to create a new coordinate system by switching the current position to the desired coordinate values. In this manual, this newly set coordinate system is called a “workpiece coordinate system.” Move commands designated after the POS command are executed in the workpiece coordinate system.

■ **Description**

The POS command is designated as follows:

```

POS [axis1] - [axis2] - ... ;
    Desired coordinate value
    
```



**Figure 2.25 POS Command (Workpiece Coordinate Change)**

- The workpiece coordinate system can be changed as often as desired using the POS command. The machine coordinate system must be set in advance. The machine coordinate system is not affected by the POS command.
- Up to 14 axes\* can be designated simultaneously with the POS command. The current value of any axis for which this command is omitted cannot be re-written.
- Move commands in a workpiece coordinate system cannot exceed the maximum programmable values when converted to coordinates in the machine coordinate system

\* Number of axes with simultaneous control for the MP930.

The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to 1.1.2 Function Performance for details.

Table 2.1 shows the setting status of the machine coordinate system and the workpiece coordinate system.

**Table 2.1 Current Position Set (POS)**

Controller Status	Incremental Position Detecting System	Absolute Position Detecting System
<b>After power ON</b>	Machine coordinate system: Default setting (*1) Workpiece coordinate system: Cancelled (*3)	Machine coordinate system: Yes (*2) Workpiece coordinate system: Cancelled
<b>After ZERO POINT RETURN (ZRN)</b>	Machine coordinate system: Set Workpiece coordinate system: Cancelled	Workpiece coordinate system: Cancelled
<b>After CURRENT POSITION SET (POS)</b>	Workpiece coordinate system: Set	Workpiece coordinate system: Set
<b>After ZERO POINT POSITION</b>	Workpiece coordinate system: Cancelled	Machine coordinate system: Set Workpiece coordinate system: Cancelled

- \* 1. Default setting: The current position is set as the machine coordinate origin when the power is turned ON. If ZERO POINT RETURN is then not executed, the software limit switch function will not be effective.
- \* 2. Yes: The machine coordinate origin is set using the position information in the absolute position detecting encoder.
- \* 3. Cancelled: The previously set workpiece coordinate system is cancelled, and the workpiece coordinate system is the same as the machine coordinate system.

**IMPORTANT**

For infinite-length axes, set a value within the range from 0 to POSMAX. If a value out of the setting range is set for any infinite-length axis, an alarm will be generated.

## 2.2.4 COORDINATE PLANE SETTING (PLN)

2

### ■ Overview

The COORDINATE PLANE SETTING (PLN) command defines two logical axes set in the parameters as a coordinate plane. Execute the PLN command before executing a command in which a coordinate plane is specified.

Once designated, the coordinate plane remains in effect until it is redefined or until PROGRAM END is executed.

### ■ Description

The PLN command is designated as follows:

X-axis name	Y-axis name
↓	↓
PLN [ <u>axis1</u> ] [ <u>axis2</u> ];	
Designate the two axes of the coordinate plane.	

### ■ Programming Examples

The following illustration shows a programming example for the PLN command.

**EXAMPLE**

PLN [A] [B]; Define a plane composed of axes A and B.

MCW [A] 50 [B] 50 R50 F1000;

Where

Starting position: A = B = 0

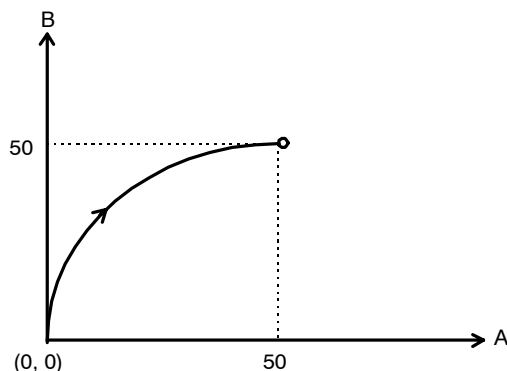
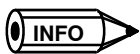
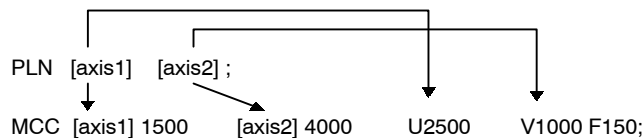


Figure 2.26 Programming Example for COORDINATE PLANE SETTING (PLN)



Designate the axes for the end position and center point position for circular interpolation and helical interpolation in the same order as the axes are specified in the PLN command.



## 2.2.5 MOVE ON MACHINE COORDINATES (MVM)

2



### Caution

- The MOVE ON MACHINE COORDINATES (MVM) command is used to position the coordinate positions in a machine coordinate system. If the machine coordinate origin is designated without being verified, unexpected move operations will result. Before starting operations, be sure to check that the position designated in the machine coordinate system is correct.

Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

### ■ Overview

The MOVE ON MACHINE COORDINATES (MVM) command is used to move axes in a machine coordinate system after a workpiece coordinate system that is different from the machine coordinate system has been set by CURRENT POSITION SET (POS).

### ■ Description

The MVM command is designated as follows:

```

MVM MOV .....;
or
MVM MVS .....;
  
```

When this command is executed, either POSITIONING (MOV) or LINEAR INTERPOLATION (MVS) is used to move on the absolute coordinate position of the machine coordinate system. This command is always executed in ABS mode, regardless of the ABS/INC setting.

The MVM command is valid only for the block in which it is designated. For example, LINEAR INTERPOLATION (MVS) commands in the next and subsequent blocks will be executed in the workpiece coordinate system. See the following illustration.

## ■ Programming Examples

The following illustration shows a programming example for the MVM command.

### ◀EXAMPLE▶

```
MVM MVS [axis1] 50 [axis2] 150 F1000;
```

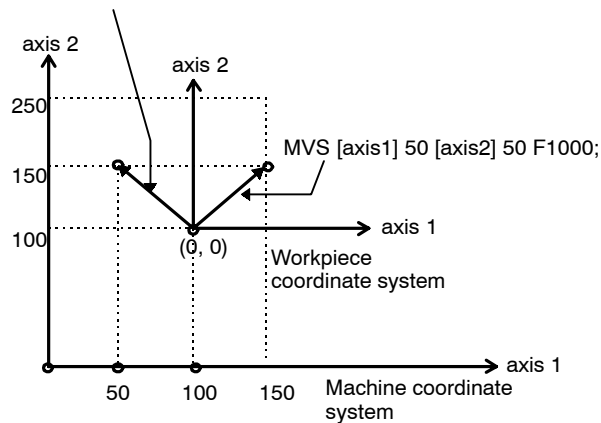


Figure 2.27 Programming Example for MOVE ON MACHINE COORDINATES (MVM)

## 2.2.6 PROGRAM CURRENT POSITION UPDATE (PLD)

### ■ Overview

If an axis movement is executed outside a motion program while the motion program is running (such as when an axis is moved by JOG, STEP, or a user function), the program current position will not be updated. If the motion program were to continue to be executed in this status, the axis would need to be moved the designated travel distance to the desired position by manual intervention. To solve this problem, PROGRAM CURRENT POSITION UPDATE (PLD) is used to update the current position.

This command is not supported in block operations. (Execution will not be stopped for it in block operation mode.)

### ■ Description

The PLD command is designated as follows:

```
PLD [axis1] [axis2] ... [axis] ;
[axis1]...[axis2]: Only the specified axes will be updated.
```



## ■ Programming Examples

The following illustrations show programming examples for the PLD command.

### ◀EXAMPLE▶

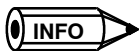
#### Manual Intervention While a Motion Program is Running

```
MPM001 "GRP1"
MOV [axis1] 1000;      ←During this time, axis 1 is moved by JOG.
PLD [axis1];          ←Updates the current position.
MVS [axis1] - 1000;
```

### ◀EXAMPLE▶

#### Axis Executed Within a Motion Program User Function

```
MPM001 "GRP1"
MOV [axis1] 1000;
UFC FNC10 MB00000 IW00100 MB00020   ←Axis 1 is moved by a user function.
PLD [axis1]; ←Updates the current position.
MVS [axis1] - 1000;
```



The PLD command can be executed by the user in some applications. The PLD command can not be used in some of the applications where manual intervention is required while the motion program is running.

## 2.2.7 DWELL TIME (TIM)

### ■ Overview

The DWELL TIME (TIM) command causes execution to pause for the period of time specified for "T" before the start of the next command and before proceeding to the next block. No other commands can be designated with the TIM command.

### ■ Description

The TIM command is designated as follows:

TIM T ; Dwell time
-----------------------

The range of time that can be designated for T is 0.01 to 600.00 seconds.

Designate the time T with no decimal point: 1=0.01 sec. The decimal point is not affected by the setting of the "number of digits after the decimal point" of the fixed parameter.

## ■ Programming Examples

The following illustration shows a programming example for the TIM command.

### ◀EXAMPLE▶

```
MOV [axis1] 100;
```

```
TIM T250; (2.5 s)
```

The TIM command is executed after positioning has been completed.

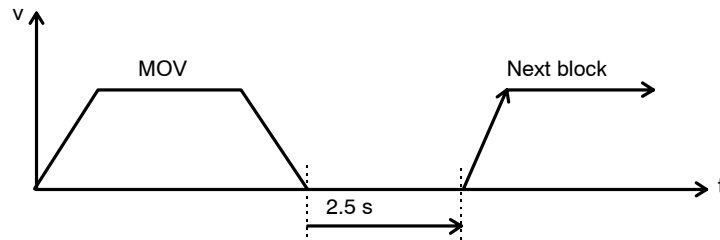


Figure 2.28 Programming Example for DWELL TIME (TIM)

## 2.2.8 PROGRAM END (END)

The PROGRAM END (END) command causes program operation to end.

No other commands can be designated in the same block as the END command.

### ■ Description

The END command is designated as follows:

END ; Program end
----------------------

Program operation will end after execution of the block for this command.

If there is a move command in the previous block, the program will end after the in-position check has been completed.

# 3

## Advanced Programming

This chapter describes the high-level control commands, and speed and acceleration/deceleration commands, used for motion control, and describes the programming methods used.

<b>3.1 High-Level Control Commands</b> .....	<b>3 -2</b>
3.1.1 SECOND IN-POSITION CHECK (PFN) .....	3 -2
3.1.2 SET SECOND IN-POSITION RANGE (INP) .....	3 -4
3.1.3 IGNORE SINGLE-BLOCK SIGNAL (SNG), SINGLE-BLOCK SIGNAL DISABLED/ENABLED (SNGD/SNGE) .....	3 -6
3.1.4 USER FUNCTION CALL (UFC) .....	3 -8
3.1.5 I/O VARIABLE WAIT (IOW) .....	3 -16
3.1.6 SUBROUTINE CALL (MSEE) .....	3 -17
3.1.7 SUBROUTINE RETURN (RET) .....	3 -18
3.1.8 ONE SCAN WAIT (EOX) .....	3 -19
3.1.9 Branching Commands: IF...ELSE...IEND .....	3 -20
3.1.10 Repeat Commands: WHILE...WEND .....	3 -21
3.1.11 Parallel Execution Commands (PFORK, JOINTO, PJOINT) .....	3 -23
3.1.12 Selective Execution Commands (SFORK, JOINTO, SJOINT) .....	3 -27
<b>3.2 Speed and Acceleration/Deceleration     Commands</b> .....	<b>3 -29</b>
3.2.1 ACCELERATION TIME CHANGE (ACC) .....	3 -29
3.2.2 DECELERATION TIME CHANGE (DCC) .....	3 -31
3.2.3 S-CURVE TIME CONSTANT CHANGE (SCC) ...	3 -32
3.2.4 SET SPEED (VEL) .....	3 -34
3.2.5 INTERPOLATION FEED SPEED RATIO SETTING (IFP) .....	3 -36
3.2.6 MAXIMUM INTERPOLATION FEED SPEED (FMX) .....	3 -37
3.2.7 INTERPOLATION ACCELERATION TIME CHANGE (IAC) .....	3 -39
3.2.8 INTERPOLATION DECELERATION TIME CHANGE (IDC) .....	3 -41

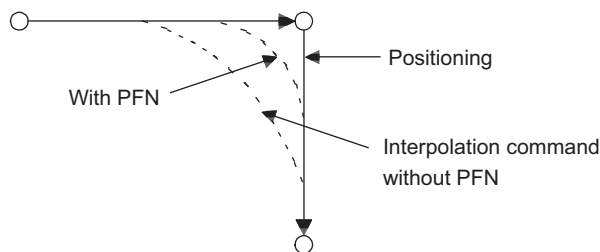
## 3.1 High-Level Control Commands

This section describes the programming methods for the high-level commands used for motion control.

### 3.1.1 SECOND IN-POSITION CHECK (PFN)

#### Overview

The IN-POSITION CHECK (PFN) command is used to pass near the end position (rounding off) when movement is based on interpolation. It is used together with the INP command, and operations proceed to the next block after the second in-position check is completed as set by the INP command.



#### When using an MP900-series Machine Controller

After the interpolation command is sent, the second in-position completed (IWxx17 bit 2) is turned ON, and operations proceed to the next block.

The timing at which the second in-position completed signal is turned ON varies in accordance with the series of the machine controller used and the settings of the parameters.

Machine Controller Series		Setting of the second in-position width (OWxx32)	
		Not 0	0
MP930, MP940	If PENMODE (OWxx01 Bit 10) is OFF	When satisfying the requirements of <Condition 1> and <Condition 2>	When satisfying the requirements of <Condition 1>
	If PENMODE (OWxx01 Bit 10) is ON	When satisfying the requirements of <Condition 2>	When satisfying the requirements of <Condition 3>
MP920		When satisfying the requirements of <Condition 2>	When satisfying the requirements of <Condition 3>

Condition 1: Discharging completed [DEN (IWxx15 bit2)] is ON.

Condition 2:  $|MPOS - APOS| \leq$  Second in-position width (OWxx32)

MPOS: Machine coordinate system position (ILxx18)

APOS: Machine coordinate feedback position (ILxx08)

Condition 3: Positioning completed signal [POSCOMP (IWxx00 Bit13)] is ON.



#### ◆ In-position check

After the movement of the designated block has started deceleration, this function detects that the axis movement has entered the positioning completed range.

## When using an MP2000-series Machine Controller

After the interpolation command is sent, the position proximity (NEAR) signal (IWxx0C bit 3) is turned ON and operations proceed to the next block.

The timing at which the second in-position completed signal is turned ON is determined in accordance with the following conditions.

- If the positioning completed width 2 (OLxx20) is set to any value other than 0:  
 $|MPOS - APOS| \leq \text{Positioning completed width 2 (OLxx20)}$   
 MPOS: Machine coordinate system position (ILxx12)  
 APOS: Machine coordinate feedback position (ILxx16)
- If positioning completed width 2 (OLxx20) is set to 0:  
 Discharging completed [DEN (IWxx15 bit2)] is ON.

### ■ Description

The PFN command is designated in the following two ways:

#### When designating in the same block as the interpolation command

```
MVS [axis1] 100. [axis2] 200. F1000 PFN;
```

After the MVS command has been sent, the second in-position check is completed as set by the INP command, then operations proceed to the next block.

#### When making a single designation

```
MVS[axis1] - [axis2];  
PFN[axis1][axis2];  
MVS[axis1] - [axis2] -;
```

Here, [axis1] and [axis2] are used for the second in-position check as set by the INP command, then operations proceed to the next block if it is within the range.



If there is no PFN command, or if the second in-position range set by the INP command is 0, execution of the next block begins as soon as the MVS command pulses have been distributed.

### ■ Programming Examples

#### ◀ EXAMPLE ▶

```
MVS [axis1] 200. F100 PFN;      Second in-position check after linear interpolation  
MOV [axis1] 400. [axis2] 400.;  Move by positioning
```

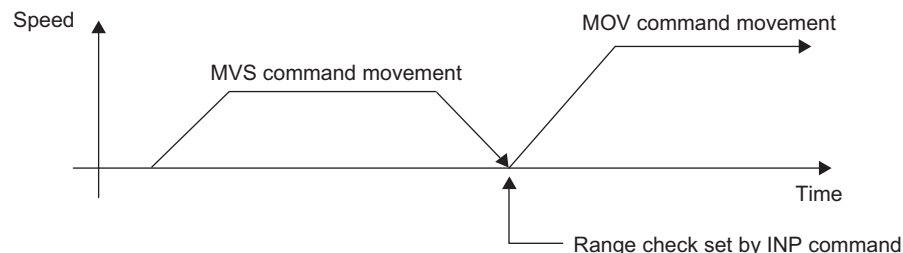


Figure 3.1 Second In-position Check Set by the INP Command

### 3.1.2 SET SECOND IN-POSITION RANGE (INP)

#### Overview

The SET SECOND IN-POSITION RANGE (INP) command is used to pass through a position near the end position (rounding off) when movement is based on interpolation with PFN designation.

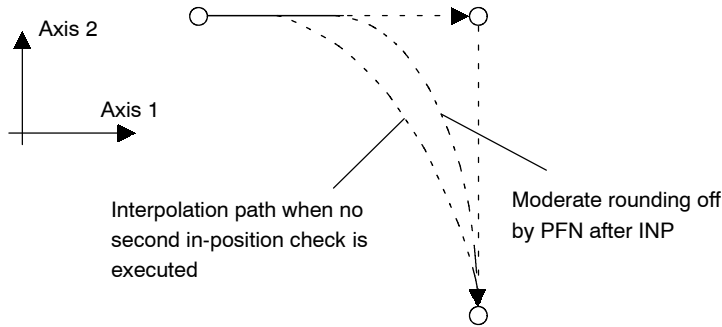


Figure 3.2 SET SECOND IN-POSITION RANGE Command

#### Description

The INP command is designated as follows:

```
INP [axis1] - [axis2] - ... ;
Second positioning completed range
```

The command above stores the command values in the following parameter.

- (a) When using an MP900-series Machine Controller  
Second in-position width (OWxx32)
- When using an MP2000-series Machine Controller  
Positioning completed width 2 (OLxx20)

Execution proceeds to the next block when the deviation between the machine coordinate system position and the machine coordinate feedback position is detected as entering the range of the above parameter by an interpolation command for which PFN is designated or by a single PFN command.

If there is more than one moving axis, execution proceeds to the next block after the designated range has been entered by all axes.

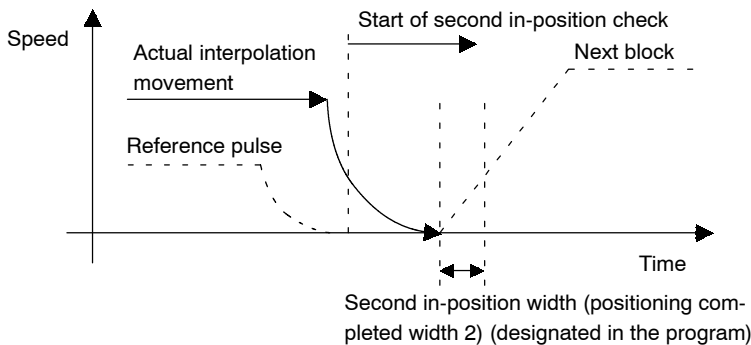


Figure 3.3 SET SECOND IN-POSITION RANGE Operation

3

The setting for the second in-position width (MP2000 Series: positioning completed width 2) is as follows:

1 to 65,535 [reference units]

As shown in the above illustration, the second in-position check starts as soon as pulse distribution for the designated block has been completed. Therefore, if a fairly large value is set by the INP command, the second in-position check will end immediately at the point where pulse distribution is completed, and execution will proceed to the next block.

- Once the INP command has been set, the second in-position width (MP2000 Series: positioning completed width 2) will remain in effect for all subsequent interpolation commands for which PFN is designated until it is cancelled.
- To cancel the second in-position check, specify 0 for the second in-position width (MP2000 Series: positioning completed width 2) for the axis to be cancelled. Only those axes for which 0 is specified will be cancelled.

## ■ Programming Examples

### ◀EXAMPLE▶

ABS MOV [axis1]0 [axis2]0;	Positioning to zero point
INP [axis1](a) [axis2](b);	Sets the second in-position width
MVS [axis1]100 PFN;	Linear interpolation in X-axis direction
MVS [axis2]100 PFN;	Linear interpolation in Y-axis direction
MVS [axis1]-100 PFN;	Linear interpolation in X-axis direction

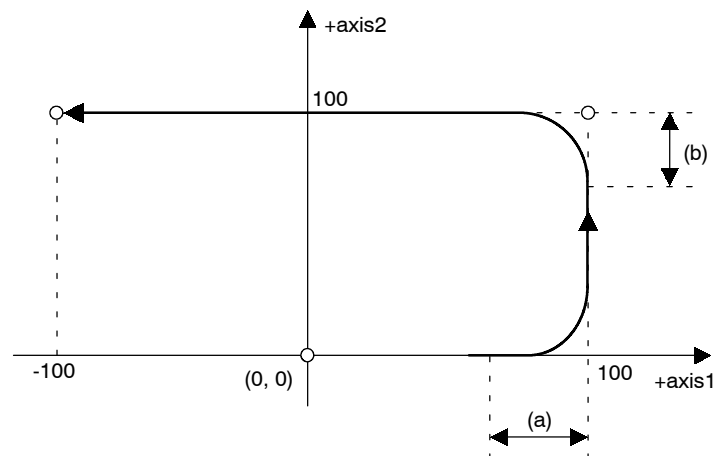


Figure 3.4 Programming Example for SECOND IN-POSITION CHECK

### 3.1.3 IGNORE SINGLE-BLOCK SIGNAL (SNG), SINGLE-BLOCK SIGNAL DISABLED/ENABLED (SNGD/SNGE)

#### ■ MP900 Series: IGNORE SINGLE-BLOCK SIGNAL (SNG)

##### Overview

The SNG command is used to continuously execute specific blocks only when executing a program in single-block operation mode.

##### Description

The SNG command is designated as follows:

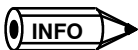
<p>SNG &lt;Command that moves axes&gt; or SNG &lt;Command that does not move axes&gt;</p>
---

In single-block operation mode, a block in which the SNG command is coded is executed immediately without a single-block stop.

- The following commands can be designated in the SNG command:

MOV, MVS, MCW, MCC, ZRN, SKP, MVT,  
EXM, ACC, SCC, PFN, VEL, INP, EOX

Since commands other than those listed above are completed in one scan, the SNG command cannot be designated.



Single-block execution is possible only for axis move commands and speed and acceleration/deceleration commands. It is meaningless to use the SNG command with any other commands.

##### Programming Examples

###### ◀EXAMPLE▶

```
MVS [axis1] 0 [axis2] 0;
SNG MVS [axis1] 100 [axis2] 200;
MB000101 = 1;
MB000102 = 1;
MB000103 = 1;
```

Execution will be continuous, even in single-block operation mode.



- ◆ Single-block operation mode

In single-block operation mode, a stop is executed for each block. However, the following restriction applies to MP900-series Machine Controllers. (There is no restriction on MP2000-series Machine Controllers.)

Restriction: A stop cannot be executed for each block for any commands other than axis move commands and speed and acceleration/deceleration commands.



## ■ MP2000 Series: SINGLE-BLOCK SIGNAL DISABLED/ENABLED (SNGD/SNGE)

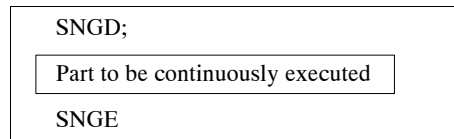
### Overview

The SNGD command and SNGE command are used to continuously execute blocks contained within SNGD and SNGE when executing a program in single-block operation mode.

The SNGD and SNGE commands are only used by the MP2000 Series. They cannot be used by the MP900 Series.

### Description

The SNGD and SNGE commands are designated as follows:



In single-block operation mode, a block that is contained within SNGD and SNGE is executed immediately without a single-block stop.

### Programming Examples

#### ◀EXAMPLE▶

```
MVS [axis1]0 [axis2]0;
```

**SNGD**

```
MVS [axis1]100 [axis2]200;"①"
```

```
MB000101 = 1;"②"
```

```
MB000102 = 1;"③"
```

**SNGE**

```
MB000103 = 1;
```

Execution will be continuous, even in single-block operation mode, for blocks ① to ③ that are contained within SNGD and SNGE.

### 3.1.4 USER FUNCTION CALL (UFC)

#### ■ Overview

The UFC command is used to call functions created by the user, using function name designations.

#### ■ Description

The UFC command is designated as follows:

```

UFC Function_name Input_data, Input_address, Output_data
                                     *
Function name: ASCII 8 bytes
Input data:    Max. 16 data items (at least 1 data item is required)
Input address: Max. 1 address
Output data:   Max. 16 data items (1 data item is mandatory)
    
```

\* The input address can be omitted.

[input data, output data] is used if there is no input address. At least one input data item and one output data item are mandatory.

The UFC command calls a user function. When execution of the user functions has been completed, execution proceeds to the next block after UFC.

#### ■ Programming Examples

```

UFC KANSUU      MB00000 IW0010 MB00020, MA00100,
    Function name      Input data      Input address

MB00001 MW00200 ML00201;
    Output data
    
```

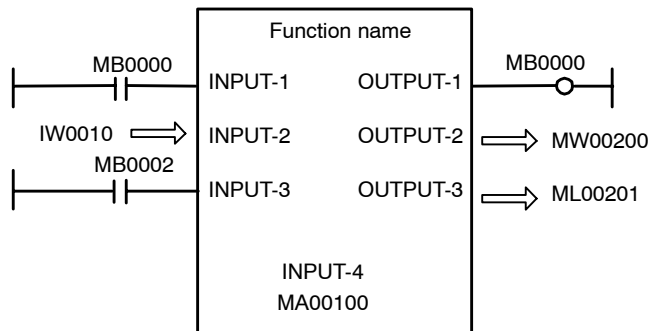


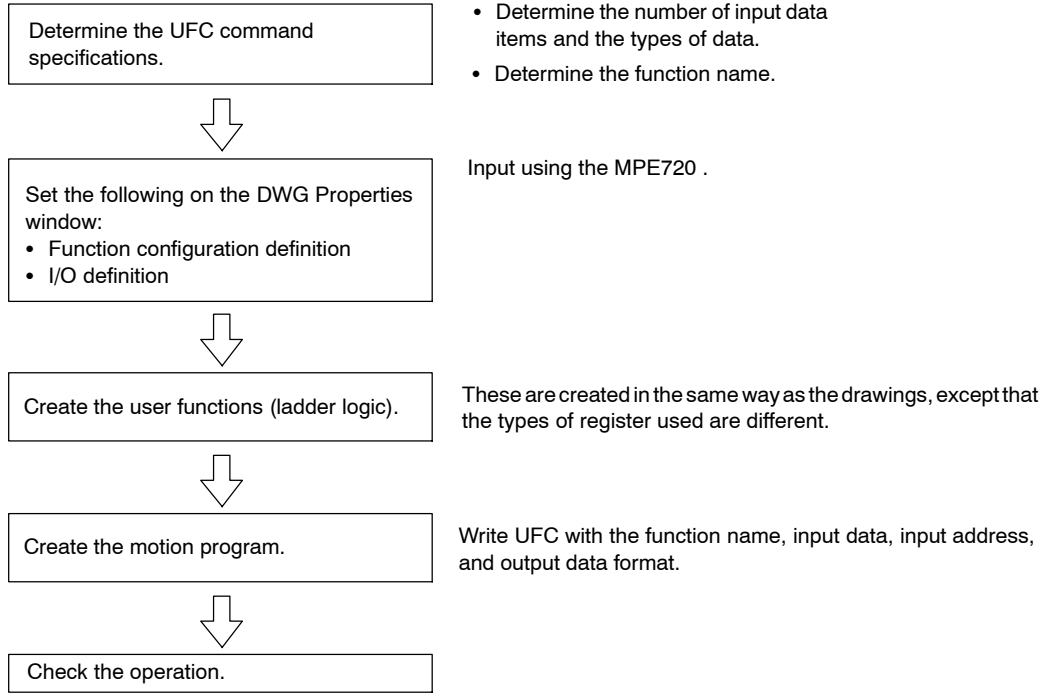
Figure 3.5 Programming Example for USER FUNCTION CALL (UFC)

**IMPORTANT**

If creating a user function, set the flag corresponding to YB00 so that it turns ON when the user function has successfully been executed. In this way, the execution of the function will continue to the next block when the YB00 turns ON. If the YB00 does not turn ON, execution of the function may not proceed to the next block.

## ■ UFC Command Creation Procedure

The procedure for creating a UFC command is as follows:



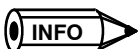
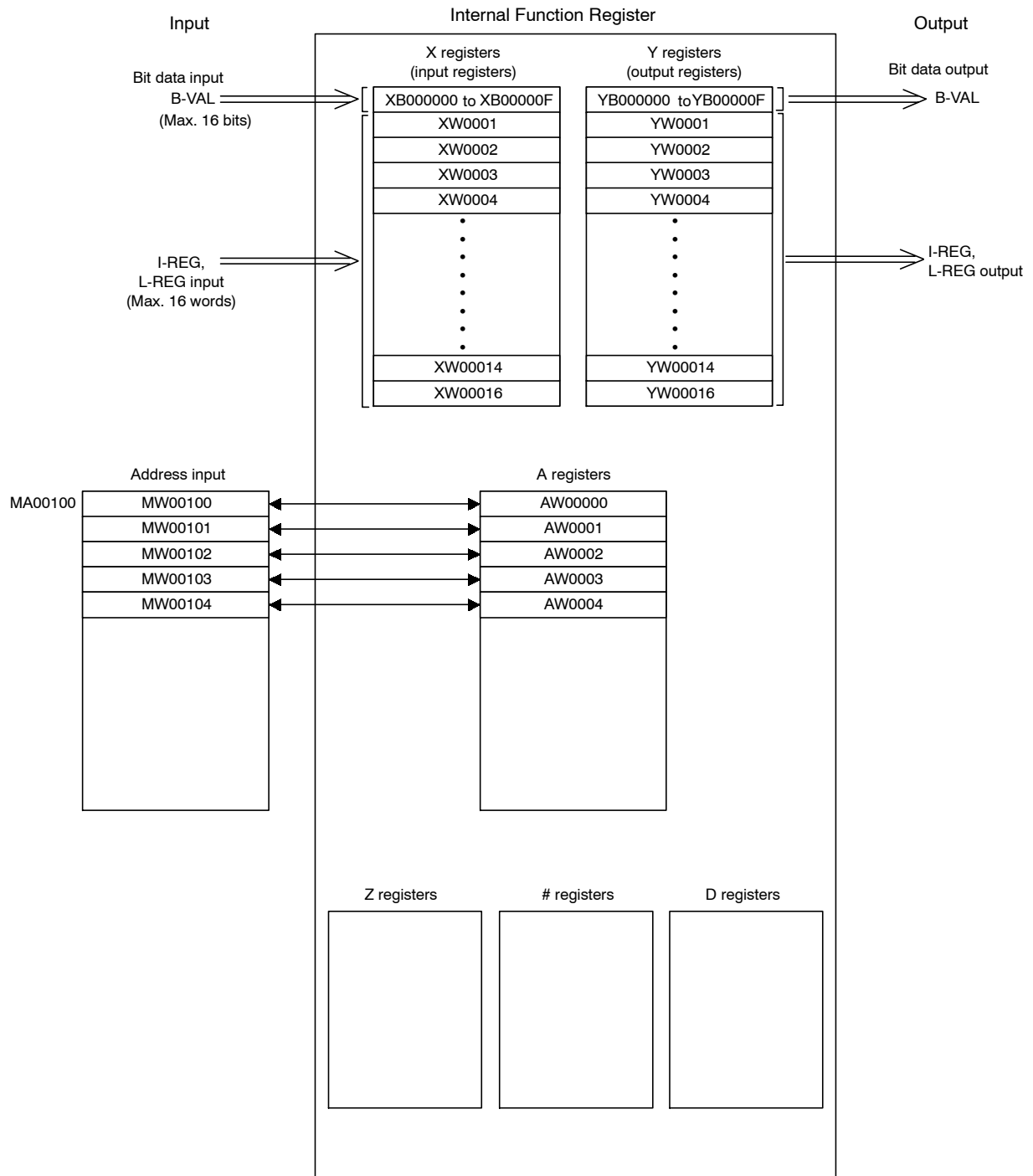
## ■ Registers Used Within User Functions

The data types are as follows:

- B-VAL ⇒ Bit
- I-VAL ⇒ For future use
- L-VAL ⇒
- I-REG ⇒ Integer
- L-REG ⇒ Double-length integer or real number

## ■ Relationship Between I/O Registers and Internal Function Registers

The correspondence between the I/O registers designated by the UFC command and the function registers is shown in the following illustration.



The S, M, I, O, and C registers can be used in the same way as they are in drawings.

The 11 types of register shown in the following table can be used in functions.

**Table 3.1 Function Registers**

Type	Name	Designation Method	Description	Characteristic
<b>X</b>	<b>Function Input Registers</b>	XB,XW,XL,XFnnnnn	Input to a function Bit input: XB000000 to XB0000F Integer input: XW00001 to XW00016 Double integer input: XL00001 to XL00015 Register No. nnnnn is a decimal expression.	Specific to each function
<b>Y</b>	<b>Function Output Registers</b>	YB,YW,YL,YFnnnnn	Input to a function Bit input: YB000000 to YB0000F Integer input: YW00001 to YW00016 Double integer input: YL00001 to YL00015 Register No. nnnnn is a decimal expression.	
<b>Z</b>	<b>Internal Function Registers</b>	ZB,ZW,ZL,ZFnnnnn	Internal registers unique to each function. Can be used by the function for internal processes. Register No. nnnnn is a decimal expression.	
<b>A</b>	<b>External Function Registers</b>	AB,AW,AL,AFnnnnn	External registers that use the address input value as the base address. For linking with S, M, I, O, #, and Dannnnn. Register No. nnnnn is a decimal expression.	
<b>#</b>	<b># Registers</b>	#B,#W,#L,#Fnnnnn (#Annnnn)	Registers that can be referenced only by a program. Can be referenced only by the corresponding drawing. The actual range used is specified by the user in the MPE720. Register No. nnnnn is a decimal expression.	
<b>D</b>	<b>D Registers</b>	DB,DW,DL,DFnnnnn (DAnnnnn)	Registers unique to each drawing. Can be referenced only by the corresponding drawing. The actual range used is specified by the user in the MPE720. Register No. nnnnn is a decimal expression.	
<b>S</b>	<b>System Register</b>	SB,SW,SL,SFnnnnn (SAnnnnn)	Same as the drawing registers. These registers are common to drawings and functions. Care is required in using them when the same function is referenced from drawings with different priority levels.	Common to drawings
<b>M</b>	<b>Data Register</b>	MB,MW,ML,MFnnnnn (MAnnnnn)		
<b>I</b>	<b>Input Register</b>	IB,IW,IL,IFhhhh (IAhhhh)		
<b>O</b>	<b>Output Register</b>	OB,OW,OL,Ofhhhh (OAhhhh)		
<b>C</b>	<b>Constant Register</b>	CB,CW,CL,CFhhhhh (CAnnnnn)		

SA, MA, IA, OA, DA, #A, and CA can also be used inside functions.

An example of the transfer of I/O registers is shown in the following illustration.

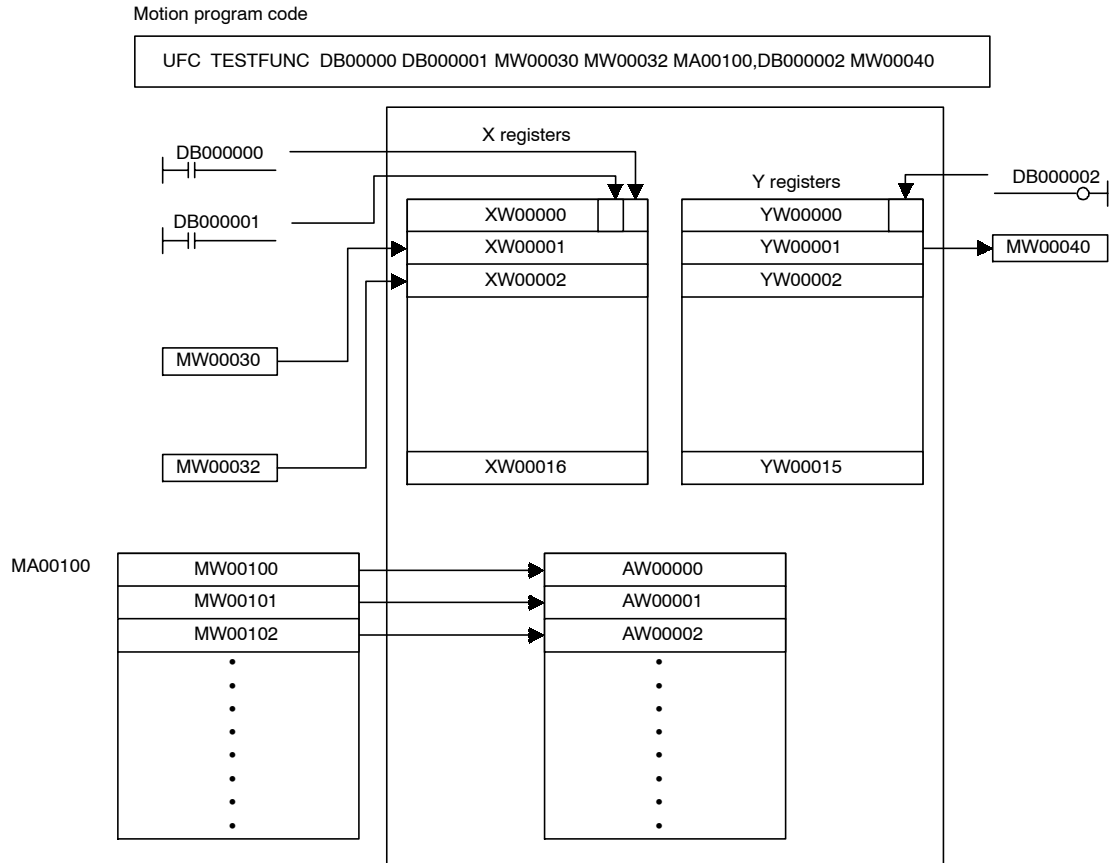


Figure 3.6 Motion Program Coding

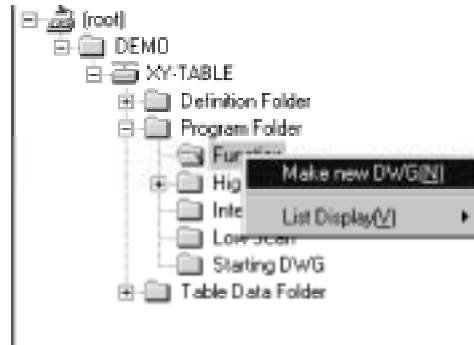
### ■ Creating User Functions

This section describes the procedure for creating user functions with an example using the following specifications for an MP930 Machine Controller. For MP2000-series Machine Controllers, the parameter addresses are different from the address shown in the following table.

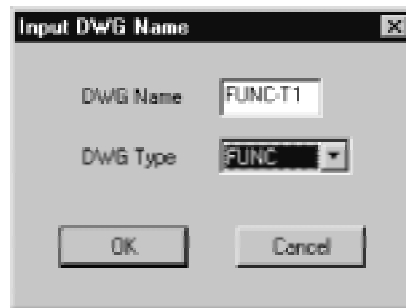
<b>Specifications</b>	Designate the servo axis No. and speed data, and set this in the Rapid Traverse Speed setting parameter (OLxx22).
<b>Motion Program</b>	MW00030 = Servo axis No. (1 or 2) ML00032 = Rapid traverse speed UFC FUNC-T1 MW00030 ML00032, DB000001;

The procedure for creating the user function is as follows:

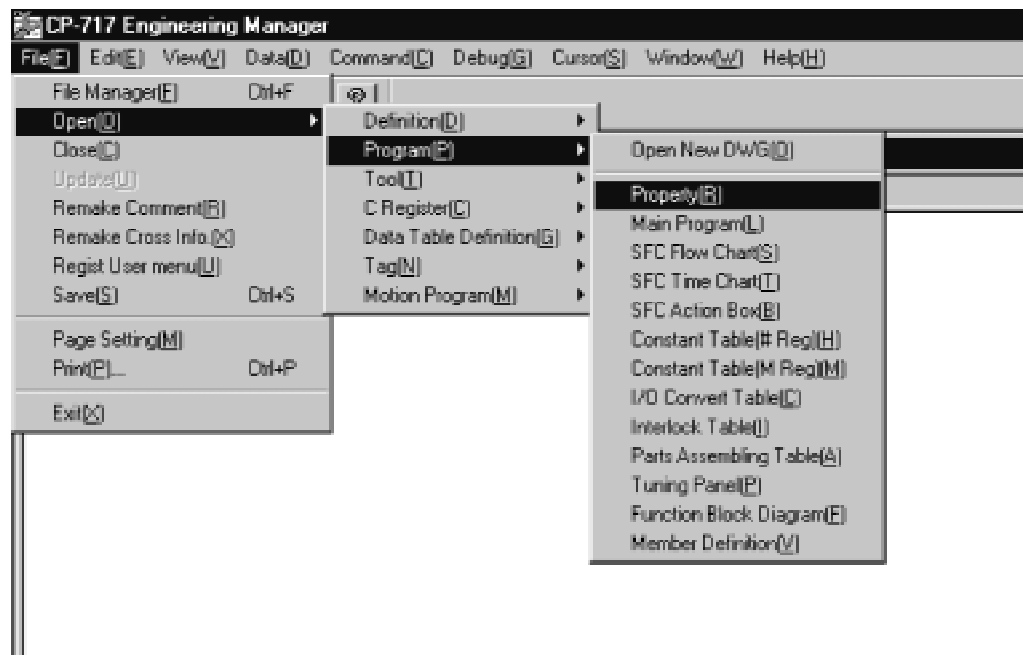
1. On the File Manager window, open **Program Folder** and then **Function**, right-click **Function** and then double-click **Make new DWG (N)**.



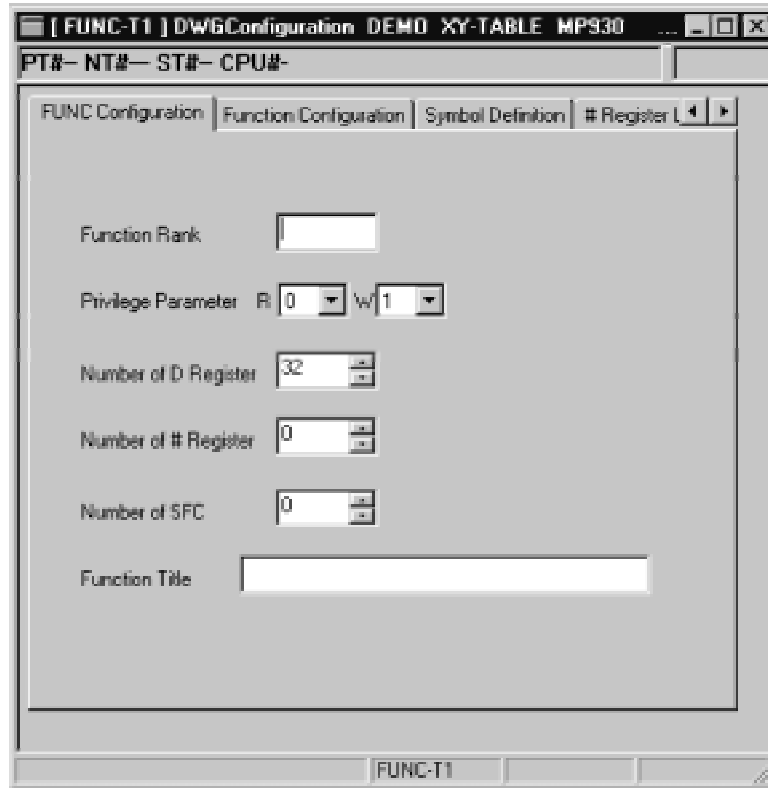
2. Input the drawing name and type of drawing in the Input DWG Name message box, and click the **OK** button.



3. The Ladder window (a white screen with no program) will be displayed. From the File (F) menu, select **Open (O)** → **Program (P)** → **Property (R)**, and open the DWG Properties window.

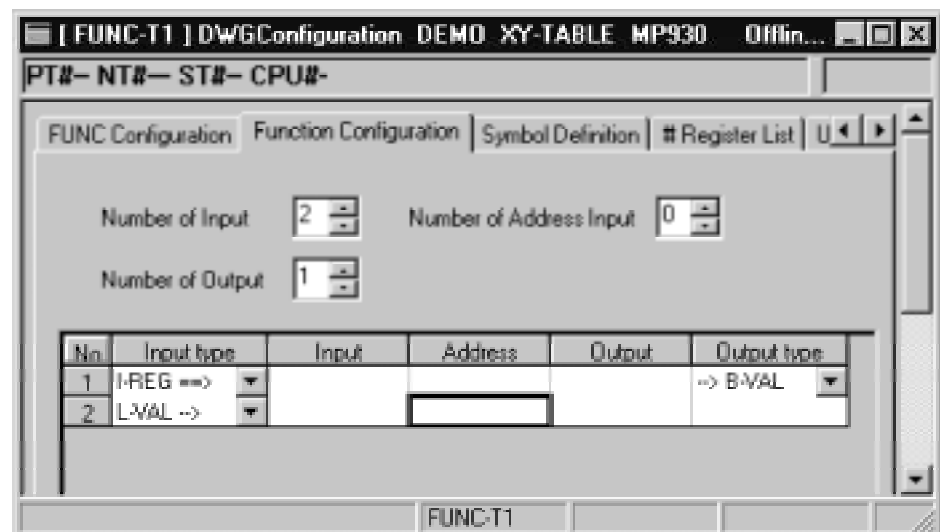


- Set the number of D registers in the FUNC Configuration Tag Page on the DWG Properties window. (The default value is 32 items.)



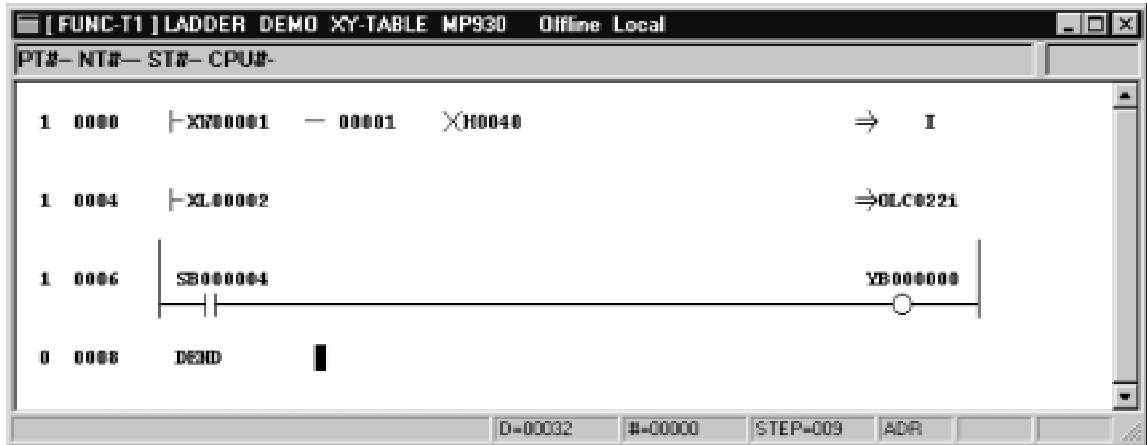
- Click the I/O Definition Tab Page, and set the number of I/O items for the function and the data type.

**Example:** UFC FUNC-T1 MW00030 ML00032,,DB000001;  
The following settings would be used for the above code.

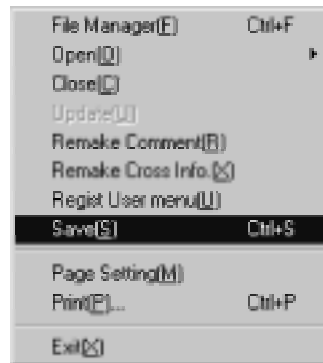




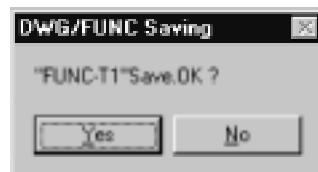
- Close the DWG Properties window, and edit the user function program on the Ladder window.



- Select *Save (S)* from the File (F) menu on the Ladder window.



- The DWG/FUNC Saving Dialog Box will be displayed. Click **Yes**.



- The Saving Completed Dialog Box will be displayed. Click **OK**.



- This ends the creation of the user function to be called from the motion program.

11. On the Motion Editor window, input the commands to call the user function.

```

MPM001 ***;
fmx t0000000;           "setup interpolation max. feed speed
mw0001=10000;          "interpolation speed override=100.0%
iac t300;              "time constant acceleration for interpolation
idc t500;              "time constant acceleration for interpolation
vel [x]6000 [y]5000;   "setup axis x,y feed speed
zrn [x]0 [y]0;         "home position return
mw30=1;                "servo No.=1
ml32=500;              "feed speed=500
ufc func:t1 mw30 ml32_db01; "user function
mov [x]100.0 [y]100.0; "positioning (100,100)
inc mvs [x]200.0 t5000000; "linear interpolation axis x +200.0(inc. mode)
mvs [y]200.0;          "axis-y +200.0 (inc. mode)
mvs [x]-200.0;         "axis-x -200.0 (inc. mode)
mvs [y]-200.0;         "axis-y -200.0 (inc. mode)
abs mvs [x]200.0 [y]200.0; "axis-x,y(200,200) (abs. mode)
pln [x] [y];           "circular interpolation plane
mcc [x]200.0 [y]200.0 u200.0 v300.0 t5000000; "circular interpolation
mvs [x]100.0 [y]100.0;
end;

```

12. Execute the motion program and check the operation.

### 3.1.5 I/O VARIABLE WAIT (IOW)

#### ■ Overview

The I/O VARIABLE WAIT (IOW) command causes execution to wait until the status specified by a conditional expression is satisfied. When the condition is satisfied, execution proceeds to the next block.

#### ■ Description

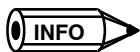
The IOW command is designated as follows:

```

IOW IB000001&IB000002= =|;
           A           B
A: Conditional expression
B: Status

```

The IOW command causes execution to wait until the conditional expression is satisfied, then ends the current block and proceeds to the next block.



- A conditional expression can be written only on the left side.
- With a bit expression, only = = can be used for the comparison command.  
The conditional expression cannot be used.  
IOWIB000001&IB000002< >0;
- Parentheses ( ) can also be used in a conditional expression.
- Any comparison command can be used if the conditional expression is not a bit expression.  
IOWMW00100+MW00101>1000;  
IOWML00100\*3/100>=20;  
IOWMW00100&MW00101vMW00102= =3355H;

## ■ Programming Examples

### ◀EXAMPLE▶

```
IOW MB001001&MB001002= =1;
MOV [axis1] 1000;
```

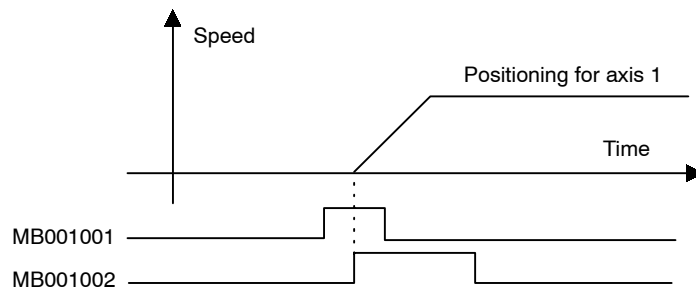


Figure 3.7 Programming Example for I/O VARIABLE WAIT (IOW)

## 3.1.6 SUBROUTINE CALL (MSEE)

### ■ Overview

The MSEE command can call from the motion program a subroutine that has previously been stored in motion program memory.

### ■ Description

The MSEE command is designated as follows:

```
MSEE MPS-;
    Call subroutine number
```

The MSEE command executes the subroutine with the number designated by MPS. There is no restriction on the nesting levels for calling subroutines from other subroutines.

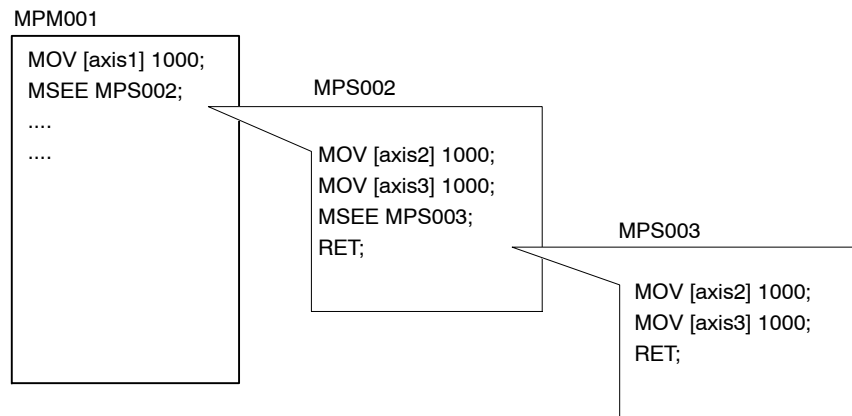


Figure 3.8 Subroutine

SUBROUTINE RETURN (RET) must be designated at the end of the subroutine.

**IMPORTANT**

Subroutine Restrictions

The following restrictions apply to the motion program coding within a subroutine. Write subroutines with care.

- A maximum of two parallel executions with the PFORK command.
- When the PFORK command is used within a subroutine, axis move commands can be coded on only one side.
- From 1 to 256 main programs (MPM□□□) and subroutines (MPS□□□) can be used. The same number cannot be used for both a MPM and MPS.
- If the program number of a main program is called by the MSEE command, the program will not be executed.

### 3.1.7 SUBROUTINE RETURN (RET)

3

■ **Overview**

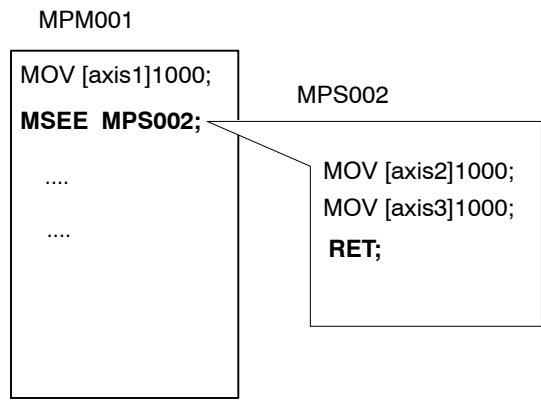
The SUBROUTINE RETURN (RET) command is designated at the end of a subroutine.

■ **Description**

The RET command is designated as follows:

```
RET;
Subroutine ended.
```

With the RET command, program execution proceeds to the next block of the program (main program or subroutine) after the SUBROUTINE CALL (MSEE) command that called the subroutine.



### 3.1.8 ONE SCAN WAIT (EOX)

#### ■ Overview

The continuous sequence commands in a motion program are normally executed in one scan. The EOX command is used to execute continuous sequence commands in several scans.

This command supports block operation, i.e., operation will stop for it in block operation mode.

#### ■ Designation Method

```
MW00001=100;
OB00010=1;
EOX;
OB00011=0;
```

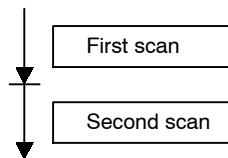
The block after EOX will be executed in the next scan.

#### ◀EXAMPLE▶

#### ■ Programming Examples

##### For Above Example

```
MW00001=100;
OB00010=1;
EOX;
OB00011=0;
```



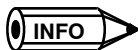
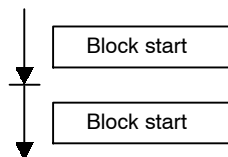
##### Example Using WHILE

```
WHILE OB00010==1;
EOX;
WEND;
```

A WHILE statement can be programmed only for sequence commands.

##### Sequence Command Debugging Example

```
EOX;
OB00010=1;
EOX;
OB00011=0;
```



#### Block operation

A motion program is executed in block units. The block operation function is provided for checking whether the program is running normally. Execution of one block at a time is possible using block operation mode and a block operation start signal.

Some commands can be stopped and some commands cannot be stopped with block operation.

- Commands that can be stopped: Commands for which processing is not completed in one scan (axis move commands such as MOV and MVS, including EOX)
- Commands that cannot be stopped: Commands for which processing is completed in one scan (sequence commands such as arithmetic commands, including PLD)

## 3.1.9 Branching Commands: IF...ELSE...IEND

### ■ Overview

The Branching commands (IF...ELSE...IEND) execute the block between IF and ELSE when a conditional expression is satisfied. If the conditional expression is not satisfied, the block between ELSE and IEND is executed.

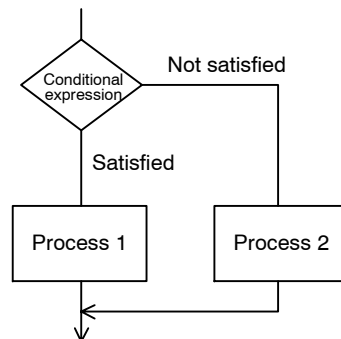
### ■ Description

The Branching commands (IF...ELSE...IEND) are designated as follows:

```
IF (conditional_expression);
... (process_1)
ELSE;
... (process_2)
IEND;
```

With branching commands, process 1 is executed if the conditional expression is satisfied, and process 2 is executed if the conditional expression is not satisfied.

- ELSE can be omitted. If it is omitted and the conditional expression is not satisfied, execution will continue from the block after IEND.



#### IMPORTANT

Nesting of the branching commands (IF...ELSE...IEND) is restricted to a maximum of eight levels.

#### EXAMPLE

### ■ Programming Examples

```
IF MB==1;
MOV [axis 1] 10000;    ← When MB is ON, positioning is executed for axis 1.
ELSE;
MOV [axis 2] 10000;    ← When MB is OFF, positioning is executed for axis 2.
IEND;
```

### 3.1.10 Repeat Commands: WHILE...WEND

#### ■ Overview

The repeat commands (WHILE...WEND) repeatedly execute a designated range of blocks for as long as the conditional expression is satisfied.

#### ■ Description

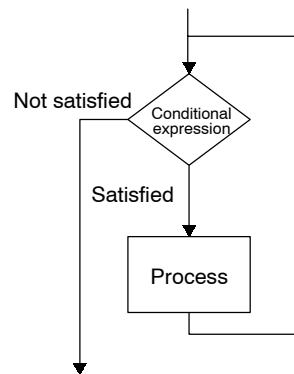
The repeat commands (WHILE...WEND) are designated as follows:

```

WHILE (conditional_expression);
...;
...; (process_1)
...;
WEND ;    ← End of REPEAT

```

With these commands, the blocks from WHILE to WEND are repeatedly executed as long as the conditional expression is satisfied. When the conditional expression is no longer satisfied, program execution will jump to the next block after WEND.



#### IMPORTANT

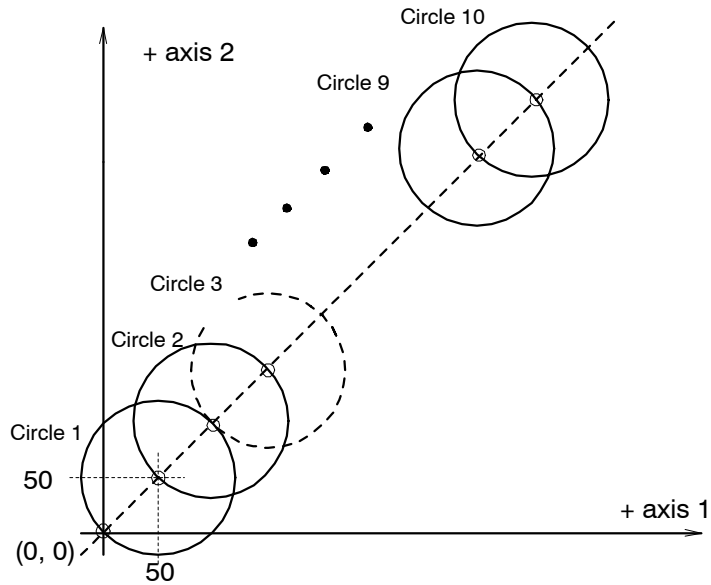
1. Nesting of the repeat commands (WHILE...WEND) is restricted to a maximum of eight levels.
2. If the repeated program section is created using only commands for which processing is completed in one scan, an infinite loop will be created and a watchdog timer error (system error) will be generated. When using a command for which processing is executed in one scan, be sure to enter the SCAN WAIT (EOX) command.
3. The commands for which processing is not completed in one scan are as follows:  
MOV, MOS, MCW/MCC, ZRN, SKP, MVT, EXM, ACC, SCC, PFN, VEL, INP, TIM, and IOW
4. Processing of all commands other than those listed above is completed in one scan.  
ABS, INC, IFP, PLN, IAC, IDC, FMX, POS, PLD, MSEE, END, RET, IF, WHILE, PFORK, SFORK, and all sequence commands

## ■ Programming Examples

### ◀EXAMPLE▶

In the following program, ten circles with a radius of 50 will be drawn.

MOV [axis1] 0 [axis2] 0;	←POSITIONING
MW00001 = 10000;	←INTERPOLATION OVERRIDE (100%)*
MW00100 = 1;	←Counter preset
INC;	←Incremental mode designation
PLN [axis1] [axis2];	←COORDINATE PLANE SETTING
WHILE MW0001 <= 10;	←Start of repeat programming
MCW [axis1] 0 [axis2] 0 U50. V50.F8000;	←CLOCKWISE CIRCULAR INTERPOLATION
MOV [axis1] 50. [axis2] 50.;	←POSITIONING
MW00100 = MW00100 + 1;	←Counter increment
WEND;	←End of repeat programming



\* Example of MP930

The interpolation override setting differs depending on the Machine Controller model. Refer to 1.2.3 *Feed Speeds* for details.

3



### 3.1.11 Parallel Execution Commands (PFORK, JOINTO, PJOINT)

#### ■ Overview

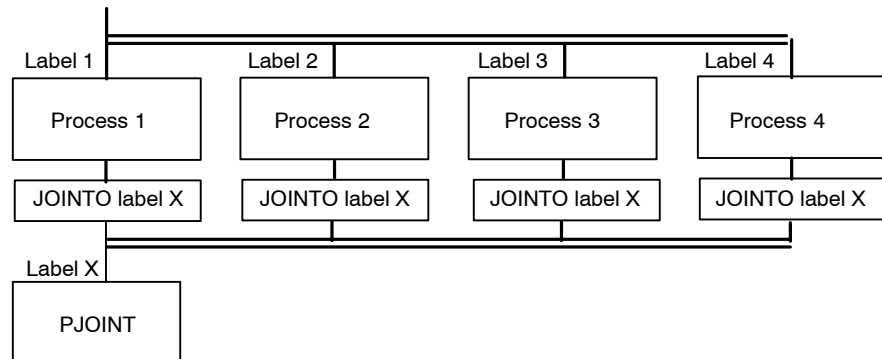
The PARALLEL FORK (PFORK) command performs parallel execution for blocks with the designated labels. After each parallel process has been executed, execution is merged at the label designated by the JOINTO command. A maximum of four parallel processes can be designated.

For further information on the labels, refer to *1.2.1 Programming Format*.

#### ■ Description

An example of the method used to designate the PARALLEL FORK (PFORK) command is shown in the following illustration.

```
PFORK Label 1 Label 2 Label 3 ...
Label 1: Process 1
        JOINTO Label X;
Label 2: Process 2
        JOINTO Label X;
Label 3: Process 3
        JOINTO Label X;
        :
        :
Label X: PJOINT
```



**Figure 3.9 Designating Parallel Execution Commands (PFORK, JOINTO, PJOINT)**

With the above commands, the labelled blocks (process 1, process 2, process 3, ...) designated by the PFORK command are executed in parallel. After each parallel process has been executed, execution is merged at the label designated by the JOINTO commands. A maximum of four parallel processes can be designated.

These commands make it possible to designate any combination of commands for parallel execution, such as axis move commands and sequence commands, or axis move commands and another axis move commands.

### Commands Designated Before PFORK

Values set by commands designated before the PFORK command, e.g., FMX, ABS/INC, F designation, IFP, PLN, IAC/IDC, are effective in processes executed in parallel for the parallel execution commands. Commands can also be used to set different values in each of the parallel processes. After merging, processing will continue using the values set in the the leftmost process.

### Parallel Execution Commands in Subroutines

The following restrictions apply to the parallel execution commands in subroutines:

- A maximum of two parallel processes are possible in a subroutine.
- For the MP900-series Machine Controllers, an axis move command can be coded only in the blocks designated by the first label.

```
PFORK 0002 0003;
0002:MVS [X]100.[Z]100.;
      JOINTO 0004;
0003:IOW MW10000==1;
      JOINTO 0004;
0004:PJOINT;
```

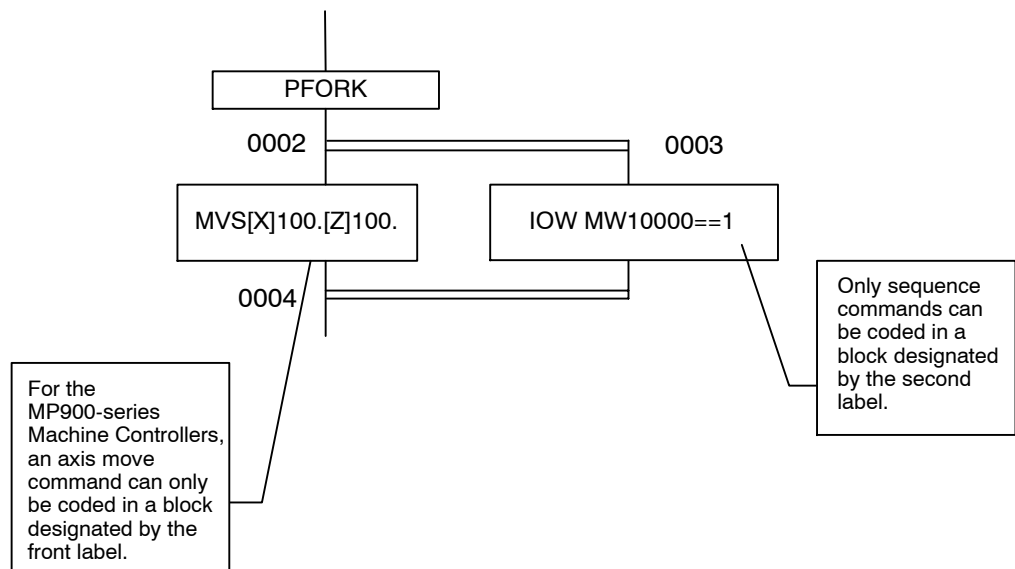


Figure 3.10 Parallel Execution Commands in Subroutines

**IMPORTANT**

1. If the same label is used more than once in a program, an error will result (“Duplicate labels are defined”).
2. If the number of PFORK branches and the number of labels are different, an error will result.

## ■ Programming Examples

### ◀EXAMPLE▶

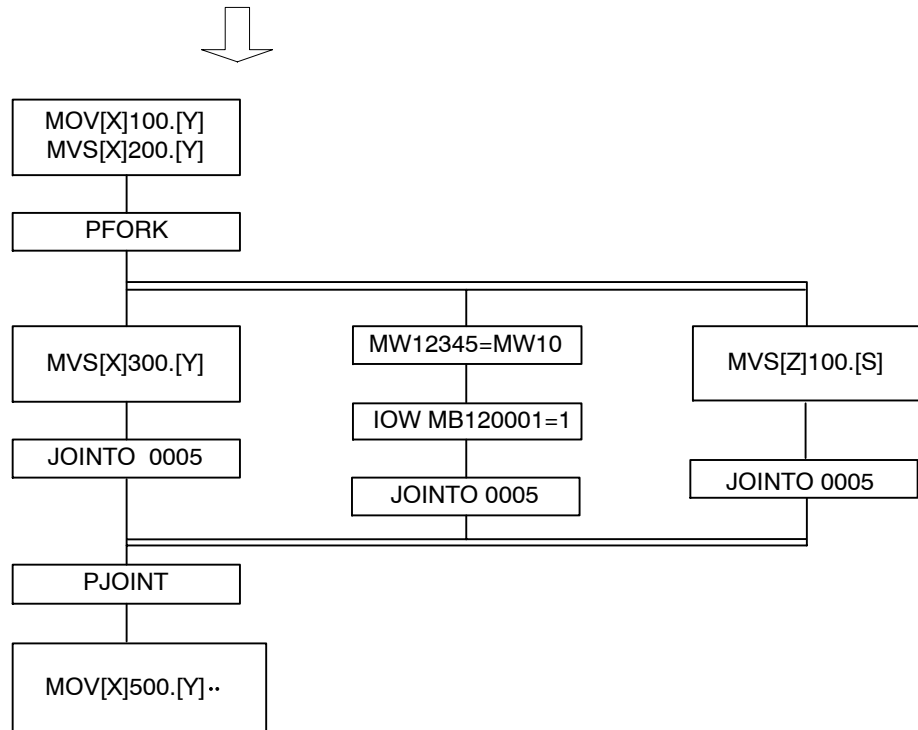
### Basic Pattern

A programming example for the basic pattern is shown in the following illustration.

```

0001: MOV[X]100.[Y]150.;
      MVS [X]200. [Y]250. F1000;
      PFORK 0002 0003 0004;
0002: MVS [X]300. [Z]100.
      JOINTO 0005;
0003: MW12345=MW10000+MW10002;
      IOW MB120001==1;
      JOINTO 0005;
0004: MVS [Z]100. [S]100. F3000;
      JOINTO 0005;
0005: PJOINT;
      MOV [X]500. [Y]500. [Z]500.;

```

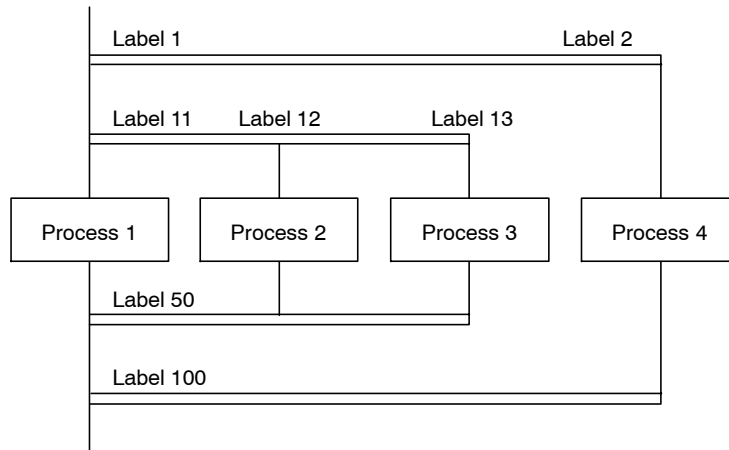


**Figure 3.11 Programming Example for Parallel Execution Commands (PFORK, JOINTO, PJOINT)**

### Application Pattern 1

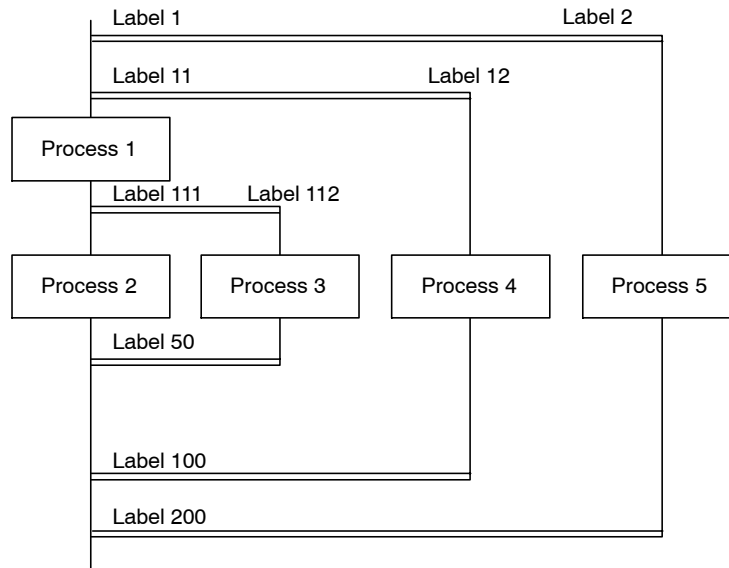
**EXAMPLE**

The following type of parallel execution is possible.



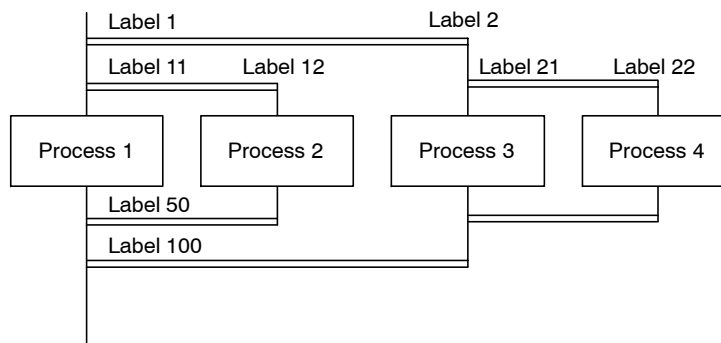
### Application Pattern 2

The following type of parallel execution is possible.



**IMPORTANT**

The following type of parallel execution is not possible.



### 3.1.12 Selective Execution Commands (SFORK, JOINTO, SJOINT)

#### ■ Overview

The selective execution commands execute labelled blocks following question marks (?) when the designated conditional expressions are satisfied.

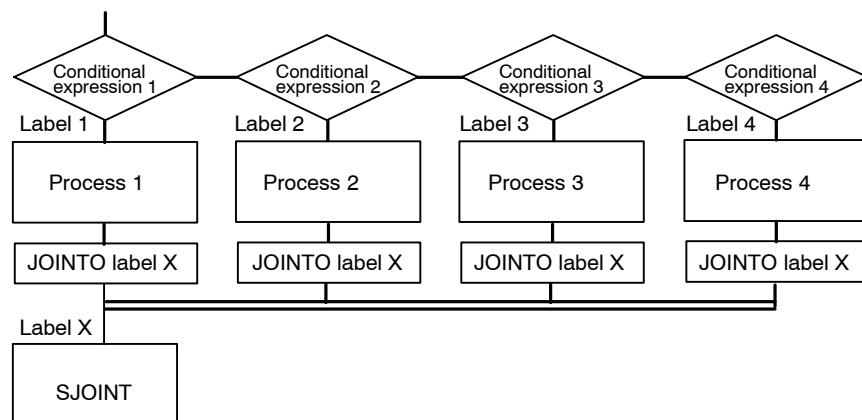
After each parallel process has been executed, execution is merged at the block with the label designated by the JOINTO commands.

Up to 16 conditional expressions can be designated.

#### ■ Description

An example of the method used to designate the SELECTIVE FORK (SFORK) command is shown in the following illustration.

```
SFORK Conditional expression 1 ? Label 1, Conditional expression 2 ? Label 2,
      Conditional expression 3 ? Label 3, Conditional expression 4 ? Label 4;
Label 1: Process 1
      JOINTO Label X
Label 2: Process 2
      JOINTO Label X
Label 3: Process 3
      JOINTO Label X
Label 4: Process 4
      JOINTO Label X
      .
      .
Label X: SJOINT
```



**Figure 3.12 Method of Designating Selection Execution Commands (SFORK, JOINTO, SJOINT)**

The above commands execute the labelled blocks when the conditional expressions designated by the SFORK command are satisfied. After each parallel process has been executed, execution merges at the label designated in the JOINTO commands.



1. The conditional expressions are examined in order from conditional expression 1. Even when more than one conditional expression is satisfied, processing is executed from the label that first satisfies the conditional expression.
2. Be sure to code conditions that will be satisfied. If a condition is not satisfied, processing will remain in wait status at the SFORK command block until the condition is satisfied.

### ■ Programming Examples

```

0001: MOV [X]100.[Y]150.;
      MVS [X]200.[Y]250.F1000;
      SFORK MW00100==1 ? 0002,MW00100==2 ? 0003,MW00100==3 ? 0004;
0002: MVS [X]300.[Z]100.F3000;
      JOINTO 0005
0003: MVS [X]300.[Z]100.F3000;
      JOINTO 0005
0004: MVS [Z]300.[S]100.F3000;
      JOINTO 0005
0005: SJOINT;
      MOV [X]500.[Y]500.[Z]500.;
    
```

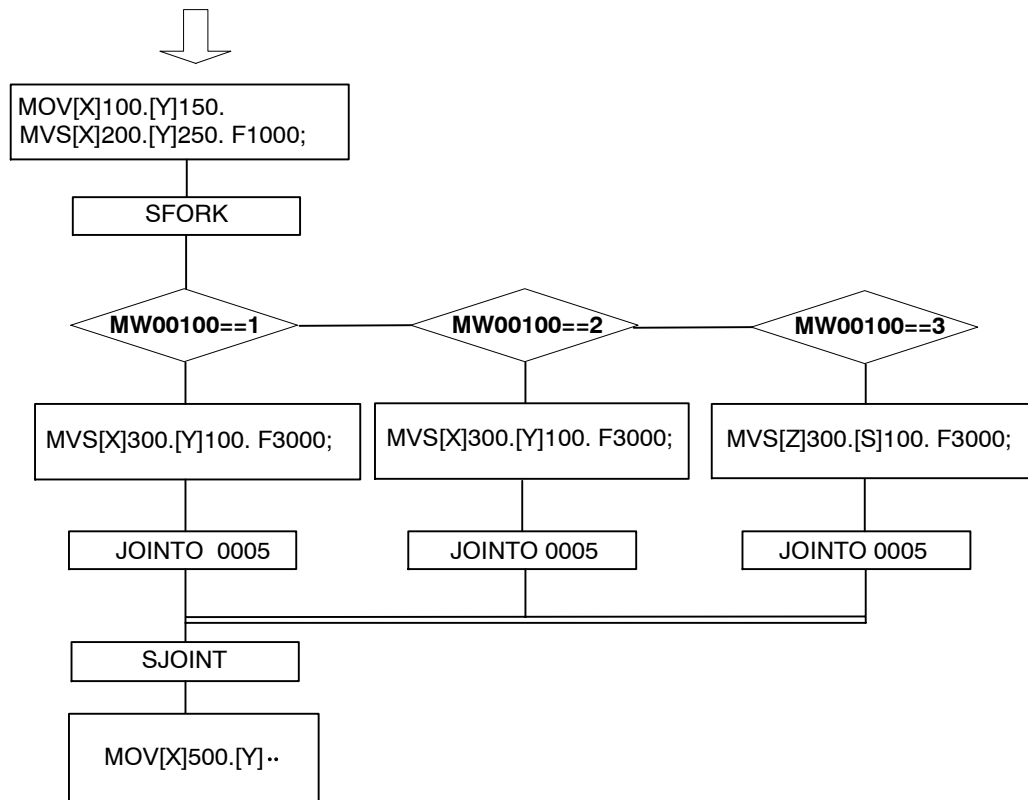


Figure 3.13 Programming Example for Selection Execution Commands (SFORK, JOINTO, SJOINT)

3

## 3.2 Speed and Acceleration/Deceleration Commands

This section explains the instructions for setting the speeds and accelerations/decelerations for axis move commands, and gives details of the programming methods.

### 3.2.1 ACCELERATION TIME CHANGE (ACC)

#### ■ Overview

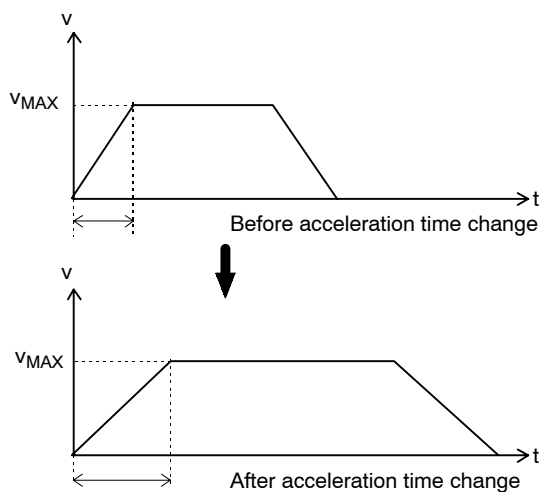
This command changes the acceleration times of axes for which positioning commands (MOV, MVT, EXM) are used.

#### ■ Description

The ACC command is designated as follows:

<code>ACC [axis1] - [axis2] - ...;</code> Acceleration time
--

The ACC command changes the acceleration time of each axis. The acceleration time of an unspecified axis is not changed. The acceleration time set by POSITIONING (MOV) command is changed.



**Figure 3.14 ACCELERATION TIME CHANGE (ACC)**

The acceleration time changed by ACC command execution remains in effect until it is reset by the next ACC command.

The ACC command reference range is as follows:

1 to 32,767 [ms]

**Note** The setting unit is [ms] for MP900-series Machine Controllers. That for MP2000-series Machine Controllers depends on the setting of OWxx03 bit 4 to bit 7 (Acceleration/Deceleration Unit Selection).



1. The setting of the setting parameter OWxx0C/OWxx36 (Linear Acceleration Time Setting) for each axis is changed by the ACC command.
2. For the MP920 (SVA-01, -02) Module, OWxx0C = xxxx can also be programmed in the programming instead of using the ACC command.
3. For the MP930, set the acceleration time to reach the feed speed set by the VEL command. For the Machine Controllers other than the MP930, set the time to reach the rated speed.
4. If using a SGD-□□□N or SGDB-□□AN SERVOPACK, the ACC command changes the acceleration/deceleration times.

## ■ Programming Examples

### ◀EXAMPLE▶

```

INC;
VEL [axis1] 6000;      Feed speed setting
ACC [axis1] 1000;     Acceleration time 1 second
MOV [axis1] 100000;   Positioning
ACC [axis1] 500;      Acceleration time 0.5 seconds
MOV [axis1] 50000;

```

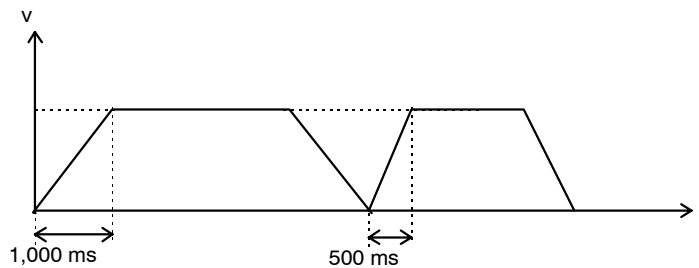


Figure 3.15 Programming Example for ACCELERATION TIME CHANGE (ACC)



### 3.2.2 DECELERATION TIME CHANGE (DCC)

#### ■ Overview

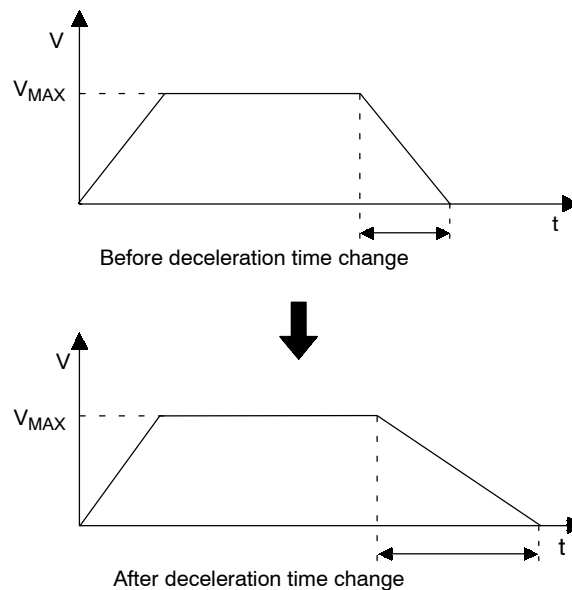
This command changes the deceleration time of axes for which positioning commands (MOV, MVT, EXM) are used.

#### ■ Description

The DCC command is designated as follows:

```
DCC [axis1] - [axis2] - ...;
    Deceleration time
```

The DCC command is used to change the deceleration time of each axis. The deceleration time of an unspecified axis remains unchanged. Only the deceleration time specified by the POSITIONING (MOV) command is changed.



**Figure 3.16 DECELERATION TIME CHANGE (DCC)**

The deceleration time changed by the DCC command execution remains in effect until it is reset by the next DCC command.

Reference range for the DCC command: 1 to 32,767 [ms]

**Note** The setting unit is [ms] for MP900-series Machine Controllers. That for MP2000-series Machine Controllers depends on the setting of OWxx03 bit 4 to bit 7 (Acceleration/Deceleration Unit Selection).



1. The setting of the parameter OWxx0D/OWxx38 (Linear Deceleration Time Setting) for each axis is changed by the DCC command.
2. For the MP920 (SVA-01, -02) Module, Owxx0D = xxxx can also be programmed in the programming instead of using the DCC command.
3. For the MP930, set the time to decelerate from the feed speed set by the VEL command to the time that the reference distribution is completed. For the Machine Controllers other than the MP930, set the time to decelerate from the rated speed to the time that the reference distribution is completed.
4. If using a SGD-□□□N or SGDB-□□AN SERVOPACK, the DCC command cannot be used.

### ■ Programming Example

◀EXAMPLE▶

```

INC;
VEL [axis1]6000;           Feed speed setting
DCC [axis1]1000;         Deceleration time 1 second
MOV [axis1]100000;       Positioning
DCC [axis1]500;          Deceleration time 0.5 seconds
MOV [axis1]50000;
    
```

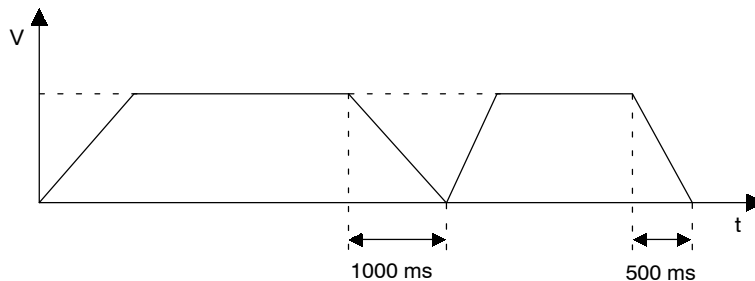


Figure 3.17 Programming Example of DECELERATION TIME CHANGE (DCC)

## 3.2.3 S-CURVE TIME CONSTANT CHANGE (SCC)

### ■ Overview

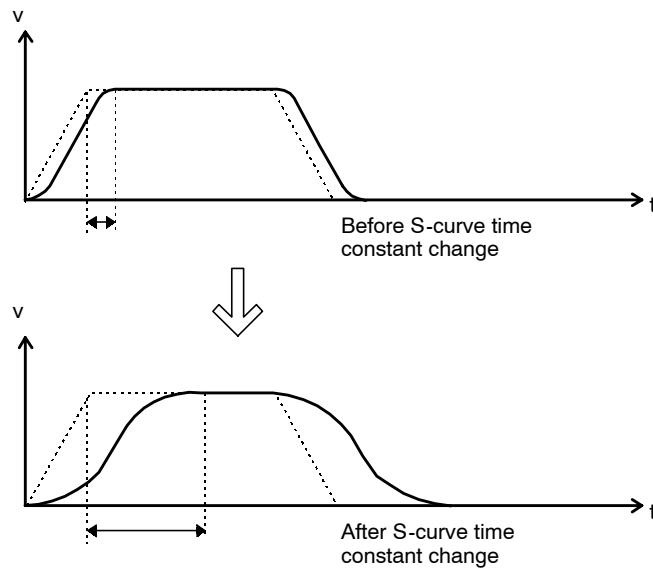
This command sets the S-curve acceleration/deceleration function parameter that controls vibrations of the machine system during acceleration and deceleration. It changes the S-curve time constant of each axis.

### ■ Description

The SCC command is designated as follows:

SCC [axis1] - [axis2] -... ; S-curve time constant
---

The SCC command changes the S-curve time constant of each axis. The S-curve time constant of an unspecified axis is not changed.



**Figure 3.18 S-CURVE TIME CONSTANT CHANGE (SCC)**

The SCC command reference range is as follows:

1 to 510 [ms]



1. When the SCC command is not executed, the S-curve time constant is determined by the S-curve Time Constant setting parameter.
2. The S-curve time constant changed by SCC command execution remains in effect until it is reset by the next SCC command.

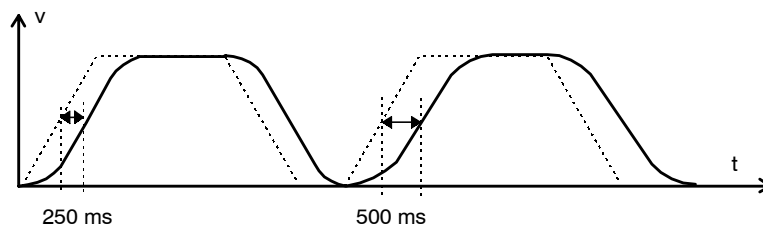
**IMPORTANT**

Do not use the SCC command with the SVA Modules.

**■ Programming Examples**

**◀EXAMPLE▶**

```
INC;
SCC [axis1] 250;
MOV [axis1] 100000;
SCC [axis1] 500;
MOV [axis1] 100000;
```



**Figure 3.19 Programming Example for S-CURVE TIME CONSTANT CHANGE (SCC)**

**IMPORTANT****■ How to validate S-curve acceleration/deceleration**

In order to validate the S-curve acceleration/deceleration, set the filter type.  
Change the filter type according to the following program.

- MP-900 series Machine Controllers

OWC021 = OWC021 & FF0FH ;

OWC021 = OWC021 + 0020H ;

IOW OWC020 == 0 ;

OWC020 = 13 ;

IOW IWC014 == 13 ;

OWC020 = 0 ;

IOW IWC014 == 0 ;

Filter type = selection of average moving filter

No motion command?

Motion command 13 = changing filter type

Waiting for command response

NOP command

- MP2000-series Machine Controllers

OW8003 & F0FH

OW8003 = OW8003 + 0200H;

IOW OW8008 == 0;

OW8003 = 13;

IOW IW8008 == 13;

IOW IB80098 == 1;

OW8008 = 0;

IOW IW 8008 == 0

Filter type = selection of average moving filter

No motion command?

Motion command 13 = changing filter type

Waiting for command response

Completion of command execution

NOP command

Execute the program as shown above for all the axes which performs S-curve acceleration/deceleration.

### 3.2.4 SET SPEED (VEL)

#### ■ Overview

The VEL command changes the feed speeds of the axes for which positioning commands (MOV, MVT, EXM) are used.

#### ■ Description

The VEL command is designated as follows:

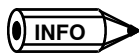
VEL <u>[axis1] - [axis2] - ... ;</u> Feed speed
--

The VEL command changes the feed speed of each axis. The feed speed of an unspecified axis is not changed.

The VEL command reference range is as follows:

0 to  $2^{31} - 1$  [ $10^n$  reference unit/min]  
 n = number of digits after the decimal point

**Note** The setting unit is [ $10^n$  reference unit/min] for MP900-series Machine Controllers. That for MP2000-series Machine Controllers depends on the setting of OWxx03 bit 0 to bit 3 (Acceleration/Deceleration Unit Selection).



1. When the VEL command is not executed, the feed speed is determined by the Feed Speed setting parameter for each axis, or by the speed set by the previous VEL command.
2. Speeds changed by VEL command execution remain in effect until reset by the next VEL command, or until an MVT command is executed.
3. The VEL command cannot be designated in the same block as the MOV command.
4. The setting parameter (Rapid Traverse Speed (OLxx22/OLxx10)) for each axis is changed by the VEL command. Therefore, OLxx22 = xxxxx can also be programmed in place of the VEL command for the MP900-series Machine Controllers.

#### Typical Setting for MP930

#### ◀EXAMPLE▶

- Motor rated r/min:  $3000 \text{ min}^{-1}$
- Fixed parameter setting No.17: in mm  
                                   No.18: Three digits below decimal point  
                                   No.19: Machine one rotation  $10000 = 10.000 \text{ mm}$

In the above case, to command the motor rated  $\text{min}^{-1}$  with the VEL command, the following equation can be established.

$$3000 \text{ min}^{-1} \times 10.000 \text{ mm} = 30000 \text{ mm/min}$$

Therefore, axis 1 becomes the rated  $\text{min}^{-1}$  with the following command:

VEL [axis 1] 30000;

■ Programming Examples

```

INC;
VEL [axis1] 500 [axis2] 500;
MOV [axis1] 1000 [axis2] 1000;
VEL [axis1] 1000;
MOV [axis1] 1000 [axis2] 1000;
    
```

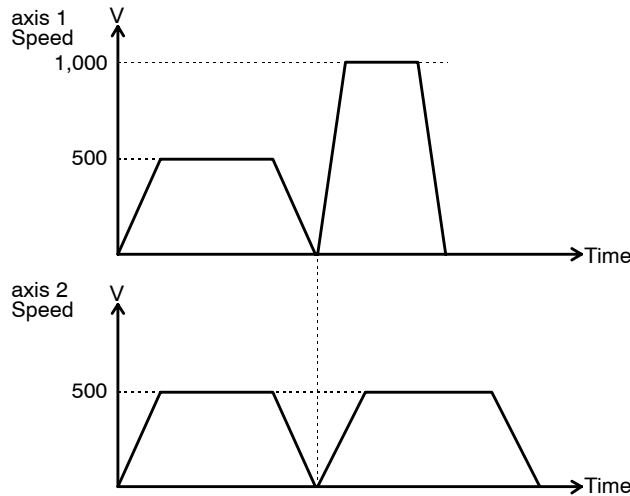


Figure 3.20 Programming Example for SET SPEED (VEL)

3.2.5 INTERPOLATION FEED SPEED RATIO SETTING (IFP)

■ Overview

The IFP command sets a percentage of the maximum interpolation feed speed (FMX) as the interpolation feed speed.

■ Description

The IFP command is designated as follows:

<pre> IFP P-;     Speed reference value (%)         </pre>
--

The IFP command sets the specified percentage of the maximum interpolation feed speed (FMX) as the interpolation feed speed.

The IFP command reference range is as follows:

1 to 100 [%]

**IMPORTANT**

1. In a motion program that uses interpolation commands, be sure to designate the FMX command at the beginning of the program to set the maximum interpolation feed speed. If FMX is not designated and the F designation or IFP command is specified, an error will result.
2. The IFP command cannot be designated in the same block as the interpolation command.
3. When the interpolation feed speed is designated by the IFP command (when the F designation is not used), designate the maximum interpolation feed speed in the FMX command before executing the interpolation command.
4. If the IFP command is executed after the F designation, the interpolation feed speed designated by F will be cancelled.
5. The speed designated by the IFP command remains in effect until it is reset by the next IFP command or by an F designation.

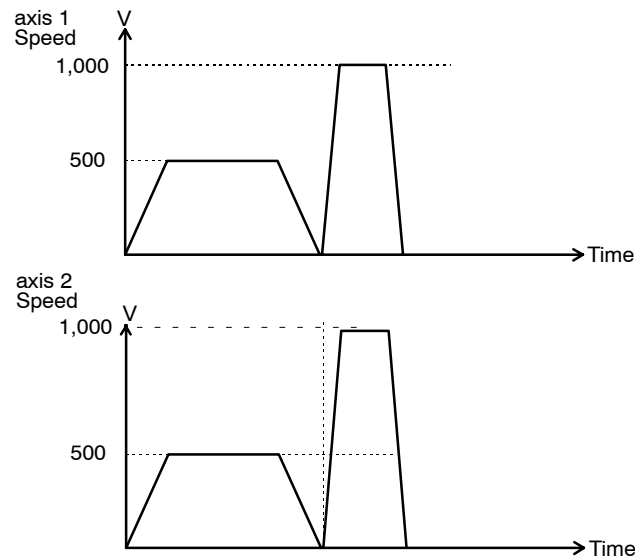
### ■ Programming Examples

#### ◀EXAMPLE▶

```

INC;
FMX T1000;
IFP P50
MVS [axis1] 1000 [axis2] 1000;
IFP P100
MVS [axis1] 1000 [axis2] 1000;

```



**Figure 3.21 Programming Example for INTERPOLATION FEED SPEED RATIO SETTING (IFP)**

## 3.2.6 MAXIMUM INTERPOLATION FEED SPEED (FMX)

### ■ Overview

The FMX command sets the maximum speed used when the interpolation speed commands (MVS, MCC/MCW, SKP) are executed. The acceleration/deceleration time set by the IAC and IDC commands is determined by this speed.

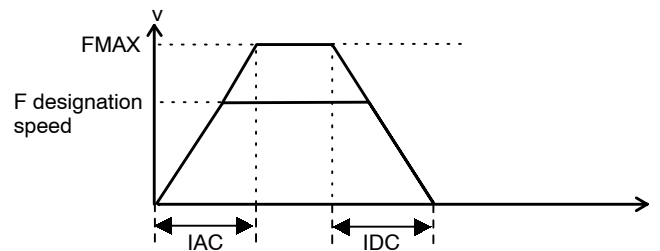
### ■ Description

The FMX command is designated as follows:

FMX T-;  
Maximum interpolation feed speed

The FMX command sets the maximum interpolation feed speed. When creating a motion program that uses interpolation commands, be sure to designate this command at the beginning of the program. Once it has been set, it remains in effect until it is reset.

- The interpolation acceleration/deceleration time set by the IAC and IDC commands will be the acceleration time until FMX is reached, and the deceleration time from FMX.



The INTERPOLATION FEED SPEED RATIO SETTING (IFP) command sets a percentage of the maximum speed designated by FMX as the interpolation feed speed.

- The FMX command reference range is as follows:

1 to  $2^{31} - 1$  [reference units/min]

#### ◀EXAMPLE▶

Typical Setting

- Motor rated r/min:  $3000 \text{ min}^{-1}$
- Fixed parameter setting No.17: in mm  
No.18: Three digits below decimal point  
No.19: Machine one rotation  $10000 = 10.000 \text{ mm}$

In the above case, in order to rotate the one-axis motor at the rated r/min with the FMX command, the following equation can be obtained.

$$3000 \times 10.000 \times 10^3 = 30000000.$$

↑ Number of digits below decimal point

Therefore, axis 1 becomes the rated  $\text{min}^{-1}$  with the following command:

FMX T30000000;

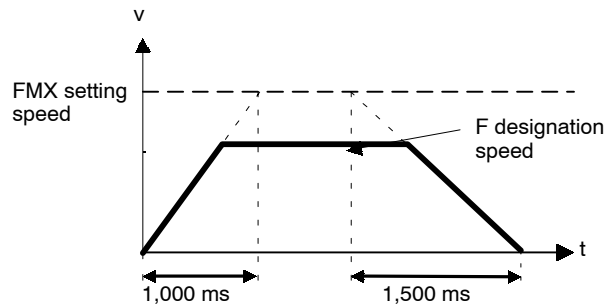


## ■ Programming Examples

```

INC;
FMX T1000000;
IAC T1000;
IDC T1500;
MVS [axis1] 100000 F700000;

```



**Figure 3.22 Programming Example for MAXIMUM INTERPOLATION FEED SPEED SETTING (FMX)**

### IMPORTANT

1. When FMX is not designated in a motion program that uses interpolation commands, an alarm will be generated. Program carefully.
2. A decimal point cannot be used directly in the FMX designation. The setting is made by adding the required number of zeroes after the decimal point.

**Example:** Number of digits after the decimal point = 3

Reference unit = mm

Maximum interpolation speed is 50000.000 mm ...

To set the above, input as follows:

```
FMX T50000000;
```

### 3.2.7 INTERPOLATION ACCELERATION TIME CHANGE (IAC)

#### ■ Overview

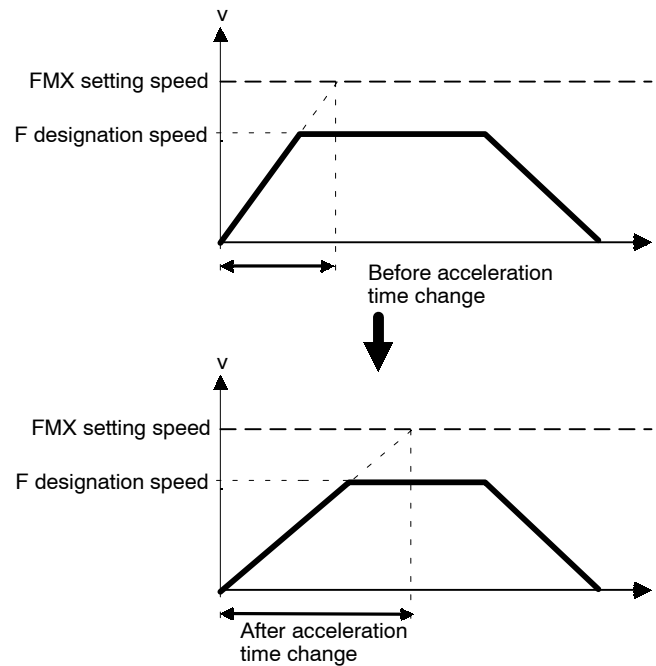
The IAC command sets the acceleration time when the interpolation speed commands (MVS, MCC/MCW, SKP) are executed. Any setting can be used in a motion program.

#### ■ Description

The IAC command is designated as follows:

```
IAC Tn;
Acceleration time
```

The interpolation acceleration time designated by the IAC command is the time until the maximum interpolation feed speed designated by the FMX command is reached.



**Figure 3.23 INTERPOLATION ACCELERATION TIME CHANGE (IAC)**

The IAC command reference range is as follows:

0 to 32,767 [ms]



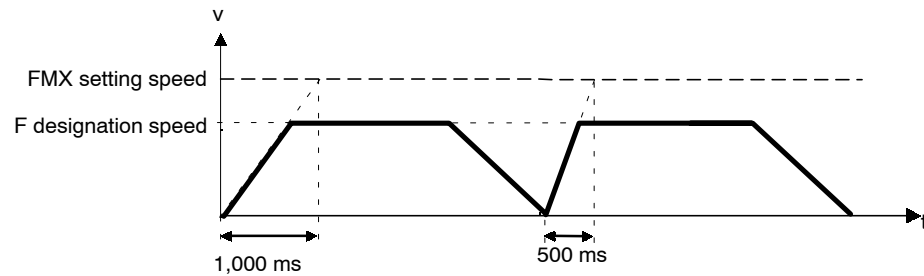
1. When the IAC command is not executed, the acceleration time will be 0 (rectangle).
2. The acceleration time changed by IAC command execution remains in effect until it is reset by the next IAC command.

## ■ Programming Examples

```

FMX T10000;
INC;
IAC T1000;
MVS [axis1] 100000 F70000;
IAC T500;
MVS [axis1] 100000;

```



**Figure 3.24 Programming Example for INTERPOLATION ACCELERATION TIME CHANGE (IAC)**

### IMPORTANT

Before designating the IAC command, set the maximum interpolation feed speed in the FMX command. If the IAC command is designated without first designating the FMX command, an error will result.

### 3.2.8 INTERPOLATION DECELERATION TIME CHANGE (IDC)

#### ■ Overview

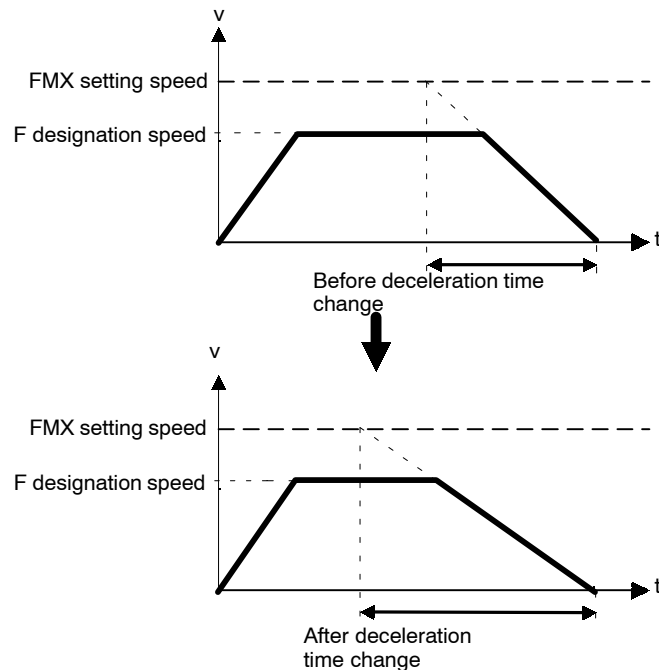
The IDC command sets the deceleration time when the interpolation speed commands (MVS, MCC/MCW, SKP) are executed. Any setting can be used in a motion program.

#### ■ Description

The IDC command is designated as follows:

IDC T<sub>n</sub>;  
Deceleration time

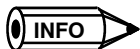
The interpolation deceleration time designated by the IDC command is the time from the speed designated by the FMX command until the axis stops.



**Figure 3.25 INTERPOLATION DECELERATION TIME CHANGE (IDC)**

The IDC command reference range is as follows:

0 to 32,767 [ms]



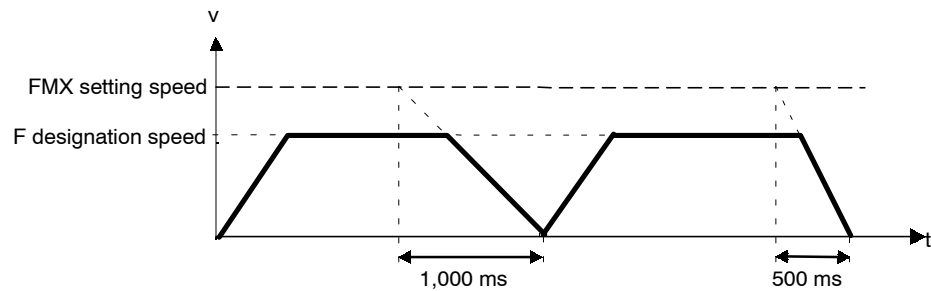
1. When the IDC command is not executed, the deceleration time constant will be 0 (rectangle).
2. The deceleration time changed by IDC command execution remains in effect until it is reset by the next IDC command.

## ■ Programming Examples

```

FMX T100000;
INC;
IDC T1000;
MVS [axis1] 100000 F70000;
IDC T500;
MVS [axis1] 100000;

```



**Figure 3.26 Programming Example for INTERPOLATION DECELERATION TIME CHANGE (IDC)**

### IMPORTANT

Before designating the IDC command, set the maximum interpolation feed speed in the FMX command. If the IDC command is designated without first designating the FMX command, an error will result.

# 4

---

## Sequence Commands

This chapter explains the sequence commands, such as the math commands, and describes the programming methods used.

<b>4.1 Overview of Sequence Commands</b> .....	<b>4 -3</b>
4.1.1 Math Commands .....	4 -3
4.1.2 Combinations of Math Operations .....	4 -4
4.1.3 Combinations of Logic Operations .....	4 -4
<b>4.2 Arithmetic Operations</b> .....	<b>4 -6</b>
4.2.1 SUBSTITUTE (=) .....	4 -6
4.2.2 ADD (+) .....	4 -7
4.2.3 SUBTRACT (-) .....	4 -8
4.2.4 MULTIPLY (*) .....	4 -8
4.2.5 DIVIDE (/) .....	4 -9
4.2.6 REMAINDER (MOD) .....	4 -10
<b>4.3 Logic Operations</b> .....	<b>4 -11</b>
4.3.1 OR ( ) .....	4 -11
4.3.2 AND (&) .....	4 -12
4.3.3 XOR (^) .....	4 -13
4.3.4 NOT (!) .....	4 -14
<b>4.4 Data Comparisons</b> .....	<b>4 -16</b>
4.4.1 Data Comparison Commands (=, <, >, <=, >=, <=, >=) .....	4 -16
<b>4.5 Data Operations</b> .....	<b>4 -18</b>
4.5.1 BIT RIGHT SHIFT (SFR) .....	4 -18
4.5.2 BIT LEFT SHIFT (SFL) .....	4 -19
4.5.3 BLOCK MOVE (BLK) .....	4 -20
4.5.4 CLEAR (CLR) .....	4 -22

4.5.5 ASCII CONVERSION 1 (ASCII) .....	4 -23
<b>4.6 Basic Functions .....</b>	<b>4 -25</b>
4.6.1 SINE (SIN) .....	4 -25
4.6.2 COSINE (COS) .....	4 -26
4.6.3 TANGENT (TAN) .....	4 -28
4.6.4 ARC SINE (ASN) .....	4 -29
4.6.5 ARC COSINE (ACS) .....	4 -30
4.6.6 ARC TANGENT (ATN) .....	4 -31
4.6.7 SQUARE ROOT (SQT) .....	4 -32
4.6.8 BCD-TO-BINARY (BIN) .....	4 -34
4.6.9 BINARY-TO-BCD (BCD) .....	4 -35
4.6.10 SET BIT (S{ }) .....	4 -36
4.6.11 RESET BIT (R{ }) .....	4 -37

## 4.1 Overview of Sequence Commands

This section outlines the sequence commands and explains combining commands.

### 4.1.1 Math Commands

Global variables, local variables, and constants can be used to perform ordinary math operations in combination with operators and functions, and the results can be substituted in variables.

Calculations take the basic form of *variable* = *<math\_expression>*. The math expressions and functions listed in the following table can be used.

Classification	Command	Name	Programming Format
<b>Arithmetic Operations</b>	=	SUBSTITUTE	MW- = MW- ;
	+	ADD	MW- = MW- + MW- ;
	-	SUBTRACT	MW- = MW- - MW- ;
	*	MULTIPLY	MW- = MW- * MW- ;
	/	DIVIDE	MW- = MW- / MW- ;
	MOD	REMAINDER	MW- = MOD;
<b>Logic Operations</b>		OR (logical OR)	MB- = MB-   MB- ;
	^	XOR (logical exclusive OR)	MW- = MW- ^ MW- ;
	&	AND (logical AND)	MB- = MB- & MB- ;
	!	NOT (Inversion)	MB- = MB- & ! MB- ;
<b>Comparisons</b>	==	MATCH	IF MW- == MW- ;
	<>	MISMATCH	IF MW- <> MW- ;
	>	GREATER THAN	IF MW- > MW- ;
	<	LESS THAN	IF MW- < MW- ;
	>=	GREATER THAN OR EQUAL TO	IF MW- >= MW- ;
	<=	LESS THAN OR EQUAL TO	IF MW- <= MW- ;
<b>Data Operations</b>	SFR	RIGHT SHIFT	SFR MB- N- W- ;
	SFL	LEFT SHIFT	SFL MB- N- W- ;
	BLK	BLOCK MOVE	BLK MW- MW- W- ;
	CLR	CLEAR	CLR MB- W- ;



Classification	Command	Name	Programming Format
Basic Functions	SIN	SINE	SIN(MW- );
	COS	COSINE	COS(MW- );
	TAN	TANGENT	TAN(MW- );
	ASN	ARC SINE	ASN(MW- );
	ACS	ARC COSINE	ACS(MW- );
	ATN	ARC TANGENT	ATN(MW- );
	SQRT	SQUARE ROOT	SQRT(MW- );
	BIN	BCD-TO-BINARY	BIN(MW- );
	BCD	BINARY-TO-BCD	BCD(MW- );
	S{}	SET BIT	S{MB-} = MB- &MB- ;
	R{}	RESET BIT	R{MB-} = MB- &MB- ;

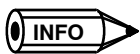
## 4.1.2 Combinations of Math Operations

Operations can be performed by combining math commands and functions. The order of priority of operations does not have to be considered. Perform the operations from left to right. To change the order, use parentheses.

The following example shows a math expression.

**Example:**  $MW00100 = MW00102 \pm MW00104 \frac{*}{1} \frac{*}{2} SIN(MW00106);$

The operation is performed in the order shown by the numbers.



The following examples show the changes in the operation order when parentheses ( ) are used.

- $MW00100 = 100 \pm 200 \frac{*}{1} \frac{*}{2} \frac{*}{3} SIN(30.);$   
(75.)
- $MW00100 = 100 \pm (200 \frac{*}{3} \frac{*}{1} \frac{*}{2} SIN(30.));$   
(150.)
- $MW00100 = 100 \pm (200 \frac{*}{2} \frac{*}{1} \frac{*}{3} SIN(30.));$   
(100.)

## 4.1.3 Combinations of Logic Operations

Logic operations can be performed by combining logic operation commands and functions.

There is no order of priority for operations. Although operations that use a combination of math operations and mixed expressions are also possible, real number operations cannot be performed.

The following example shows a logic expression.

**Example:**  $MW00100 = MW00102 \underset{1}{\&} MW00104 \underset{2}{|} MW00106 \underset{3}{\wedge} MW00108;$

The operation is performed in the order shown by the numbers.



The following examples show the changes in the operation order when parentheses ( ) are used.

- $MW00100 = MW00101 \underset{1}{\&} MW00102 \underset{2}{|} MW00103 \underset{3}{\wedge} MW00104;$
  - $MW00100 = MW00101 \underset{2}{\&} MW00102 \underset{3}{|} (MW00103 \underset{1}{\wedge} MW00104);$
-

## 4.2 Arithmetic Operations

This section explains the arithmetic operation commands.

### 4.2.1 SUBSTITUTE (=)

#### ■ Overview

The operation result on the right side of the expression is substituted in the register on the left side.

The operation order of priority is different for math operations and logic operations. For details, see *4.1.2 Combinations of Math Operations* and *4.1.3 Combinations of Logic Operations*.

#### ■ Description

##### Designation Method

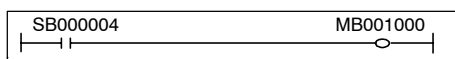
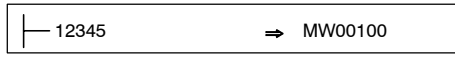
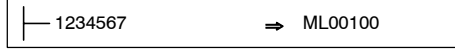

$$(Result) = (math\_expression);$$

##### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)
Yes	Yes	Yes	Yes

#### ■ Programming Examples

◀EXAMPLE▶

Type	Motion Program	Ladder Logic Program
<b>B</b>	MB001000=1;	
<b>W</b>	MW00100=12345;	
<b>L</b>	ML00100=1234567;	
<b>F</b>	MF00100=1.2345;	

## 4.2.2 ADD (+)

### ■ Overview

ADD (+) performs integer and real number addition on the right side and stores the result in the register on the left side. With mixed integers and real numbers, the data type on the left side is also stored.

### ■ Description

#### Designation Method

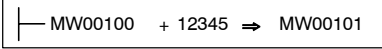
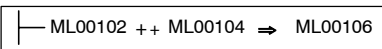
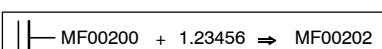
```
MW--=MW+MW-;
```

#### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)
No	Yes	Yes	Yes

#### ◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	-	-
<b>W</b>	MW00101=MW00100+12345;	
<b>L</b>	ML00106=ML00102+ML00104;	
<b>F</b>	MF00202=MF00200+1.23456;	

#### IMPORTANT

With an operation where the variables are of different data types, the result will be stored according to the data type on the left side. For details, see *Precautions on Variable Operations in 5.1.2 Global Variables and Local Variables*.

## 4.2.3 SUBTRACT (-)

### ■ Overview

SUBTRACT (-) performs integer and real number subtraction on the right side and stores the result in the register on the left side. With mixed integers and real numbers, the data type on the left side is stored.

### ■ Description

#### Designation Method

```
MW-=MW-- MW-;
```

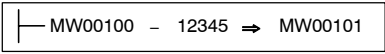
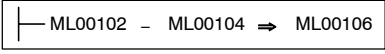
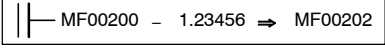
#### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)
No	Yes	Yes	Yes

4

#### ◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	-	-
<b>W</b>	MW00101=MW00100-12345;	
<b>L</b>	ML00106=ML00102-ML00104;	
<b>F</b>	MF00202=MF00200-1.23456;	

## 4.2.4 MULTIPLY (\*)

### ■ Overview

MULTIPLY (\*) performs integer and real number multiplication on the right side and stores the result in the register on the left side. With mixed integers and real numbers, the data type on the left side is stored.

### ■ Description

#### Designation Method

```
MW-=MW-- * MW-;
```

#### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)
No	Yes	Yes	Yes

## ◀EXAMPLE▶

## ■ Programming Examples

Type	Motion Program	Ladder Logic Program
B	-	-
W	MW00102=MW00100*MW00101;	
L	ML00106=ML00102*ML00104;	
F	MF00202=MF00200*1.23456;	

## 4.2.5 DIVIDE (/)

## ■ Overview

DIVIDE (/) performs integer and real number division on the right side and stores the result in the register on the left side. With mixed integers and real numbers, the data type on the left side is stored.

## ■ Description

## Designation Method

MW--MW- / MW-;
----------------

## Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)
No	Yes	Yes	Yes

## ◀EXAMPLE▶

## ■ Programming Examples

Type	Motion Program	Ladder Logic Program
B	-	-
W	MW00102=MW00100/MW00101;	
L	ML00106=ML00102/ML00104;	
F	MF00202=MF00200/1.23456;	

## 4.2.6 REMAINDER (MOD)

### ■ Overview

When specified in the next block after DIVIDE, MOD stores the remainder of the division in the specified variable. The remainder is stored as the data type on the left side, even when there are mixed integers and real numbers in the DIVIDE command block.

### ■ Description

#### Designation Method

MW00001=1000 / 999;	
MW00002=MOD;	→ MW0002 = 1

#### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)
No	Yes	Yes	No

4

#### ◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program				
<b>B</b>	-	-				
<b>W</b>	MW00101 = MW00100 / 3; MW00102 = MOD;	<table border="1"> <tr> <td>┌ MW00100 / 3</td> <td>⇒ MW00101</td> </tr> <tr> <td>└ MOD</td> <td>⇒ MW00102</td> </tr> </table>	┌ MW00100 / 3	⇒ MW00101	└ MOD	⇒ MW00102
┌ MW00100 / 3	⇒ MW00101					
└ MOD	⇒ MW00102					
<b>L</b>	ML00106 = ML00102 / ML00104; ML00108 = MOD;	<table border="1"> <tr> <td>┌ ML00102 / ML00104</td> <td>⇒ ML00106</td> </tr> <tr> <td>└ MOD</td> <td>⇒ ML00108</td> </tr> </table>	┌ ML00102 / ML00104	⇒ ML00106	└ MOD	⇒ ML00108
┌ ML00102 / ML00104	⇒ ML00106					
└ MOD	⇒ ML00108					
<b>F</b>	-	-				

#### Example: Double Integers

ML00106=ML00100 \* ML00102 / ML00104;  
 (173575) (100000) (60000) (34567)  
 ML00108 = MOD;  
 (32975)

#### IMPORTANT

The MOD command must be specified in the next block after DIVIDE . If it is not executed in the next block after DIVIDE, the operation result cannot be guaranteed.

## 4.3 Logic Operations

This section explains the commands used to perform bit and integer logic operations.

There is no order of priority for operations. Calculations are performed in order from the first item on the right side. Parts enclosed in parentheses are calculated first.

Although operations that combine math operations are also possible, real number operations cannot be performed.

### 4.3.1 OR (|)

#### ■ Overview

OR performs a logical OR for the immediately preceding operation result and the specified registers, and returns the operation result. Real number registers cannot be used.

**Table 4.1 Logical OR Truth Table (OR:C= A|B)**

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

#### ■ Description

##### Designation Method

```
MB001000=IB000100|SB000200;
MW00100=DW00102|AAAAH;
ML00104=DL00106|IL00100;
```


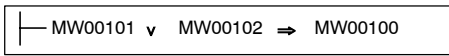
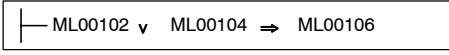
##### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
Yes	Yes	Yes	No	Yes



◀EXAMPLE▶

■ Programming Examples

Type	Motion Program	Ladder Logic Program
B	MB001000=MB001010 MB001011;	
W	MW00100=MW00101 MW00102;	
L	ML00106=ML00102 ML00104;	
F	-	-

4.3.2 AND (&)

4

■ Overview

AND performs a logical AND for the immediately preceding operation result and the specified registers, and returns the operation result. Real number registers cannot be used.

Table 4.2 Logical AND Truth Table (AND:C=A&B)

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

■ Description

Designation Method

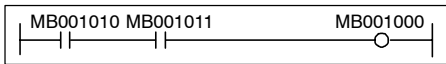
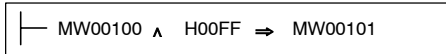
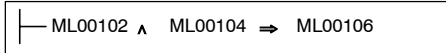
```
MB001000=IB001000&MB001010;
MW00100=MW00101&MW00102;
ML00100=ML00102&5555ACACH;
```

Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
Yes	Yes	Yes	No	Yes

◀EXAMPLE▶

## ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	MB001000=MB001010&MB001011;	
<b>W</b>	MW00101=MW00100&00FFH;	
<b>L</b>	ML00106=ML00102&ML00104;	
<b>F</b>	-	-

### 4.3.3 XOR (^)

#### ■ Overview

XOR performs an exclusive logical OR for the immediately preceding operation result and the specified registers, and returns the operation result. Real number registers cannot be used.

**Table 4.3 Exclusive OR Truth Table (XOR:C=A ^ B)**

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

#### ■ Description

##### Designation Method


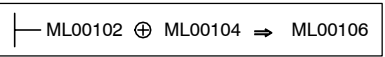
```
MW00100=MW00101^MW00102;
ML00100=ML00102^12345678H;
```

##### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
No	Yes	Yes	No	Yes

◀EXAMPLE▶

■ Programming Examples

Type	Motion Program	Ladder Logic Program
B		
W	MW00101=MW00100∧00FFH;	
L	ML00106=ML00102∧ML00104;	
F	-	-

4.3.4 NOT (!)

■ Overview

NOT inverts the data in the specified register and returns the operation result. Real number registers cannot be used.

■ Description

Designation Method

```
MB001000=!MW00101;
MW00100=!MW00101;
ML00100=!ML00102;
```

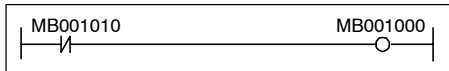


Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
Yes	Yes	Yes	No	Conditional*

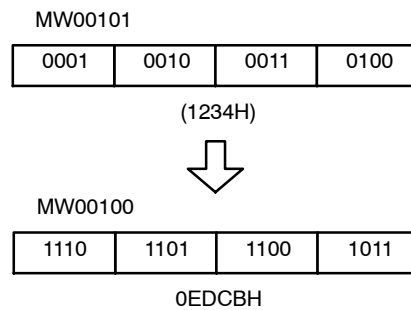
\* Bit constants cannot be specified.

◀EXAMPLE▶

## ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	MB001000=!MB001010;	
<b>W</b>	MW00100=!MW00101;	
<b>L</b>	ML00100=!ML00102	
<b>F</b>	-	-

**Example:** MW00100=!MW00101:



## 4.4 Data Comparisons

This section explains the data comparison commands that are used for conditional expressions.

### 4.4.1 Data Comparison Commands (=, <, >, <=, >=, <>)

#### ■ Overview

These commands are used to determine conditional expressions for commands such as branching commands (IF), repeat commands (WHILE), and I/O WAIT (IOW).

The following six comparison commands are provided.

Comparison Command	Meaning
==	MATCH
<>	MISMATCH
>	GREATER THAN
<	LESS THAN
>=	GREATER THAN OR EQUAL TO
<=	LESS THAN OR EQUAL TO

#### ■ Description

##### Designation Method


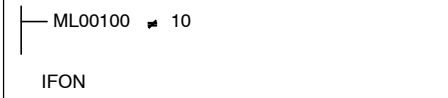
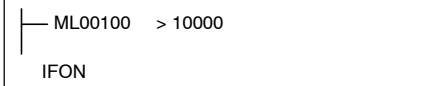

```
IF MW00100==MW00102+MW00104;
WHILE MB001001<>1;
```

##### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)
Yes*	Yes	Yes	Yes

\* Only the “=” command can be used in a bit conditional expression.

**EXAMPLE****■ Programming Examples**

Type	Motion Program	Ladder Logic Program
<b>B</b>	IF MB001000= =1;	
<b>W</b>	IF MW00100 < >10;	
<b>L</b>	IF ML00100 > 10000;	
<b>F</b>	IF MF00100 > = 3.0;	

**IMPORTANT**

With ladder logic commands, store the comparison results in bit registers. Use the IFON command to determine the comparison results.

## 4.5 Data Operations

This section explains the data commands that are used to shift, transfer, and initialize data.

### 4.5.1 BIT RIGHT SHIFT (SFR)

#### ■ Overview

The SFR command shifts a bit string designated by the leading bit number and bit width the specified number of shifts to the right.

#### ■ Description

##### Designation Method

```
SFR MB001000 N5 W10 ;
      A      B  C
```

A: Leading bit number  
B: Number to be shifted  
C: Bit width

##### Data Types

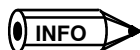
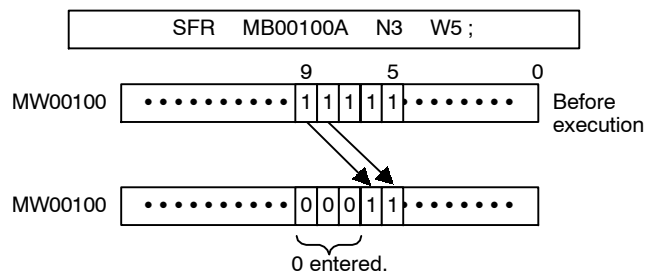
	Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
<b>Leading Bit Number</b>	Yes	No	No	No	No
<b>Number to be Shifted</b>	No	Yes	No	No	Yes
<b>Bit Width</b>	No	Yes	No	No	Yes

#### ◀EXAMPLE▶

#### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	–	–
<b>W</b>	SFR MB001000 N=5 W=10;	SHFTR MB001000 N=5 W=10
<b>L</b>	–	–
<b>F</b>	–	–

**Example:** Five bits with MB001005 (bit 5 of MW00100) as the leading bit are shifted three bits to the right.



With the SFR command, if the number of shifts is greater than the bit width, all data with the specified bit width will be set to 0.

## 4.5.2 BIT LEFT SHIFT (SFL)

### ■ Overview

The SFL command shifts a bit string designated by the leading bit number and bit width the specified number of shifts to the left.

### ■ Description

#### Designation Method

```
SFL MB001000 N5 W10 ;
      A      B  C
```

A: Leading bit number  
B: Number to be shifted  
C: Bit width

#### Data Types

	Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
<b>Leading Bit Number</b>	Yes	No	No	No	No
<b>Number to be Shifted</b>	No	Yes	No	No	Yes
<b>Bit Width</b>	No	Yes	No	No	Yes

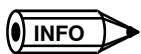
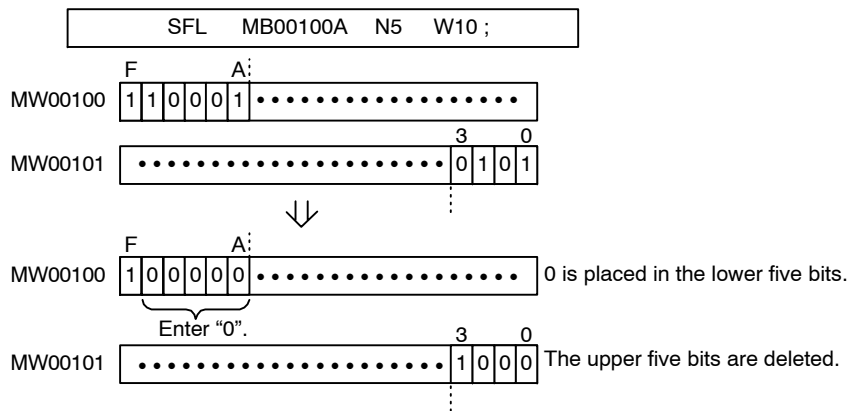


EXAMPLE

■ Programming Examples

Type	Motion Program	Ladder Logic Program
B	-	-
W	SFL MB001000 N=5 W=10	SHFTL MB001000 N=5 W=10
L	-	-
F	-	-

**Example:** The 10 bits with MB00100A (bit A of MW00100) as the leading bit are shifted five bits to the left.



With the SFL command, if the number of shifts is greater than the bit width, all data with the specified bit width will be set to 0.

4.5.3 BLOCK MOVE (BLK)

■ Overview

The BLK command moves the specified number of words from the beginning of the source register to the beginning of the destination register.

■ Description

Designation Method

```
BLK MW001000 N5 W10 ;
      A      B  C
```

A: Leading bit number  
 B: Number to be shifted  
 C: Bit width

### Data Types

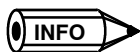
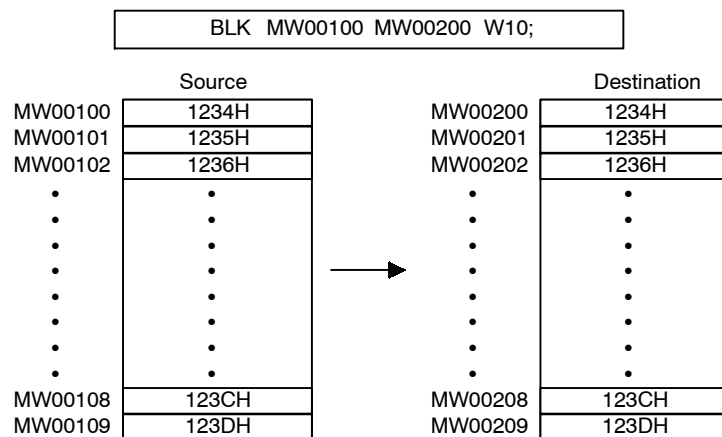
	Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
Source Leading Register	No	Yes	No	No	No
Destination Leading Register	No	Yes	No	No	No
Number of Blocks to be Moved	No	Yes	No	No	Yes

◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
B	-	-
W	BLK MW00100 DW00100 W10;	COPYW MW00100⇒DW00100 W=10
L	-	-
F	-	-

**Example:** MW00100 to MW00109 are moved to MW00200 to MW00209.



Even if the source and destination overlap, the source data block will be moved to the destination without alteration.

## 4.5.4 CLEAR (CLR)

### ■ Overview

The CLR command clears the specified number of blocks from the leading data clear register, i.e., sets it to 0.

### ■ Description

#### Designation Method

```
CLR MW001000 W10 ;
      A      B
A: Leading bit number
B: Number to be shifted
C: Bit width
```

#### Data Types

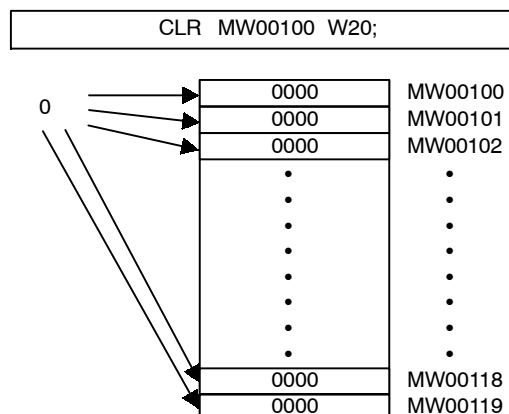
	Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
<b>Data Clear Register</b>	No	Yes	No	No	No
<b>Number of Blocks</b>	No	Yes	No	No	Yes

#### ◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	-	-
<b>W</b>	CLR MW00100 W10;	SETW MW00100 D=00000 W=10
<b>L</b>	-	-
<b>F</b>	-	-

**Example:** The contents of MW00100 to MW00119 are cleared to “0”.



## 4.5.5 ASCII CONVERSION 1 (ASCII)

<b>Use in a sequence program</b>	Possible
<b>Use in a motion program</b>	Possible

### ■ Overview

The ASCII command converts the character string specified during command execution into ASCII code and stores it in the specified register (integer register). Upper and lower case letters can be distinguished from each other.

The first character and the second character are stored respectively in the lower byte and upper byte of the first word in order. If the number of characters in the string is odd, the upper byte of the last word in the storage destination register becomes 0. The number of input characters is up to 32.



The following-version programming software is required to use an ASCII command.

<b>MP2000 Series (excluding MP2400) Version 2.60 or later</b>	MPE720 Version 5.38 or later
	MPE720 Version 6.04 or later
<b>MP2400 Version 2.60 or later</b>	MPE720 Version 6.04 or later
	MPE720 Version 6.04 Lite or later

### ■ Description

The ASCII command is designated as follows:

ASCII	<u>'ABCDEFGH'</u>	<u>MW00200;</u>
	A	B
	<b>Symbol</b>	<b>Application</b>
	A	Character string
	B	Storage destination register No.
		<b>Usable Register</b>
		ASCII characters
		Integer register (excluding # and C registers)

The following tables show the characters that can be used in the ASCII command.

### Usable characters

<b>Alphanumeric characters</b>	a to z, A to Z, 0 to 9
<b>Symbols</b>	Space, ! # \$ % & ( ) * + , - . / : ; < = > ? @ [ ] ¥ ] ^ _ ' {   } ~

### Unusable characters

<b>Single quotation</b>	'
<b>Double quotation</b>	“
<b>Double slash</b>	//

## ■ Programming Examples

The following illustration provides a programming example for the ASCII command.

### When character string “ABCD” is stored in MW00100 to MW00101

```
ASCII 'ABCD' MW00100;
```

	Upper byte	Lower byte	
MW00100	42H('B')	41H('A')	MW00100 = 4241H
MW00101	44H('D')	43H('C')	MW00101 = 4443H

### When character string “ABCDEFG” is stored in MW00100 to MW00103

```
ASCII 'ABCDEFG' MW00100;
```

	Upper byte	Lower byte	
MW00100	42H('B')	41H('A')	MW00100 = 4241H
MW00101	44H('D')	43H('C')	MW00101 = 4443H
MW00102	46H('F')	45H('E')	MW00102 = 4645H
MW00103	00H	47H('G')	MW00103 = 0047H

↑0 will be entered in the remaining byte.

## 4.6 Basic Functions

This section explains the basic function commands, such as trigonometric functions, square root, binary data conversion, and BCD data conversion.

### 4.6.1 SINE (SIN)

#### ■ Overview

The SIN command returns the sine of integer or real number data as the operation result. Double-length integer data cannot be used.

#### ■ Description

##### Designation Method

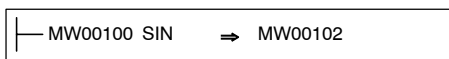
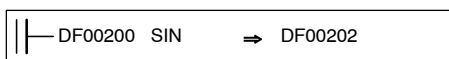
```
MW00100=SIN(MW00101);
MW00100=SIN(90);
MF00200=SIN(MF00202);
```

##### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
No	Yes	No	Yes	Yes

#### ◀EXAMPLE▶

#### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	-	-
<b>W</b>	MW00102=SIN(MW00100);	
<b>L</b>	-	-
<b>F</b>	DF00202=SIN(DF00200);	

The input units and output results are different for integer and real number data.

##### ● Integer Data

Integer data can be used within a range of  $-327.68$  to  $327.67$  degrees. The immediately preceding operation result (integer data) is used as the input, and the operation result is returned in an integer register (input unit 1 = 0.01 degrees). The operation result is multiplied by 10000 before being output.

4.6.2 COSINE (COS)

- Real Number Data

The command will use the immediately preceding operation result (real number data) as input, and return the sine in a real number register (unit = degrees).

**Example:**

- Integer Data

$$\boxed{\begin{array}{l} \text{MW00102} = \text{SIN} (\text{MW00100}) ; \\ (05000) \quad (03000) \end{array}} \xrightarrow{\text{Equivalent}} 0.5 = \text{SIN} 30^\circ$$

- Real Number Data

$$\boxed{\begin{array}{l} \text{MF00102} = \text{SIN} (\text{MF00100}) ; \\ (0.5) \quad (30.0) \end{array}}$$

**IMPORTANT**

If integer data is input outside the range of  $-327.68$  to  $327.67$  degrees, a correct result cannot be obtained.

**4.6.2 COSINE (COS)**

■ **Overview**

The COS command returns the cosine of integer or real number data as the operation result. Double integer data cannot be used.

■ **Description**

**Designation Method**

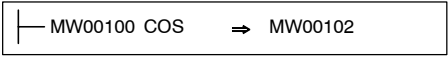

```
MW00100=COS(MW00101);
MW00100=COS(90);
MF00200=COS(MF00202);
```

**Data Types**

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
No	Yes	No	Yes	Yes

◀EXAMPLE▶

## ■ Programming Examples

Type	Motion Program	Ladder Logic Program
B	-	-
W	MW00102=COS(MW00100);	
L	-	-
F	DF00202=COS(DF00200);	

The input units and output results are different for integer and real number data.

- Integer Data

Integer data can be used within a range of  $-327.68$  to  $327.67$  degrees. The immediately preceding operation result (integer data) is used as input, and the operation result is returned in an integer register (input unit  $1 = 0.01$  degrees). The operation result is multiplied by 10000 before being output.

- Real Number Data

The command uses the immediately preceding operation result (real number data) as input, and returns the cosine in a real number register (unit = degrees).

**Example:**

- Integer Data

MW00102 = COS (MW00100) ; (05000)            (06000)	Equivalent $\Rightarrow$	$0.5 = \text{COS}60^\circ$
---	-----------------------------	----------------------------

- Real Number Data

MF00102 = COS (MF00100) ; (0.5)                    (60.0)
--

**IMPORTANT**

If integer data is input outside the range of  $-327.68$  to  $327.67$  degrees, a correct result cannot be obtained.



## 4.6.3 TANGENT (TAN)

### ■ Overview

The TAN command uses the specified variable or constant (unit = degrees) as input and returns the tangent in a real number register.

### ■ Description

#### Designation Method

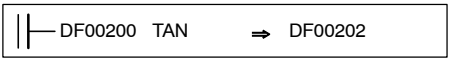
```
MF00100=TAN(MF00102);
MF00200=TAN(1.0);
```

#### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
No	No	No	Yes	Yes

#### ◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
B	-	-
W	-	-
L	-	-
F	DF00202=TAN(DF00200);	

**Example:** Calculates the tangent of the input value ( $\theta = 45.0$  degrees):  $TAN(\theta) = 1.0$ .

```
DF00102=TAN(DF00100);
(1.0)          (45.0)
```

#### IMPORTANT

The TAN command can use only real number data. If bits, integers, or double integers are specified, an error will result at compilation.

## 4.6.4 ARC SINE (ASN)

### ■ Overview

The ASN command uses the specified variable or constant as input and returns the arc sine (unit = degrees) in a real number register.

### ■ Description

#### Designation Method


```
MF00100=ASN(MF00102);
MF00200=ASN(1.0);
```

#### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
No	No	No	Yes	Yes

#### ◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	-	-
<b>W</b>	-	-
<b>L</b>	-	-
<b>F</b>	DF00202=ASN(DF00200);	

**Example:** Calculates the arc sine of the input value (0.5):  $ASN(0.5) = 30.0$  degrees.

```
MF00202=ASN(MF00200);
(30.0)      (0.5)
```

#### IMPORTANT

The ASN command can use only real number data. If bits, integers, or double integers are specified, an error will result at compilation.

## 4.6.5 ARC COSINE (ACS)

### ■ Overview

The ACS command uses the specified variable or constant as input and returns the arc cosine (unit = degrees) in a real number register.

### ■ Description

#### Designation Method

```
MF00100=ACS(MF00102);
MF00200=ACS(60.0);
```

#### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
No	No	No	Yes	Yes

#### ◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
B	-	-
W	-	-
L	-	-
F	DF00202=ACS(DF00200);	

**Example:** Calculates the arc cosine of the input value (0.5):  $ACS(0.5) = 60.0$  degrees.

```
MF00100=ACS(MF00102);
      (0.5)      (60.0)
```

#### IMPORTANT

The ACS command can use only real number data. If bits, integers, or double integers are specified, an error will result at compilation.

## 4.6.6 ARC TANGENT (ATN)

### ■ Overview

The ATN command returns the arc tangent of integer or real number data as the operation result. Double integer data cannot be used.

### ■ Description

#### Designation Method

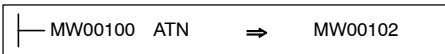
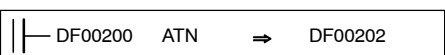
```
MW00100=ATN(MW00101);
MW00100=ATN(100);
MF00200=ATN(MF00202);
```

#### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
No	Yes	No	Yes	Yes

#### ◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	-	-
<b>W</b>	MW00102=ATN(MW00100);	
<b>L</b>	-	-
<b>F</b>	DF00202=ATN(DF00200);	

The input units and output results are different for integer and real number data.

- Integer Data

Integer data can be used within a range of  $-327.68$  to  $327.67$  degrees. The immediately preceding operation result (integer data) is used as the input, and the operation result is returned in an integer register (input unit 1 = 0.01 degrees). The operation result is multiplied by 100 before being output.

- Real Number Data

The command uses the immediately preceding operation result (real number data) as input, and returns the arc tangent in a real number register.

**Example:**

- Integer Data

MW00100 = ATN (MW00102);	⇒	45=ATN(1.0)
(04500)      (00100)		

Equivalent

- Real Number Data

MF00100 = ATN (MF00102);
(45.0)      (1.0)

## 4.6.7 SQUARE ROOT (SQT)

### ■ Overview

The SQT command returns the square root of an integer or real number as the operation result. Double integer data cannot be used.

### ■ Description

#### Designation Method

MW00100=SQT(MW00101);  
 MW00100=SQT(100);  
 MF00200=SQT(MF00202);

#### Data Types

Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
No	Yes	No	Yes	Yes

◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	-	-
<b>W</b>	MW00102=SQT(MW00100);	
<b>L</b>	-	-
<b>F</b>	DF00202=SQT(DF00200);	

The input units and output results are different for integer and real number data.

- Integer Data

The result is different from that obtained for the mathematical square root, and is calculated using the following formula:

$$32768 * \text{sign}(A) * \sqrt{|A|/32768}$$

sign(A): Sign of A register

|A|: Absolute value of A register

That is to say, the output is the result of the mathematically expressed square root multiplied by  $\sqrt{32768}$ . When the input is a negative number, an absolute square root is calculated, and the negative number is taken as the operation result. The operation error is a maximum of  $\pm 2$ .

- Real Number Data

The SQT command uses the immediately preceding operation result (real number data) as input and returns the square root in a real number register.

**Example:**

- Integer Data

Integer Input

MW00100 = SQT (MW00102);	$\sqrt{64} \times \sqrt{32768} = 1448$
(01448)            (00064)	

Negative Number Input

MW00100 = SQT (MW00102);	$-(\sqrt{64} \times \sqrt{32768}) = -1448$
(-01448)            (-00064)	

- Real Number Data

Integer Input

MF00100 = SQT (MF00102);
(8.0)            (64.0)

Negative Number Input

MF00100 = SQT (MF00102);
(-8.0)            (-64.0)

## 4.6.8 BCD-TO-BINARY (BIN)

### Overview

The BIN command converts BCD data to binary data.

Only integer data can be used. If a non-BCD data is specified, a correct result cannot be obtained.

### Description

#### Designation Method

```
MW00100=BIN(MW00101);
MW00100=BIN(1234H);
MF00200=BIN(ML00202);
```

#### Data Types

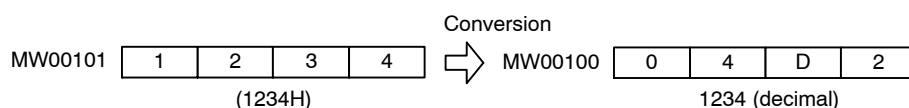
Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
No	Yes	Yes	No	Yes

#### EXAMPLE

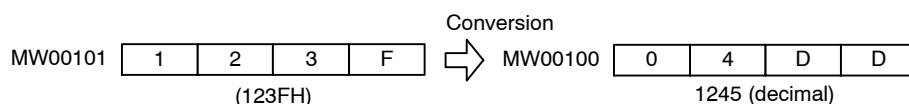
### Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	-	-
<b>W</b>	MW00101=BIN(MW00100);	
<b>L</b>	ML00102=BIN(MW00100);	
<b>F</b>	-	-

#### Example 1:



#### Example 2:



If non-BCD data is specified, a correct result cannot be obtained.

## 4.6.9 BINARY-TO-BCD (BCD)

### ■ Overview

The BCD command converts binary data to BCD data.

Only integer data can be used. If the binary data is 9999 or higher or is a negative value, a correct result cannot be obtained.

### ■ Description

#### Designation Method

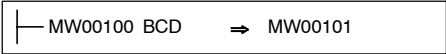
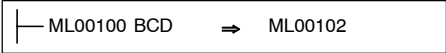
```
MW00100=BCD(MW00101);
MW00100=BCD(1234);
MF00200=BCD(ML00202);
```

#### Data Types

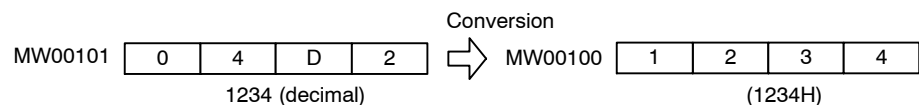
Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
No	Yes	Yes	No	Yes

### ◀EXAMPLE▶

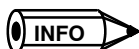
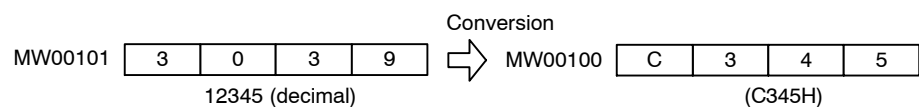
### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
B	-	-
W	MW00101=BCD(MW00100);	
L	ML00102=BCD(ML00100);	
F	-	-

#### Example 1:



#### Example 2:



If the binary data is greater than 9999, a correct result cannot be obtained.



## 4.6.10 SET BIT (S{ })

### ■ Overview

This command turns ON the specified bit if the logical operation result is true. It does not turn OFF the specified bit, even if the logical operation result is false.

### ■ Description

#### Designation Method

S{MB001000}=MB001010&MB001011;	
A	B
A: Specified bit	
B: Logic expression	

#### Data Types

	Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
<b>Specified Bit</b>	Yes	No	No	No	No
<b>Logic Expression</b>	Yes	No	No	No	Yes

#### ◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	S{MB001000}=MB001010&MB001011;	-
<b>W</b>	-	-
<b>L</b>	-	-
<b>F</b>	-	-

## 4.6.11 RESET BIT (R{ })

### ■ Overview

This command turns OFF the specified bit if the logical operation result is true. It does not turn ON the specified bit, even if the logical operation result is false.

### ■ Description

#### Designation Method

R{ <u>MB001000</u> }= <u>MB001010</u> & <u>MB001011</u> ;	
A	B
A: Specified bit	
B: Logic expression	

#### Data Types

	Bit (B)	Integer (W)	Double Integer (L)	Real Number (F)	Constant
<b>Specified Bit</b>	Yes	No	No	No	No
<b>Logic Expression</b>	Yes	No	No	No	Yes

#### ◀EXAMPLE▶

### ■ Programming Examples

Type	Motion Program	Ladder Logic Program
<b>B</b>	R{MB001000}=MB001010&MB001011;	–
<b>W</b>	–	–
<b>L</b>	–	–
<b>F</b>	–	–

# 5

---

## Variables (Registers)

This chapter explains the variables that can be used in motion programs.

<b>5.1 Overview</b> .....	<b>5 -2</b>
5.1.1 Overview of Variables .....	5 -2
5.1.2 Global Variables and Local Variables .....	5 -4
<b>5.2 Using Variables</b> .....	<b>5 -7</b>
5.2.1 System Variables (S Register) .....	5 -7
5.2.2 Data Variables (M Registers) .....	5 -8
5.2.3 Input Variables (I Registers) .....	5 -9
5.2.4 Output Variables (O Registers) .....	5 -12
5.2.5 C Variables (C Registers) .....	5 -15
5.2.6 D Variables (D Registers) .....	5 -15

## 5.1 Overview

This section summarizes the variables used in motion programs.

### 5.1.1 Overview of Variables

In a motion program, variables can be coded in place of numeric values. When variables are used in actual operations, the numeric values stored in the variable area are retrieved.

#### ■ Types of Variable (Registers)

The seven types of register shown in the following table can be used as variables in a motion program. The S, M, I, O, and C registers are global variables that can be used in common by motions programs and ladder logic programs.

The # and D registers are local variables that can be referenced only by specific programs. Because local variables are stored in user memory, the unused capacity of the programs will be reduced.

**Table 5.1 Types of Variable**

Type	Name	Designation Method	Range	Description	Characteristic
<b>S</b>	<b>System Registers</b>	SB, SW, SL, SFnnnnn	SW00000 to SW01023	Registers that can be referenced by the system. Register No. nnnnn is a decimal expression. When the system is started, SW00000 to SW00049 are cleared to 0.	Common to programs
<b>M</b>	<b>Data Registers</b>	MB, MW, ML, MFnnnnn	MW00000 to MW32767	Registers common to all drawings. Used as interfaces between drawings. Register No. nnnnn is a decimal expression.	
<b>I</b>	<b>Input Registers</b>	IB, IW, IL, IFhhhh	IW0000 to IW07FF	Registers used for I/O Module input data. Register No. hhhh is a hexadecimal expression. Register numbers C000 and onward are used as interfaces with the servo monitor parameters.	
<b>O</b>	<b>Output Registers</b>	OB, OW, OL, OFhhhh	OW0000 to OW07FF	Registers used for I/O Module output data. Register No. hhhh is a hexadecimal expression. Register numbers C000 and onward are used as interfaces with the servo setting parameters.	
<b>C</b>	<b>Constant Registers</b>	CB, CW, CL, CFnnnnn	CW00000 to CW4095	Registers that can be referenced only by a program. Register No. nnnnn is a decimal expression.	

Type	Name	Designation Method	Range	Description	Characteristic
#	# registers	#B, #W, #L, #Fnnnnn	#W00000 to #W16383	Registers that can be referenced only by a program. Can be referenced only by the corresponding drawing. The actual range used is specified by the user on the MPE720. Register No. nnnnn is a decimal expression. <u>Cannot be used in a motion program.</u>	Unique to a program
D	D registers	DB, DW, DL, DFnnnnn	DW00000 to DW16383	Registers unique to each drawing. Can only be used by the corresponding drawing. The actual range used is specified by the user on the MPE720. Register No. nnnnn is a decimal expression.	

**IMPORTANT**

A # register cannot be used in a motion program. If a # register is used, a syntax error will result during compilation of the motion program.

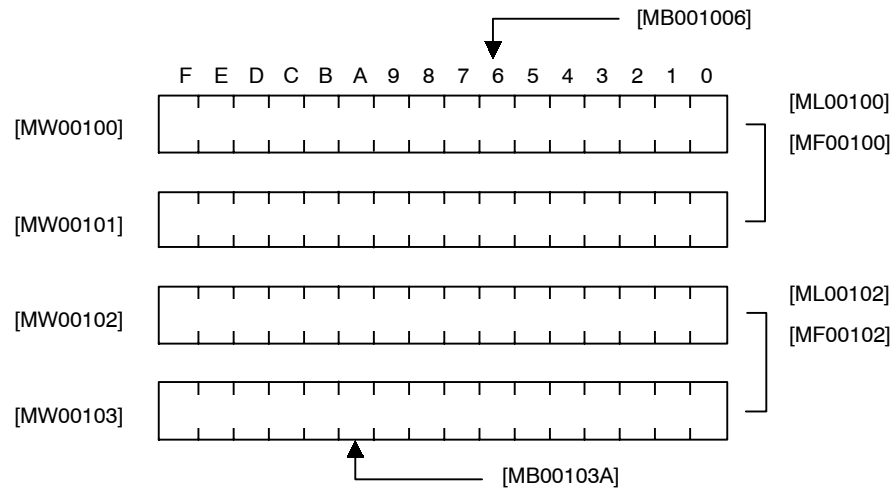
## ■ Data Types

As shown in the following table, the data types are bit, integer, double integer, and real number data. Use them as required.

**Table 5.2 Data Types**

Symbol	Data Type	Numeric Range	Remarks
<b>B</b>	<b>Bit</b>	ON(1), OFF(0)	Used to determine the relay sequence and the ON/OFF condition.
<b>W</b>	<b>Integer</b>	-32768 to +32767 (8000H to 7FFFH)	Used for arithmetic operations. The parentheses ( ) show its use in logic operations.
<b>L</b>	<b>Double Integer</b>	-2147483648 to +2147483647 (80000000H to 7FFFFFFFH)	Used for arithmetic operations. The parentheses ( ) show its use in logic operations.
<b>F</b>	<b>Real Number</b>	±(1.175E-38 to 3.402E+38), 0	Used for high-level arithmetic operations.

### Register Designation and Data Type



## 5.1.2 Global Variables and Local Variables

### ■ Global Variables

Global variables can be used in common by ladder logic programs, user functions, and the drawings in motion programs. In other words, the calculated results for a given ladder logic program can be used by other user functions and motion programs. The global variable size is stored by the system for each variable. (See the following illustration.)

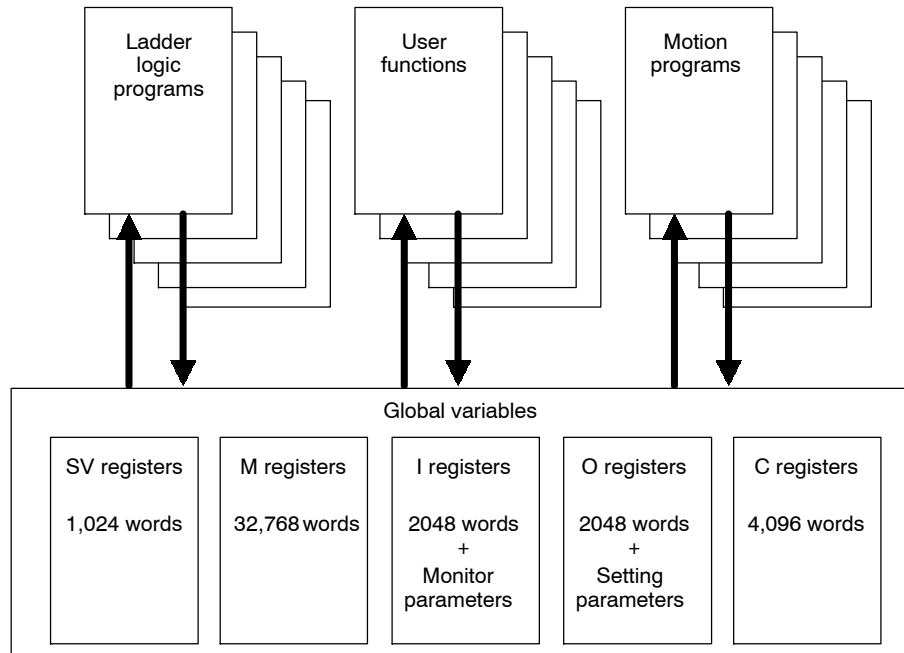
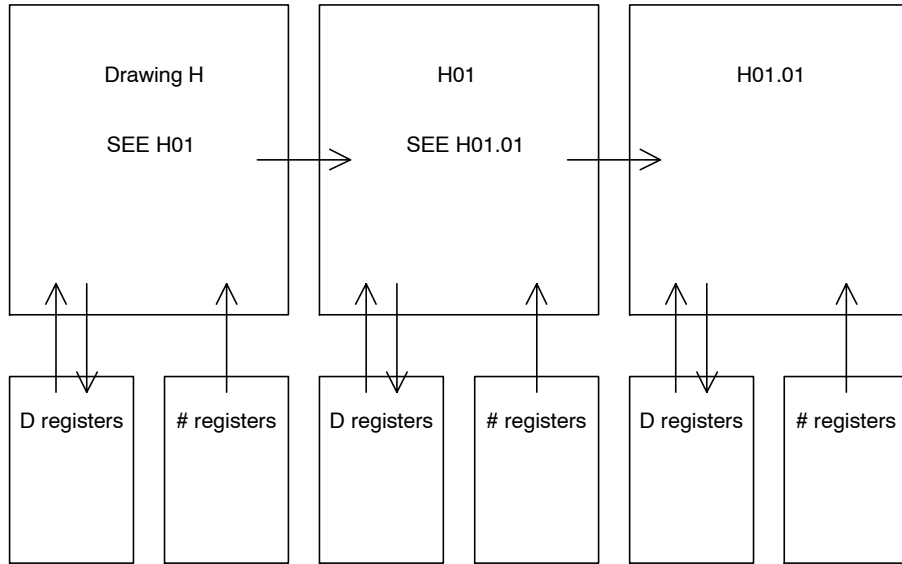


Figure 5.1 Global Variables (MP930)

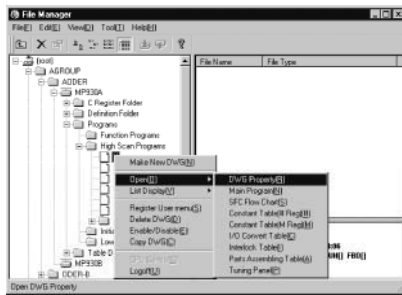
## Local Variables

Local variables are used locally by each program. The operation results in programs are stored using the variables stored independently by program.

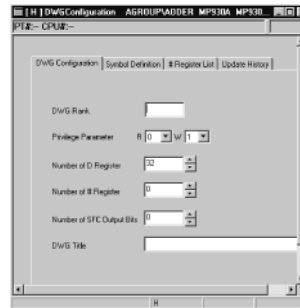


The range of variables used by each program is specified on the DWG Properties window and on the Motion Program Configuration Definition window. Up to 16 kW can be stored for one drawing.

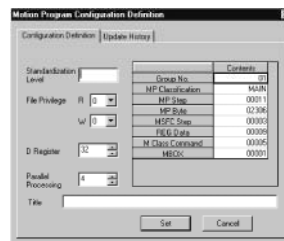
5



DWG Configuration Definition Screen



Motion Program Configuration Definition window



**IMPORTANT**

# registers cannot be used by motion programs.

**IMPORTANT****■ Precautions on Variable Operations****Commands that Generate Errors**

An error will result in commands such as the following:

- Integer Data Stored in Bit Variables

```
MB000100 = 123;
MB000100 = MW00100;
```

**Data Stored in Variables of Different Data Types**

When data is stored in variables of a different data type, results such as the following will be obtained:

- Real Number Data Stored in Integer Variables

```
MW00100 = MF00200;
(00001)    (1.234)
Real number values are converted to integers and stored.
```

- Real Number Data Stored in Double Integer Variables

```
ML00100 = MF00200;
(123457)   (123456.7)
Real number values are converted to integers and stored.
```

- Double Integer Data Stored in Integer Variables

```
MW00100 = ML00200;
(-00001)   (65535)
The lower 16 bits of double integer data is stored as it is.
```

- Real Number Data Stored in Double Integer Variables

```
ML00100 = MW00200;
(0001234)   (1234)
Integer data is converted to double integer data and stored.
```

**Real Number Data Stored in Integer Variables**

Be careful of rounding errors when real number data is stored in integer variables.

```
MW00100 = MF00200 + MF00202;
(0124)    (123.48)    (0.02)
(0123)    (123.49)    (0.01)
The operation result will vary according to the value of the variable being calculated.
```



## 5.2 Using Variables

This section explains how to use variables.

### 5.2.1 System Variables (S Registers)

#### ■ Overview

System variables (S registers) are provided by the MP-series Machine Controller system. They can be used to read system error information, the operation status, and so on. S registers are global variables that can be used in any motion program. For details, refer to the *Corresponding Machine Controller User's Manual: Design and Maintenance*.

#### ■ Description

S registers are designated as follows:

SB000000 to SB01023F
SW000000 to SW01023
SL000000 to SL01023
SF000000 to SF01023

The variable number is expressed as a decimal. When bits are specified, the bit number is expressed in hexadecimal.

#### ■ Programming Examples

##### ◀EXAMPLE▶

- Bit Designation  
OB000010 = SB000402|SB000403;
- Integer Designation  
MW00100 = SW00041;
- Double Integer Designation  
ML00100 = SL00062;

##### IMPORTANT

The system registers (S) are used exclusively for reading. If they are written to, system operations cannot be guaranteed.

## 5.2.2 Data Variables (M Registers)

### ■ Overview

M registers are general-purpose variables that can be used in ladder logic programs, user functions, and all motion programs. They are global variables that can be used as interfaces between motion programs and ladder logic programs.

### ■ Description

M registers are designated as follows:

MB000000 to MB32767F
MW000000 to MW32767
ML000000 to ML32767
MF000000 to MF32767

The M register can be used as a variable for each type of operation and substituted for the operation result, or specified as the variable for the positioning coordinate value or the speed. The variable number is expressed as a decimal.

### ■ Programming Examples

#### ◀EXAMPLE▶

#### Specifying the Position and Speed in Axis Move Commands as Variables

<ul style="list-style-type: none"> <li>Parameter Reference unit = mm When decimal point position = 3</li> </ul>	
ML00100 = 100000;	→ 1000.000 mm
ML00102 = 200000;	→ 2000.000 mm
ML00104 = 300000;	→ 3000.000 mm
ML00106 = 500000;	→ 5000.000 mm/min
MVS [X]ML00100 [Y]ML00102 [Z]ML00104 FML00106;	

#### ◀EXAMPLE▶

#### Using Variables in Operations

- Bit Designation  
MB001001 = IB000100 & IB000201;
- Integer Designation  
MW00101 = (MW00101 | MW00102) & FF0CH;
- Double Integer Designation  
ML00200 = (MI00202\*ML00204/ML00206)\*3;
- Real Number Designation  
MF00200 = MF00202\*MF00204/MF00206\*3.14;

**IMPORTANT**

When the travel distance coordinate values or speed is designated as a variable in the following motion commands, double integer data must be used.

MOV, MVS, MCW/MCC, ZRN, SKP, MVT, EXM, POS, ACC, SCC, IAC, IDC, IFP, FMX, INP

### 5.2.3 Input Variables (I Registers)

#### ■ Overview

These variables are used by external input signals and the servo monitor parameters. Although servo parameters can also be used for writing data, the values cannot be guaranteed.

#### ■ Description

I registers are designated as follows:

● MP910		
	IW0000 to IW13FF	: External input signals
	IWC000 to IWC77F	: Servo monitor parameters
● MP920		
	IW0000 to IW13FF	: External input signals
	IWC000 to IWFF7F	: Servo monitor parameters
● MP930		
	IW0000 to IW07FF	: External input signals
	IWC000 to IWC37F	: Servo monitor parameters
● MP940		
	IW0000 to IW07FF	: External input signals
	IWC000 to IWF03F	: Servo monitor parameters
● MP2000 Series Machine Controller		
	IW0000 to IW7FFF	: External input signals
	IW8000 to IWFFFF	: Servo monitor parameters

#### External Input Signal Register Numbers

- Depends on the address specified in the Module configuration definition.  
Local inputs in the MP930 MC350 module are fixed to IW0000 (IB00000 to IB0000F, 16 points).

#### Servo Monitor Parameter Register Numbers

The number of controlled axes depends on the module type. The following indicates the number of controlled axes for each module and the maximum number of modules.

**Table 5.3 Number of Controlled Axes per Module**

Module Name		Number of Controlled Axes per Module	Maximum Number of Modules
MP910		14	2
MP920	SVA-01	4	15
	SVA-02	2	16
	SVB-01	14	16
	PO-01	4	16
MP930		14	1
MP940		1	1
MP2100		16	1
MP2300		16	1

**Table 5.4 MP900-series Machine Controller Motion Parameter Register Numbers**

Axis No. Line No.	No.1 axis	No.2 axis	No.3 axis	No.4 axis	No.5 axis	. .	No.14 axis
1	C000 to C03F	C040 to C07F	C080 to C0BF	C0C0 to C0FF	C100 to C13F	. .	C340 to C37F
2	C400 to C43F	C440 to C47F	C480 to C4BF	C4C0 to C4FF	C500 to C53F	. .	C740 to C77F
3	C800 to C83F	C840 to C87F	C880 to C8BF	C8C0 to C8FF	C900 to C93F	. .	CB40 to CB7F
4	CC00 to CC3F	CC40 to CC7F	CC80 to CCBF	CCC0 to CCFE	CD00 to CD3F	. .	CF40 to CF7F
5	D000 to D03F	D040 to D07F	D080 to D0BF	D0C0 to D0FF	D100 to D13F	. .	D340 to D37F
6	D400 to D43F	D440 to D47F	D480 to D4BF	D4C0 to D4FF	D500 to D53F	. .	D740 to D77F
7	D800 to D83F	D840 to D87F	D880 to D8BF	D8C0 to D8FF	D900 to D93F	. .	DB40 to DB7F
8	DC00 to DC3F	DC40 to DC7F	DC80 to DCBF	DCC0 to DCFE	DD00 to DD3F	. .	DF40 to DF7F
9	E000 to E03F	E040 to E07F	E080 to E0BF	E0C0 to E0FF	E100 to E13F	. .	E340 to E37F
10	E400 to E43F	E440 to E47F	E480 to E4BF	E4C0 to E4FF	E500 to E53F	. .	E740 to E77F
11	E800 to E83F	E840 to E87F	E880 to E8BF	E8C0 to E8FF	E900 to E93F	. .	EB40 to EB7F
12	EC00 to EC3F	EC40 to EC7F	EC80 to ECBF	ECC0 to ECFE	ED00 to ED3F	. .	EF40 to EF7F
13	F000 to F03F	F040 to F07F	F080 to F0BF	F0C0 to F0FF	F100 to F13F	. .	F340 to F37F
14	F400 to F43F	F440 to F47F	F480 to F4BF	F4C0 to F4FF	F500 to F53F	. .	F740 to F77F
15	F800 to F83F	F840 to F87F	F880 to F8BF	F8C0 to F8FF	F900 to F93F	. .	FB40 to FB7F
16	FC00 to FC3F	FC40 to FC7F	FC80 to FCBF	FCC0 to FCFE	FD00 to FD3F	. .	FF40 to FF7F

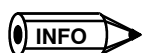
↑  
Module No. Offset

5

Table 5.5 MP2000-series Machine Controller Motion Parameter Register Numbers

Axis No. Line No.	No.1 axis	No.2 axis	No.3 axis	No.4 axis	No.5 axis	. .	No.14 axis
1	8000 to 807F	8080 to 80FF	8100 to 817F	8180 to 81FF	8200 to 827F	. .	8780 to 87FF
2	8800 to 887F	8880 to 88FF	8900 to 897F	8980 to 89FF	8A00 to 8A7F	. .	8F80 to 8FFF
3	9000 to 907F	9080 to 90FF	9100 to 917F	9180 to 91FF	9200 to 9A7F	. .	9780 to 97FF
4	9800 to 987F	9880 to 98FF	9900 to 997F	9980 to 99FF	9A00 to 997F	. .	9F80 to 9FFF
5	A000 to A07F	A080 to A0FF	A100 to A17F	A180 to A1FF	A200 to A27F	. .	A780 to A7FF
6	A800 to A87F	A880 to A8FF	A800 to A87F	A980 to A9FF	AA00 to AA7F	. .	AF80 to AFFF
7	B000 to B07F	B080 to B0FF	B100 to B17F	B180 to B1FF	B200 to B27F	. .	B780 to B7FF
8	B800 to B87F	B880 to B8FF	B900 to B97F	B980 to B9FF	BA00 to BA7F	. .	BF80 to BFFF
9	C000 to C07F	C080 to C0FF	C100 to C17F	C180 to C1FF	C200 to C27F	. .	C780 to C7FF
10	C800 to C87F	C880 to C8FF	C900 to C97F	C980 to C9FF	CA00 to CA7F	. .	CF80 to CFFF
11	D000 to D07F	D080 to D0FF	D100 to D17F	D180 to D1FF	D200 to D27F	. .	D780 to D7FF
12	D800 to D87F	D880 to D8FF	D900 to D97F	D980 to D9FF	DA00 to DA7F	. .	DF80 to DFFF
13	E000 to E07F	E080 to E0FF	E100 to E17F	E180 to E1FF	E200 to E27F	. .	E780 to E7FF
14	E800 to E87F	E880 to E8FF	E900 to E97F	E980 to E9FF	EA00 to A97F	. .	EF80 to EFFF
15	F000 to F07F	F080 to F0FF	F100 to F17F	F180 to F1FF	F200 to F27F	. .	F780 to EFFF
16	F800 to F87F	F880 to F8FF	F900 to F97F	F980 to F9FF	F900 to F97F	. .	FF80 to FFFF

↑  
Module No. Offset



The register number for each axis in the servo monitor parameter is obtained using the following formula:

MP900-series Machine Controllers

- Axis monitor parameter register address = IW0000 + module number offset + (axis No.1) × 40 (HEX)

MP2000-series Machine Controllers

- Axis monitor parameter register address = IW0000 + module number offset + (axis No.1) × 80 (HEX)

## ■ □ Programming Examples

External output signals and servo monitor parameters are read out and referenced.

- Bit Designation  
MB01000 = IB0010 & IB00105;
- Integer Designation  
MW0100 = IWC016;
- Double Integer Designation  
ML0100 = ILC022;

## 5.2.4 Output Variables (O Registers)

### ■ Overview

These variables are used for external output signals and servo setting parameters.

### ■ Description

O registers are designated as follows:

● MP910		
	OW0000 to OW13FF	: External output signals
	OWC000 to OWC77F	: Servo monitor parameters
● MP920		
	OW0000 to OW13FF	: External output signals
	OWC000 to OWFF7F	: Servo monitor parameters
● MP930		
	OW0000 to OW07FF	: External output signals
	OWC000 to OWC37F	: Servo monitor parameters
● MP940		
	OW0000 to OW07FF	: External output signals
	OWC000 to OWF03F	: Servo setting parameters
● MP2000 Series Machine Controller		
	OW0000 to OW7FFF	: External output signals
	OW8000 to OWFFFF	: Servo setting parameters

### External Output Signal Register Numbers

- Depends on the address specified in the Module configuration definition.  
Local inputs in the MP930 MC350 module are fixed to OW0000 (OB00010 to OB0001F, 16 points).

### Servo Setting Parameter Register Numbers

The number of controlled axes depends on the module type. The following indicates the number of controlled axes for each module and the maximum number of modules.

Table 5.6 Number of Controlled Axes per Module

Module Name		Number of Controlled Axes per Module	Maximum Number of Modules
MP910		14	2
MP920	SVA-01	4	15
	SVA-02	2	16
	SVB-01	14	16
	PO-01	4	16
MP930		14	1
MP940		1	1
MP2100		16	1
MP2300		16	1

Table 5.7 MP900-series Machine Controller Motion Parameter Register Numbers

Axis No. Line No.	No.1 axis	No.2 axis	No.3 axis	No.4 axis	No.5 axis	. .	No.14 axis
1	C000 to C03F	C040 to C07F	C080 to C0BF	C0C0 to C0FF	C100 to C13F	. .	C340 to C37F
2	C400 to C43F	C440 to C47F	C480 to C4BF	C4C0 to C4FF	C500 to C53F	. .	C740 to C77F
3	C800 to C83F	C840 to C87F	C880 to C8BF	C8C0 to C8FF	C900 to C93F	. .	CB40 to CB7F
4	CC00 to CC3F	CC40 to CC7F	CC80 to CCBF	CCC0 to CCFE	CD00 to CD3F	. .	CF40 to CF7F
5	D000 to D03F	D040 to D07F	D080 to D0BF	D0C0 to D0FF	D100 to D13F	. .	D340 to D37F
6	D400 to D43F	D440 to D47F	D480 to D4BF	D4C0 to D4FF	D500 to D53F	. .	D740 to D77F
7	D800 to D83F	D840 to D87F	D880 to D8BF	D8C0 to D8FF	D900 to D93F	. .	DB40 to DB7F
8	DC00 to DC3F	DC40 to DC7F	DC80 to DCBF	DCC0 to DCFE	DD00 to DD3F	. .	DF40 to DF7F
9	E000 to E03F	E040 to E07F	E080 to E0BF	E0C0 to E0FF	E100 to E13F	. .	E340 to E37F
10	E400 to E43F	E440 to E47F	E480 to E4BF	E4C0 to E4FF	E500 to E53F	. .	E740 to E77F
11	E800 to E83F	E840 to E87F	E880 to E8BF	E8C0 to E8FF	E900 to E93F	. .	EB40 to EB7F
12	EC00 to EC3F	EC40 to EC7F	EC80 to ECBF	ECC0 to ECFE	ED00 to ED3F	. .	EF40 to EF7F
13	F000 to F03F	F040 to F07F	F080 to F0BF	F0C0 to F0FF	F100 to F13F	. .	F340 to F37F
14	F400 to F43F	F440 to F47F	F480 to F4BF	F4C0 to F4FF	F500 to F53F	. .	F740 to F77F
15	F800 to F83F	F840 to F87F	F880 to F8BF	F8C0 to F8FF	F900 to F93F	. .	FB40 to FB7F
16	FC00 to FC3F	FC40 to FC7F	FC80 to FCBF	FCC0 to FCFE	FD00 to FD3F	. .	FF40 to FF7F

↑  
Module No. Offset

Table 5.8 MP2000-series Machine Controller Motion Parameter Register Numbers

Axis No. Line No.	No.1 axis	No.2 axis	No.3 axis	No.4 axis	No.5 axis	. .	No.14 axis
1	8000 to 807F	8080 to 80FF	8100 to 817F	8180 to 81FF	8200 to 827F	. .	8780 to 87FF
2	8800 to 887F	8880 to 88FF	8900 to 897F	8980 to 89FF	8A00 to 8A7F	. .	8F80 to 8FFF
3	9000 to 907F	9080 to 90FF	9100 to 917F	9180 to 91FF	9200 to 9A7F	. .	9780 to 97FF
4	9800 to 987F	9880 to 98FF	9900 to 997F	9980 to 99FF	9A00 to 997F	. .	9F80 to 9FFF
5	A000 to A07F	A080 to A0FF	A100 to A17F	A180 to A1FF	A200 to A27F	. .	A780 to A7FF
6	A800 to A87F	A880 to A8FF	A800 to A87F	A980 to A9FF	AA00 to AA7F	. .	AF80 to AFFF
7	B000 to B07F	B080 to B0FF	B100 to B17F	B180 to B1FF	B200 to B27F	. .	B780 to B7FF
8	B800 to B87F	B880 to B8FF	B900 to B97F	B980 to B9FF	BA00 to BA7F	. .	BF80 to BFFF
9	C000 to C07F	C080 to C0FF	C100 to C17F	C180 to C1FF	C200 to C27F	. .	C780 to C7FF
10	C800 to C87F	C880 to C8FF	C900 to C97F	C980 to C9FF	CA00 to CA7F	. .	CF80 to CFFF
11	D000 to D07F	D080 to D0FF	D100 to D17F	D180 to D1FF	D200 to D27F	. .	D780 to D7FF
12	D800 to D87F	D880 to D8FF	D900 to D97F	D980 to D9FF	DA00 to DA7F	. .	DF80 to DFFF
13	E000 to E07F	E080 to E0FF	E100 to E17F	E180 to E1FF	E200 to E27F	. .	E780 to E7FF
14	E800 to E87F	E880 to E8FF	E900 to E97F	E980 to E9FF	EA00 to A97F	. .	EF80 to EFFF
15	F000 to F07F	F080 to F0FF	F100 to F17F	F180 to F1FF	F200 to F27F	. .	F780 to EFFF
16	F800 to F87F	F880 to F8FF	F900 to F97F	F980 to F9FF	F900 to F97F	. .	FF80 to FFFF



Module No. Offset



The register number for each axis in the servo monitor parameter is obtained using the following formula:

MP900-series Machine Controllers

- Axis monitor parameter register address =  $0W0000 + \text{module number offset} + (\text{axis No.1}) \times 40$  (HEX)

MP2000-series Machine Controllers

- Axis setting parameter register address =  $0(I)W0000 + \text{module number offset} + (\text{axis No.1}) \times 80$  (HEX)

## ■ Programming Examples

The position and speed are specified as variables in axis move commands.

- Bit Designation  
OB01000 = MB001000 & IB00100;
- Integer Designation  
OWC020 = MW00100;
- Double Integer Designation  
OLC022 = ML00100 + ML00200;
- Real Number Designation  
OF00200 = MF00100 + MF00102;



## 5.2.5 C Variables (C Registers)

### ■ Overview

With C registers, the data created in the constant table can be referenced only by programs. Writing is not possible.

### ■ Description

C registers are designated as follows:

CW00000 to CW4095

C registers cannot be written from programs.

### ■ Programming Examples

#### Using Variables in Operations

- Bit Designation  
MB001000 = CB001001;
- Integer Designation  
MWC00100 = CW00100;
- Double Integer Designation  
MLC00100 = CL00100;
- Real Number Designation  
MF00100 = CF00100;

## 5.2.6 D Variables (D Registers)

### ■ Overview

D variables can be used only by the relevant program using specific internal registers for each motion program.

### ■ Description

D registers are designated as follows:

DW00000 to DW16383 (maximum)

The D register can be used as a variable for each type of operation and substituted for the operation result, or specified as the variable for the positioning coordinate value or the speed. The variable number is expressed as a decimal.

The size is specified in the program configuration definition (Properties). (Default: 32 words)

## ■ Programming Examples

### ◀EXAMPLE▶

### Specifying the Position and Speed in Axis Move Commands as Variables

<ul style="list-style-type: none"> <li>Parameter Reference unit = mm When decimal point position = 3</li> </ul>	
DL00100 = 100000;	→ 1000.000 mm
DL00102 = 200000;	→ 2000.000 mm
DL00104 = 300000;	→ 3000.000 mm
DL00106 = 500000;	→ 5000.000 mm/min
MVS [X]DL00100 [Y]DL00102 [Z]DL00104 FDL00106;	

### ◀EXAMPLE▶

### Using Variables in Operations

- Bit Designation  
DB001000 = IB001001 & MB000101;
- Integer Designation  
DW00102 = CW00103 | DW00104 & DW00105;
- Double Integer Designation  
DLC00106 = DL00108\*ML0011/ML00200;
- Real Number Designation  
DF00200 = MF00202\*DF00202\*3.14;

### IMPORTANT

When the travel distance coordinate value or speed is designated as a variable in the following motion commands, double integer data must be used.

MOV, MVS, MCW/MCC, ZRN, SKP, MVT, EXM, POS, ACC, SCC, AC, DC, FP, FMX, INP

# 6

---

## MP2300S/MP2400 Exclusive-use Commands

This chapter explains the programming of commands that have been newly added for the exclusive use of MP2300S and MP2400.

### 6.1 MP2300S/MP2400 Exclusive-use

Commands .....	6 -2
6.1.1 RISING PULSE (PON) .....	6 -2
6.1.2 FALLING PULSE (NON) .....	6 -4
6.1.3 ON-DELAY TIMER (TON):	
Counting unit: 0.01 second .....	6 -6
6.1.4 OFF-DELAY TIMER (TOF):	
Counting unit: 0.01 second .....	6 -7
6.1.5 USER FUNCTION CALL (FUNC) .....	6 -9
6.1.6 SEQUENCE SUBROUTINE CALL (SSEE) .....	6 -10

## 6.1 MP2300S/MP2400 Exclusive-use Commands

This section provides an overview of the programming of commands that have been newly added for the exclusive use of MP2300S and MP2400.

To use an MP2300S/MP2400 exclusive-use command, the following-version programming software is required.

<b>MP2300S Version 2.60 or later</b>	MPE720 Version 5.38 or later
	MPE720 Version 6.04 or later
<b>MP2400 Version 2.60 or later</b>	MPE720 Version 6.04 or later
	MPE720 Version 6.04 Lite or later

### 6.1.1 RISING PULSE (PON)

<b>Use in a sequence program</b>	Possible
<b>Use in a motion program</b>	Not possible

#### ■ Overview

The PON command is ON during one scan of bit output when the bit input status changes from OFF to ON. The register that stores the previous bit output value is used as a work of PON processing. Set the registers that are not used for other processes.

#### ■ Description

The PON command is designated as follows:

$\underbrace{\text{DB000002}}_A = \text{PON}(\underbrace{\text{DB000000}}_B \underbrace{\text{DB000001}}_C);$		
Description	Application	Usable Register
A	Bit output	Bit register (excluding # and C registers)
B	Bit input	All bit registers
C	To store the previous bit output value	Bit register (excluding # and C registers)

## ■ Programming Examples

The following illustration provides a programming example for the PON command.

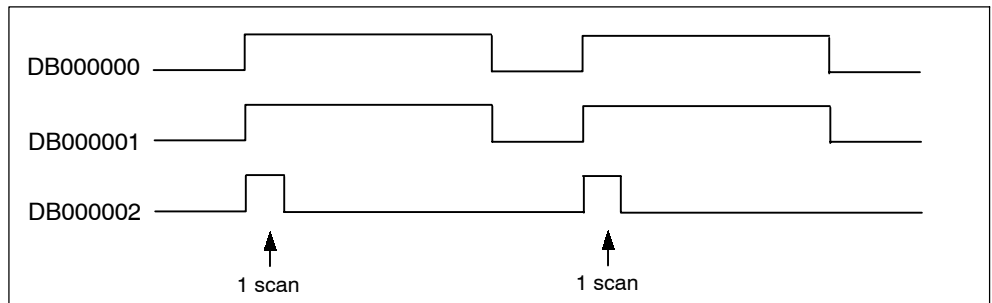
### Outputting to a coil

```
DB000002=PON(DB000000) DB000001);
```

- Ladder equivalent circuit



- Timing chart



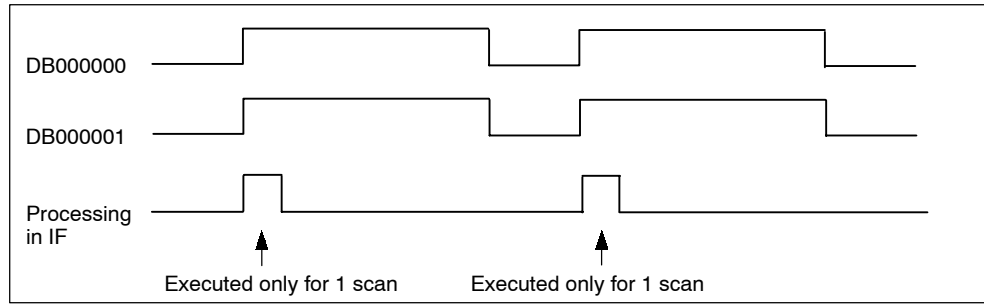
### Using the PON command together with the IF command

```
IF PON(DB000000 DB000001)==1;
  :
  :
IEND;
```

- Ladder equivalent circuit



● Timing chart



### 6.1.2 FALLING PULSE (NON)

Use in a sequence program	Possible
Use in a motion program	Not possible

■ Overview

FALLING PULSE (NON) is ON during one scan of bit output when the bit input status changes from ON to OFF. The register that stores the previous bit output value is used as a work of NON processing. Set the registers that are not used for other processes.

■ Description

The NON command is designated as follows:

$$\frac{\text{DB000002}=\text{NON}(\text{DB000000}) \text{ DB000001};}{\begin{matrix} \text{A} & \text{B} & \text{C} \end{matrix}}$$

Symbol	Application	Usable Register
A	Bit output	Bit register (excluding # and C registers)
B	Bit input	All bit registers
C	To store the previous bit output value	Bit register (excluding # and C registers)

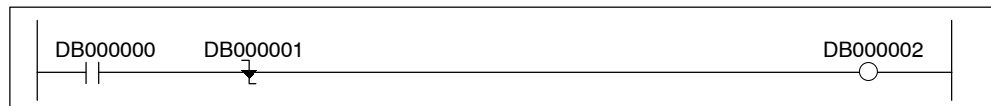
■ Programming Examples

The following illustration provides a programming example for the NON command.

**Outputting to a coil**

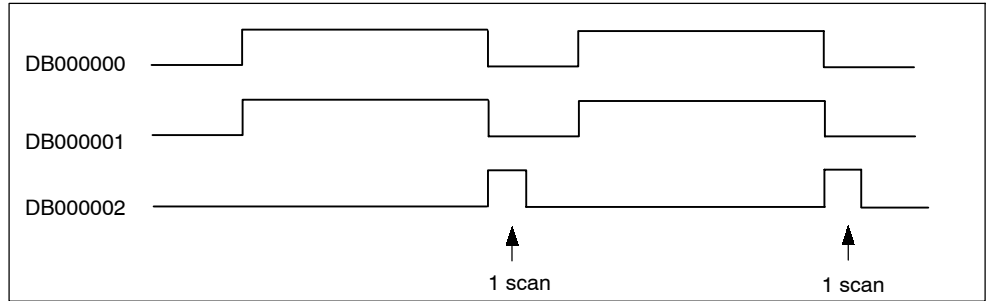
```
DB000002=NON(DB000000) DB000001;
```

● Ladder equivalent circuit



6

• Timing chart



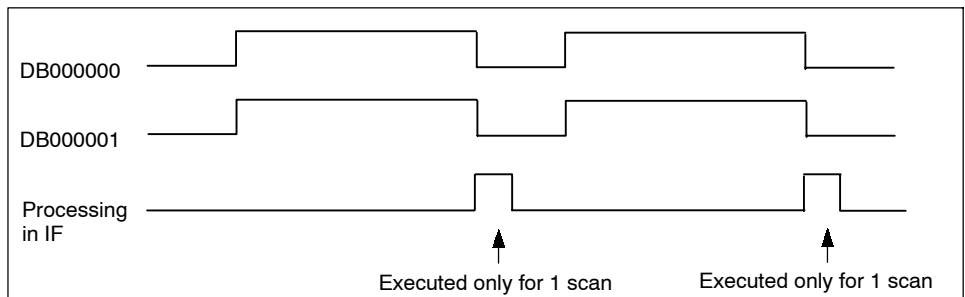
**Using the NON command together with the IF command**

```
IF NON(DB000000 DB000001)==1;
  .
  .
IEND;
```

• Ladder equivalent circuit



• Timing chart



### 6.1.3 ON-DELAY TIMER (TON): Counting unit: 0.01 second

Use in a sequence program	Possible
Use in a motion program	Not possible

#### ■ Overview

The TON command counts the milliseconds (in units of 10 ms) when the bit input is ON. When the counted value is equal to the set value, the bit output will turn ON. If the bit input turns OFF during counting, the timer operation will stop. After the bit input turns ON again, counting starts again from the beginning (0). In the registers for counting, the actual count (in units of 10 ms) is stored.

#### ■ Description

The TON command is designated as follows:

<u>DB000001</u>	<u>=</u>	<u>DB000000</u>	<u>&amp;</u>	<u>TON</u>	<u>(set value</u>	<u>DW00001)</u>
A		B			C	D

Symbol	Application	Usable Register
A	Bit output	Bit register (excluding # and C registers)
B	Bit input	All bit registers
C	Set value	All integer registers Constants (0 to 65535 (655.35S): Every 10 ms)
D	Register for timer counting	All integer registers

**IMPORTANT**

- Milliseconds are not counted while the debugging operation is stopped. Counting starts again from the current counted value after the debugging operation restarts.
- Be sure to designate bit input “DBxxxxxx&.”

6



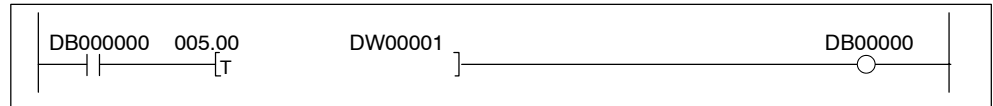
## ■ Programming Examples

The following illustration provides a programming example for the TON command.

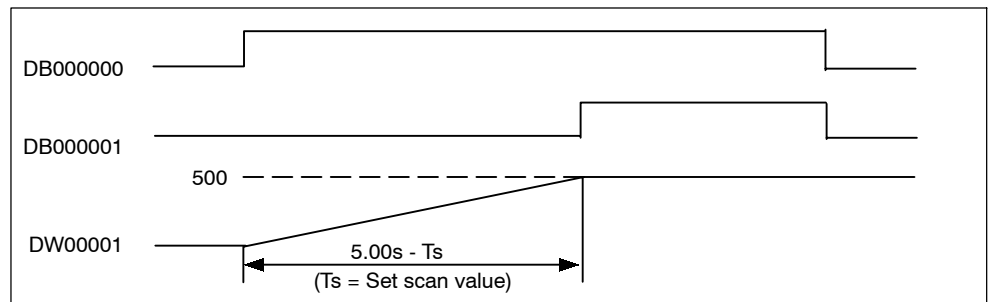
```
DB000001=TON (500 DW00001);
```

↑ Set to 5 seconds

### ● Ladder equivalent circuit



### ● Timing chart



## 6.1.4 OFF-DELAY TIMER (TOF): Counting unit: 0.01 second

Use in a sequence program	Possible
Use in a motion program	Not possible

### ■ Overview

The TOF command counts the milliseconds (in units of 10 ms) when the bit input is OFF. When the counted value is equal to the set value, the bit output will turn OFF. If the bit input turns ON during counting, the timer operation will stop. After the bit input turns OFF again, counting starts again from the beginning (0). In the registers for counting, the actual count (in units of 10 ms) is stored.

## ■ Description

The TOF command is designated as follows:

$$\frac{\text{DB000001}}{\text{A}} = \frac{\text{DB000000}}{\text{B}} \ \& \ \text{TOF} \left( \frac{\text{set value}}{\text{C}} \ \frac{\text{DW00001}}{\text{D}} \right);$$

Symbol	Application	Usable Register
A	Bit output	Bit register (excluding # and C registers)
B	Bit input	All bit registers
C	Set value	All integer registers Constants (0 to 65535 (655.35S): Every 10 ms)
D	Register for timer counting	All integer registers

**IMPORTANT**

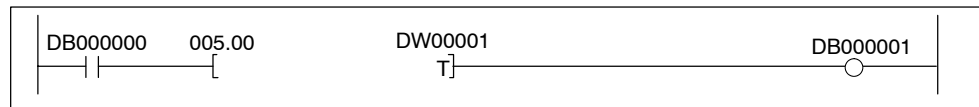
- Milliseconds are not counted while the debugging operation is stopped. Counting starts again from the current counted value after the debugging operation restarts.
- Be sure to designate bit input “DBxxxxx&.”

## ■ Programming Examples

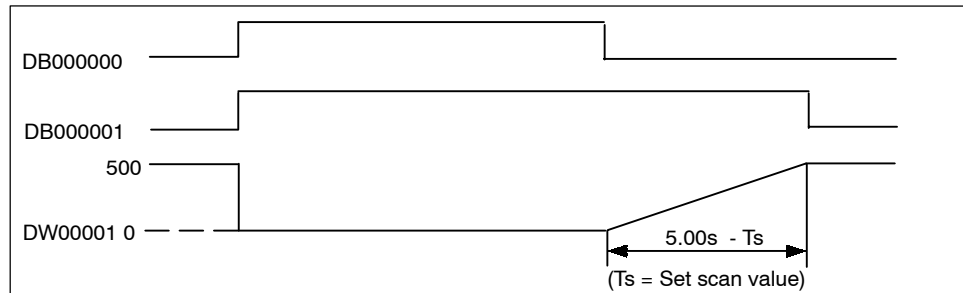
The following illustration provides a programming example for the TOF command.

```
DB000001=DB000000 & TOF (500 DW00001);
```

- Ladder equivalent circuit



- Timing chart



## 6.1.5 USER FUNCTION CALL (FUNC)

Use in a sequence program	Possible
Use in a motion program	Not possible

### ■ Overview

The FUNC command calls user functions (ladder programs) from the sequence program.

### ■ Description

The FUNC command is designated as follows:

**FUNC** Function name Input data 1, input data 2, input data 3 • • •, Input address,  
Output data 1, output data 2, output data 3 • • • ;

Function name: Up to 8 ASCII characters

Input data: Up to 16 data items (at least 1 data item is required)

Input address: Up to 1 address

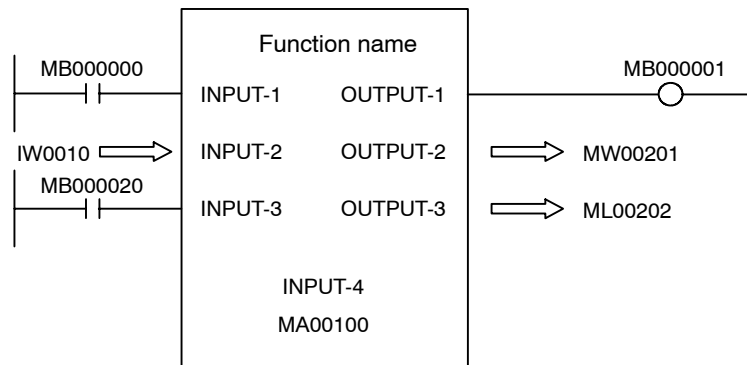
Output data: Up to 16 data items (at least 1 data item is required)

- Notes 1.** Several items for the above input data and output data can be described, respectively. (At least 1 item must be described.) The input address can be omitted. When it is omitted, describe only the comma (,).
- 2.** The FUNC command calls a user function. The execution proceeds to the next block after FUNC whether or not the user function execution has been completed.

### ■ Programming Examples

The following illustration provides a programming example for the FUNC command. In this example, input data 3, input address, and output data 3 are used.

**FUNC** KANSUU MB000000 IW0010 MB000020, MA00100  
Function name                      Input data                      Input address  
MB000001 MW00201 ML00202 ;  
Output data



**Programming Example for USER FUNCTION CALL (FUNC)**

## 6.1.6 SEQUENCE SUBROUTINE CALL (SSEE)

Use in a sequence program	Possible
Use in a motion program	Not possible

### ■ Overview

The SSEE command can call, from the sequence program, a subroutine that has previously been stored in the sequence program memory.

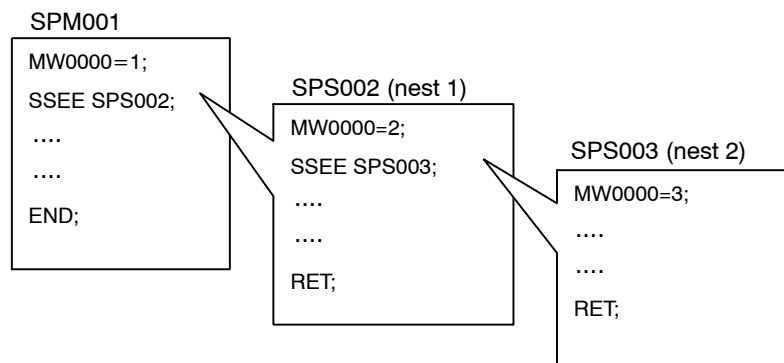
### ■ Description

The SSEE command is designated as follows:

```
SSEE SPS-;  
      Call subroutine number
```

The SSEE command executes the subroutine with the number designated by SPS.

There can be up to 8 nestings for calling subroutines from other subroutines.



SUBROUTINE RETURN (RET) must be designated at the end of the subroutine.

### ■ Programming Examples

The following illustration provides a programming example for the SSEE command (when the sequence subroutine SPS101 is called) .

```
SSEE SPS101;  
      Designate subroutine number.
```

# A

---

## Motion Command List

The table in Appendix A1 shows the motion commands.

A.1 Motion Command List ..... A - 2

## A.1 Motion Command List

The motion commands are listed in the following table.

Classification	Command	Name	Programming Format	Function/Meaning
Axis Move Commands	<b>MOV</b>	POSITIONING	MOV [ <i>axis1</i> ] – [ <i>axis2</i> ] – …; (Up to 14 axes can be designated.)	Executes positioning at rapid traverse speed for up to 14 axes* simultaneously.
	<b>MVS</b>	LINEAR INTERPOLATION	MVS [ <i>axis1</i> ] – [ <i>axis2</i> ] – …F–; (Up to 14 axes can be designated.)	Executes linear travel at interpolation feed speed F for up to 14 axes* simultaneously.
	<b>MCW</b>	CLOCKWISE CIRCULAR INTERPOLATION	MCW [ <i>axis1</i> ] – [ <i>axis2</i> ] – R– F–; MCC [ <i>axis1</i> ] – [ <i>axis2</i> ] – U– V– T– F–;	Executes circular interpolation at tangential speed F for two axes simultaneously following radius R (or designated center point coordinates).  With the center point coordinate designation, multiple circles can be designated with T–. (T– can also be omitted.)
	<b>MCC</b>	COUNTERCLOCKWISE CIRCULAR INTERPOLATION		
	<b>MCW</b>	CLOCKWISE HELICAL INTERPOLATION	MCW [ <i>axis1</i> ] – [ <i>axis2</i> ] –U–V– [ <i>axis3</i> ] –T– F–; MCC [ <i>axis1</i> ] – [ <i>axis2</i> ] –R–[ <i>axis3</i> ] –F–;	Moves three axes simultaneously in a combination of circular interpolation and linear interpolation outside of the circular interpolation plane. Speed F will be the circular interpolation tangential speed.  With the center point coordinate designation, the number of turns can be designated with T–. (T– can also be omitted.)
	<b>MCC</b>	COUNTERCLOCKWISE HELICAL INTERPOLATION		
	<b>ZRN</b>	ZERO POINT RETURN	ZRN [ <i>axis1</i> ] – [ <i>axis2</i> ] – …; (Up to 14 axes can be designated.)	Returns each axis to its zero point.
	<b>SKP</b>	SKIP	SKP [ <i>axis1</i> ]– [ <i>axis2</i> ]– … SS–; (Up to 14 axes can be designated.)	If the SKIP signal turns ON during a linear interpolation operation, skips the remaining movement and proceeds to the next block.
<b>MVT</b>	SET TIME POSITIONING	MVT [ <i>axis1</i> ]– [ <i>axis2</i> ]– … T–; (Up to 14 axes can be designated.)	Executes positioning by clamping the feed speed so that travel can be completed at the designated time.	
<b>EXM</b>	EXTERNAL POSITIONING	EXM [ <i>axis1</i> ]– D–;	When an external positioning signal is input while external positioning is being executed, only the travel distance designated by “D–” is positioned with an incremental value, and then the next command is executed.	

\* Number of axes with simultaneous control for the MP930.

The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to 1.1.2 *Function Performance* for details.

Classification	Command	Name	Programming Format	Function/Meaning
<b>Basic Control Commands</b>	<b>ABS</b>	ABSOLUTE MODE	ABS;	Treats all subsequent coordinate words as absolute values.
	<b>INC</b>	INCREMENTAL MODE	INC;	Treats all subsequent coordinate words as incremental values.
	<b>POS</b>	CURRENT POSITION SET	POS [ <i>axis1</i> ] – [ <i>axis2</i> ] – …;	Changes the current values to the desired coordinate values for up to 14 axes* simultaneously. Subsequent move commands use this new coordinate system.
	<b>PLN</b>	COORDINATE PLANE SETTING	PLN [ <i>axis1</i> ] [ <i>axis2</i> ]	Designates the coordinate plane to be used for a command requiring a plane designation command.
	<b>MVM</b>	MOVE ON MACHINE COORDINATE	MVM MOV [ <i>axis1</i> ]– [ <i>axis2</i> ]–; or MVM MVS [ <i>axis1</i> ]– [ <i>axis2</i> ]–;	Goes to the target position on the machine coordinate system. The coordinate system set automatically on completion of the zero point return is called a machine coordinate system. This coordinate system is not affected by the POS command.
	<b>PLD</b>	PROGRAM CURRENT POSITION UPDATE	PLD [ <i>axis1</i> ] – [ <i>axis2</i> ] – …;	Updates the program current position for axes shifted by manual intervention. Up to 14 axes can be designated.
<b>Speed and Acceleration/Deceleration Commands</b>	<b>ACC</b>	ACCELERATION TIME CHANGE	ACC [ <i>axis1</i> ] – [ <i>axis2</i> ] – …;	Sets the acceleration time for linear acceleration/deceleration for up to 14 axes* simultaneously.
	<b>SCC</b>	S-CURVE TIME CONSTANT CHANGE	SCC [ <i>axis1</i> ] – [ <i>axis2</i> ] – …;	Sets the time constant for moving average acceleration/deceleration for up to 14 axes* simultaneously.
	<b>VEL</b>	SET VELOCITY	VEL [ <i>axis1</i> ] – [ <i>axis2</i> ] – …;	Sets the feed speed for up to 14 axes*.
	<b>IAC</b>	INTERPOLATION ACCELERATION TIME CHANGE	IAC T–;	Sets the acceleration time for linear acceleration/deceleration for interpolation travel.
	<b>IDC</b>	INTERPOLATION DECELERATION TIME CHANGE	IDC T–;	Sets the deceleration time for linear acceleration/deceleration for interpolation travel.
	<b>IFP</b>	INTERPOLATION FEED SPEED RATIO SETTING	IFP P–;	Designates the maximum feed % for the speed designation during an interpolation feed.
	<b>FMX</b>	MAXIMUM INTERPOLATION FEED SPEED SETTING	FMX T–;	Sets the maximum speed during an interpolation feed. The interpolation acceleration time is the time from “0” until this speed is reached.

\* Number of axes with simultaneous control for the MP930.

The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to 1.1.2 *Function Performance* for details.

Classification	Command	Name	Programming Format	Function/Meaning
<b>High-Level Control Commands</b>	<b>PFN</b>	IN-POSITION CHECK	MVS [ <i>axis1</i> ] – [ <i>axis2</i> ] – ... PFN; or PFN [ <i>axis1</i> ] [ <i>axis2</i> ];	Proceeds to the next block after the positioning commanded by the interpolation travel command in the same block or a previous block enters the positioning completion range (parameter setting).
	<b>INP</b>	SECOND IN-POSITION CHECK	INP [ <i>axis1</i> ] – [ <i>axis2</i> ] – ...;	Proceeds to the next block after the positioning subsequently commanded by the interpolation travel command with PFN enters the second positioning completion range.
	<b>SNG</b>	IGNORE SINGLE BLOCK SIGNAL	SNG MVS [ <i>axis1</i> ] 100. [ <i>axis2</i> ] 200. F1000;	A block with this command will be executed continuously, even in single-block operation mode. SNG cannot be designated on its own.
	<b>UFC</b>	USER FUNCTION CALL	UFC <i>Function_name</i> <i>Input_data</i> , <i>Input_address</i> , <i>Output_data</i> ;	Calls a function created by the user.
<b>Sequence Commands</b>	=	SUBSTITUTE	(Result) = (Arithmetic expression)	Substitutes operation results. Performs calculations from left to right (with no order of priority).
	+	ADD	MW– = MW– + MW–; MW– = MW– + 123456; MW– = 123456 + MW–;	Performs integer and real number addition. Calculates combinations of integers and real numbers as real numbers.
	–	SUBTRACT	MW– = MW– – MW–; MW– = MW– – 123456; MW– = 123456 – MW–;	Performs integer and real number subtraction. Calculates combinations of integers and real numbers as real numbers.
	*	MULTIPLY	MW– = MW– * MW–; MW– = MW– * 123456; MW– = 123456 * MW–;	Performs integer and real number multiplication. Calculates combinations of integers and real numbers as real numbers.
	/	DIVIDE	MW– = MW–/MW–; MW– = MW–/123456; MW– = 123456/MW–;	Performs integer and real number division. Calculates combinations of integers and real numbers as real numbers.
	<b>MOD</b>	REMAINDER	MW– = MW–/MW–; MW– = MOD;	When programmed in the next block after a division, MOD stores the remainder in the designated register.



Classification	Command	Name	Programming Format	Function/Meaning
Sequence Commands		OR (logical OR)	MB- = MB-   MB-; MB- = MB-   1; MW- = MW-   MW-; MW- = MW-   H00FF;	Performs bit/integer logical OR.
	^	XOR (logical exclusive OR)	MW- = MW- ^ MW-; MW- = MW- ^ H00FF;	Performs integer logical exclusive OR.
	&	AND (logical AND)	MB- = MB- & MB-; MB- = MB- & 1; MW- = MW- & MW-; MW- = MW- & H00FF;	Performs bit/integer logical AND.
	!	NOT (logical complement)	MB- = !MB-; MB- = !1; MW- = !MW-; MW- = !H00FF;	Performs bit/integer logical complement (inverts bits).
	()	PARENTHESES	MW- = MW- & (MW-   MW-);	The logical arithmetic expression inside parentheses is calculated first.
	S{ }	SET BIT	S{MB-} = MB- & MB-;	If the logical operation result is "true," the designated bit turns ON. The designated bit does not turn OFF, even if the logical operation result is "false."
	R{ }	RESET BIT	R{MB-} = MB- & MB-;	If the logical operation result is "true," the designated bit turns OFF. The designated bit does not turn ON, even if the logical operation result is "false."
	SIN	SINE	SIN (MW-); SIN (90);	Obtains the sine of the integer or real number (deg), and returns a real value.
	COS	COSINE	COS (MW-); COS (90);	Obtains the cosine of the integer or real number (deg), and returns a real value.
	TAN	TANGENT	TAN (MF-); TAN (45.0);	Obtains the tangent of the real number (deg), and returns a real value.
	ASN	ARC SINE	ASN (MF-); ASN (45.0);	Obtains the arc sine of the real number (deg), and returns a real value.
	ACS	ARC COSINE	ACS (MF-); ACS (90.0);	Obtains the arc cosine of the real number (deg), and returns a real value.

Classification	Command	Name	Programming Format	Function/Meaning
Sequence Commands	<b>ATN</b>	ARC TANGENT	ATN (MW-); ATN (45);	Obtains the arc tangent of the integer or real number (deg), and returns a real value.
	<b>SQRT</b>	SQUARE ROOT	SQT (MW-); SQT (100);	Obtains the square root of the integer or real number, and returns a real value.
	<b>BIN</b>	BCD-TO-BINARY	BIN (MW-);	Converts BCD data to binary data.
	<b>BCD</b>	BINARY-TO-BCD	BCD (MW-);	Converts binary data to BCD data.
	<b>= =</b>	MATCH	IF MW- = = MW-; WHILE MW- = = MW-;	Used in an IF or WHILE conditional expression. If the left side and right side match, the condition is "true."
	<b>&lt; &gt;</b>	MISMATCH	IF MW- < > MW-; WHILE MW- < > MW-;	Used in an IF or WHILE conditional expression. If the left side and right side do not match, the condition is "true."
	<b>&gt;</b>	GREATER THAN	IF MW- > MW-; WHILE MW- > MW-;	Used in an IF or WHILE conditional expression. If the left side is greater than the right side, the condition is "true."
	<b>&lt;</b>	LESS THAN	IF MW- < MW-; WHILE MW- < MW-;	Used in an IF or WHILE conditional expression. If the left side is less than the right side, the condition is "true."
	<b>&gt; =</b>	GREATER THAN OR EQUAL TO	IF MW- >= MW-; WHILE MW- >= MW-;	Used in an IF or WHILE conditional expression. If the left side is greater than or equal to the right side, the condition is "true."
	<b>&lt; =</b>	LESS THAN OR EQUAL TO	IF MW- <= MW-; WHILE MW- <= MW-;	Used in an IF or WHILE conditional expression. If the left side is less than or equal to the right side, the condition is "true."
	<b>SFR</b>	RIGHT SHIFT	SFR MB- N- W-;	Shifts only the designated number of word variables to the right.
	<b>SFL</b>	LEFT SHIFT	SFL MB- N- W-;	Shifts only the designated number of word variables to the left.
	<b>BLK</b>	BLOCK MOVE	BLK MW- MW- W-;	Moves the block (constant designation) beginning with the designated bit (word) variable.
<b>CLR</b>	CLEAR	CLR MB- W-;	Sets the number of constants specified in the variable group beginning with the designated bit (word) variable to OFF ("0").	

Classification	Command	Name	Programming Format	Function/Meaning
Control Commands	<b>MSEE</b>	SUBROUTINE CALL	MSEE MPS- ;	Executes the MPS- subroutine.
	<b>TIM</b>	DWELL TIME	TIM T-;	Waits for the period of time specified by T, and then proceeds to the next block.
	<b>IOW</b>	I/O WAIT	IOW MB- = = ***;	Stops execution of the motion program until the conditional expression given in the command is satisfied.
	<b>END</b>	PROGRAM END	END;	Ends the motion program.
	<b>RET</b>	SUBROUTINE RETURN	RET;	Ends the subroutine.
	<b>EOX</b>	ONE SCAN WAIT	EOX;	Separates continuous sequence instructions and forces a wait of one scan before continuing execution.
	<b>IF ELSE IEND</b>	Branching commands	IF (conditional expression) ; (process 1) ELSE; (process 2) IEND;	Executes process 1 if the conditional expression is satisfied, and executes process 2 if the conditional expression is not satisfied.
	<b>WHILE WEND</b>	Repeat commands	WHILE (conditional expression) ; ... WEND;	Repeatedly executes WHILE to WEND processing for as long as the conditional expression is satisfied.
	<b>PFORK JOINTO PJOINT</b>	Parallel execution commands	PFORK label 1, label 2,...; Label 1: Process 1 JOINTO label X Label 2: Process 2 JOINTO label X Label • • Label X: PJOINT;	Executes the blocks designated by the labels in parallel. With a subroutine, a maximum of two labels can be designated. Also, a motion command cannot be used in the block designated by the second label.  END and RET cannot be used during parallel execution processing.
<b>SFORK JOINTO SJOINT</b>	Selective execution commands	SFORK conditional expression 1? label 1, Conditional expression 2? label 2,...; Label 1: Process 1 JOINTO label X Label 2: Process 2 JOINTO label X Label • • Label X: SJOINT;	Executes process 1 if conditional expression 1 is satisfied, and executes process 2 if the conditional expression 2 is satisfied.	

# B

---

## Motion Program Alarm List



The table in Appendix B shows the motion program alarms.

B.1 Motion Program Alarm Storage Location .	B -2
B.2 Motion Program Alarm List .....	B -4

## B.1 Motion Program Alarm Storage Location

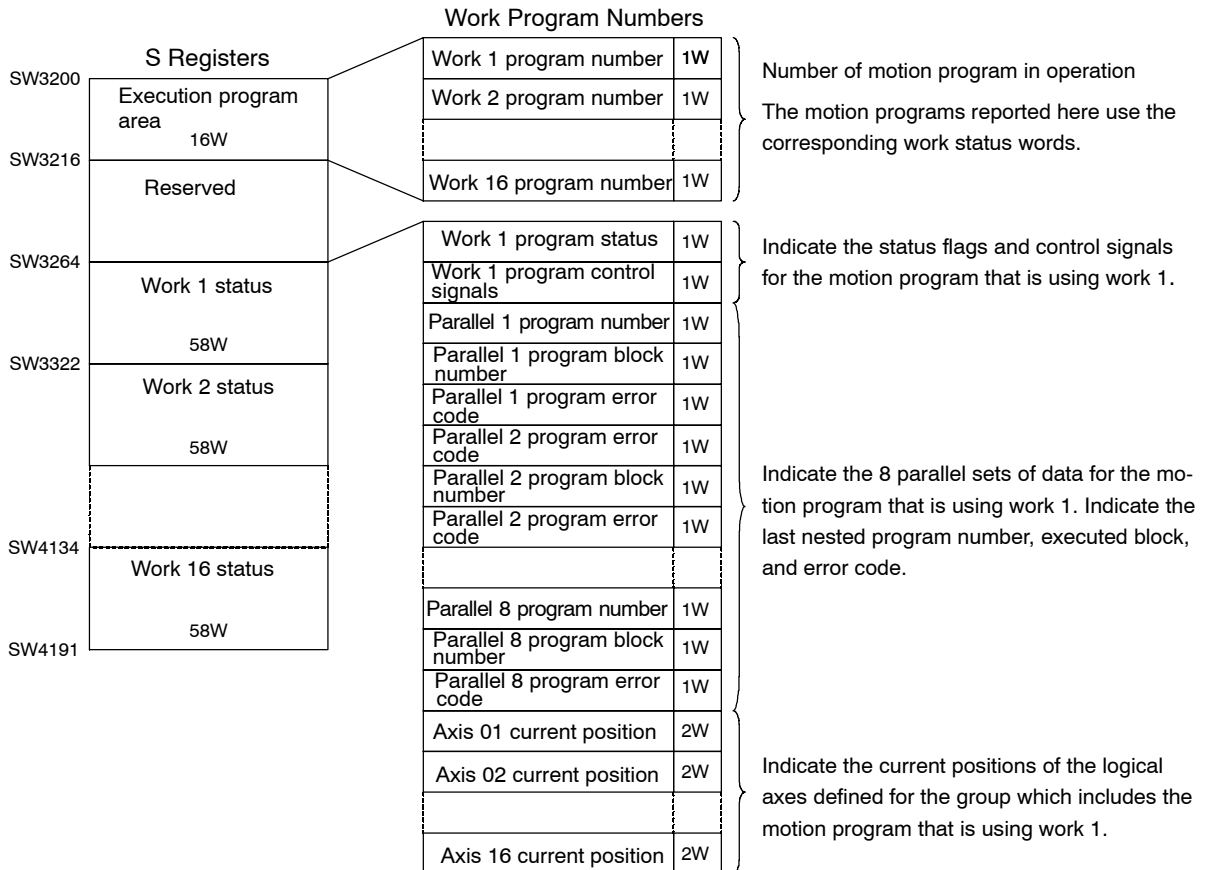
### ■ MP900-series Machine Controllers

Motion program alarms stored in the alarm output register set by the Group Definition settings are displayed. Set the leading register number for outputting alarms. The number of continuous alarm registers containing alarms starting from the set leading register will be used as the number of parallel processes used for the motions programs in the group. For example, if MW00010 is defined as the leading alarm output register and four parallel processes will be used, then registers MW00010 to MW00013 will be used in the same order as the number of the parallel process.

The POS command is designated as follows:

### ■ MP2000-series Machine Controllers

In the MP2000-series Machine Controllers, the status of the motion program is reported in the S registers. The following figure shows the S-registers used by the motion program as well as details on the work status.



1. Work Status used by the Motion Program

The main motion program uses a single work status area during operation. The motion program alarms of the motion program and the current positions of the logical axes are displayed in the work status area.

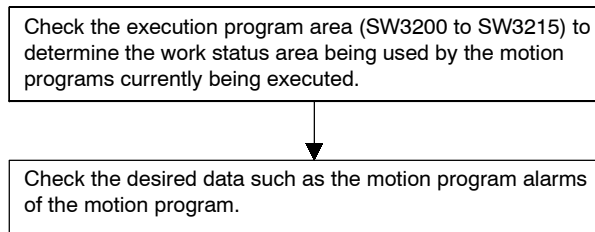
● Procedure when Specifying the Work Status Area

To specify the work status area used by the motion program, turn ON bit 13 (System work number setting) in the second word of the MSEE instruction's work registers. When the System work number setting is turned ON, the motion program will use the work status area set in the fourth word (System work number) of the MSEE instruction's work registers. If the specified system work number is out of range or already allocated, bit 14 ("No system work" error) will be turned ON in the first word of the work registers.

● Procedure when Not Specifying the Work Status Area

The work status areas are used in the order that the motion programs are started, beginning from the work 1 status. When two or more motion programs are started in the same scan, the work status areas are used in the order that the MSEE instructions are called. Each work status area is released when the corresponding motion program ends.

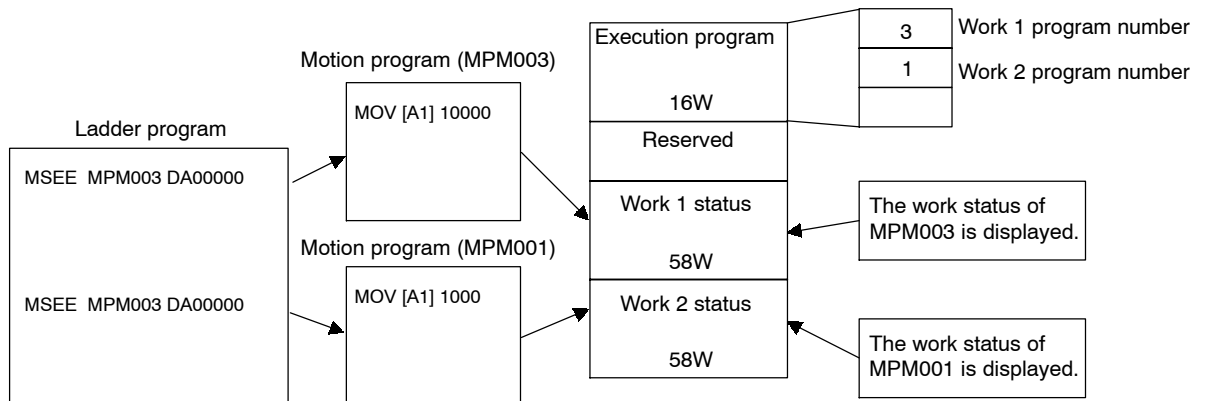
Follow the steps outlined below to view the motion program alarms of the motion program and the current positions of the logical axes.



◀EXAMPLE▶

2. Executing Multiple Motion Programs

In the following example, motion program MPM001 is started after motion program MPM003 is started.



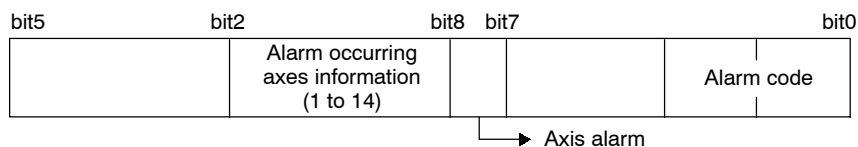
When MPM003 has been started, the number 3 is reported to the work 1 program number (SW3200) in the execution program area and MPM003 uses the work 1 status area. The motion program alarms and current positions of the logical axes can be checked by reading the work 1 status information.

When MPM001 is started after MPM003 has started, the number 1 is reported to the work 2 program number (SW3201) in the execution program area and MPM001 uses the work 2 status area. The motion program alarms and current positions of the logical axes can be checked by reading the work 2 status information.

## B.2 Motion Program Alarm List

The following table shows the hexadecimal codes that are used for motion program alarms.

Change the display mode to HEX (H).



The following table describes the motion program alarm codes and the corrective actions.

Alarm Code	Alarm Name	Alarm Contents	Corrective Action
02h	Division Error	Division is made with data 0.	Recheck the motion program.
10h	Improper Designation of One Cycle at Radius Designation	Designation of Number of Turns (T) is specified in an arc interpolation command or helical interpolation command by radius designation.	<ul style="list-style-type: none"> <li>Change radius designation to center coordinate designation to execute arc interpolation command or helical command.</li> <li>Do not specify the number of turns.</li> </ul>
11h	Interpolation Feeding Speed Exceeded	Interpolation feeding speed designation is set exceeding the range specified by FMX command.	Set again interpolation feeding speed designation of interpolation command.
12h	Interpolation Feeding Speed Not specified	Interpolation feeding speed designation is not set at all. (Setting it even once, it can be omitted in the same motion program.)	Specify interpolation feeding speed in interpolation command.
13h	Range Exceeded after Acceleration Parameter Conversation	Acceleration speed given by indirect designation is set exceeding the setting range.	Change the register value where indirect designation is done.
14h	Arc Length Exceeded LONG_MAX.	Arc length designation exceeds the setting range in arc interpolation command or helical interpolation command.	Recheck arc length designation in arc interpolation command or interpolation command or helical interpolation command.
15h	Vertical Axis Not Specified at Arc Plane Designation	Vertical axis designation is not set in arc interpolation command or helical interpolation command.	Specify the axis by PLN command.
16h	Horizontal Axis Not Specified at Arc Plane Designation	Horizontal axis designation is not set in arc interpolation command or helical interpolation command.	Specify the axis by PLN command.
17h	Excessive Specified Axes	Excessive number of axes is set in arc interpolation command (2 axes) or helical interpolation command (3 axes).	Set again axis designation for arc interpolation command or helical interpolation command.
18h	Excessive Number of Turns Specified	Number of specified turns exceeds the setting range in arc interpolation command or helical interpolation command.	Set again the number of turns for arc interpolation command or helical interpolation command.

<b>Alarm Code</b>	<b>Alarm Name</b>	<b>Alarm Contents</b>	<b>Corrective Action</b>
<b>19h</b>	Radius Exceeds LONG_MAX.	Radius designation exceeds the setting range in arc interpolation command or helical interpolation command.	Recheck radius designation for arc interpolation command or helical interpolation command.
<b>1Ah</b>	Improper Center Point Designation Error	Center point is not specified correctly in arc interpolation command or helical interpolation command.	Specify the center point correctly for arc interpolation command or helical interpolation command.





Date of Publication	Rev. No.	WEB Rev. No.	Section	Revised Content
June 2000	③	0	All chapters	Partly revised
July 1999	②	0	All chapters	Partly revised
December 1998	①	0	Back cover	Revision: Address
July 1998	–	–	–	First edition

# Machine Controller MP900/MP2000 Series

# USER'S MANUAL

# MOTION PROGRAMMING

---

**IRUMA BUSINESS CENTER (SOLUTION CENTER)**

480, Kamifujisawa, Iruma, Saitama 358-8555, Japan  
Phone 81-4-2962-5151 Fax 81-4-2962-6138  
<http://www.yaskawa.co.jp>

**YASKAWA AMERICA, INC.**

2121 Norman Drive South, Waukegan, IL 60085, U.S.A.  
Phone 1-800-YASKAWA (927-5292) or 1-847-887-7000 Fax 1-847-887-7310  
<http://www.yaskawa.com>

**YASKAWA ELÉTRICO DO BRASIL LTDA.**

Avenida Piraporinha 777, Diadema, São Paulo, 09950-000, Brasil  
Phone 55-11-3585-1100 Fax 55-11-3585-1187  
<http://www.yaskawa.com.br>

**YASKAWA EUROPE GmbH**

Hauptstraße 185, Eschborn 65760, Germany  
Phone 49-6196-569-300 Fax 49-6196-569-398  
<http://www.yaskawa.eu.com>

**YASKAWA ELECTRIC KOREA CORPORATION**

9F, Kyobo Securities Bldg. 26-4, Yeouido-dong, Yeongdeungpo-gu, Seoul, 150-737, Korea  
Phone 82-2-784-7844 Fax 82-2-784-8495  
<http://www.yaskawa.co.kr>

**YASKAWA ELECTRIC (SINGAPORE) PTE. LTD.**

151 Lorong Chuan, #04-02A, New Tech Park 556741, Singapore  
Phone 65-6282-3003 Fax 65-6289-3003  
<http://www.yaskawa.com.sg>

**YASKAWA ELECTRIC (CHINA) CO., LTD.**

12F, Carlton Bld., No.21 HuangHe Road, HuangPu District, Shanghai 200003, China  
Phone 86-21-5385-2200 Fax 86-21-5385-3299  
<http://www.yaskawa.com.cn>

**YASKAWA ELECTRIC (CHINA) CO., LTD. BEIJING OFFICE**

Room 1011, Tower W3 Oriental Plaza, No.1 East Chang An Ave.,  
Dong Cheng District, Beijing 100738, China  
Phone 86-10-8518-4086 Fax 86-10-8518-4082

**YASKAWA ELECTRIC TAIWAN CORPORATION**

9F, 16, Nanking E. Rd., Sec. 3, Taipei 104, Taiwan  
Phone 886-2-2502-5003 Fax 886-2-2505-1280




YASKAWA ELECTRIC CORPORATION

In the event that the end user of this product is to be the military and said product is to be employed in any weapons systems or the manufacture thereof, the export will fall under the relevant regulations as stipulated in the Foreign Exchange and Foreign Trade Regulations. Therefore, be sure to follow all procedures and submit all relevant documentation according to any and all rules, regulations and laws that may apply.

Specifications are subject to change without notice for ongoing product modifications and improvements.

© 1998-2013 YASKAWA ELECTRIC CORPORATION. All rights reserved.

MANUAL NO. SIEZ-C887-1.3D

Published in Japan July 2013 98-7 -0  
13-6-9