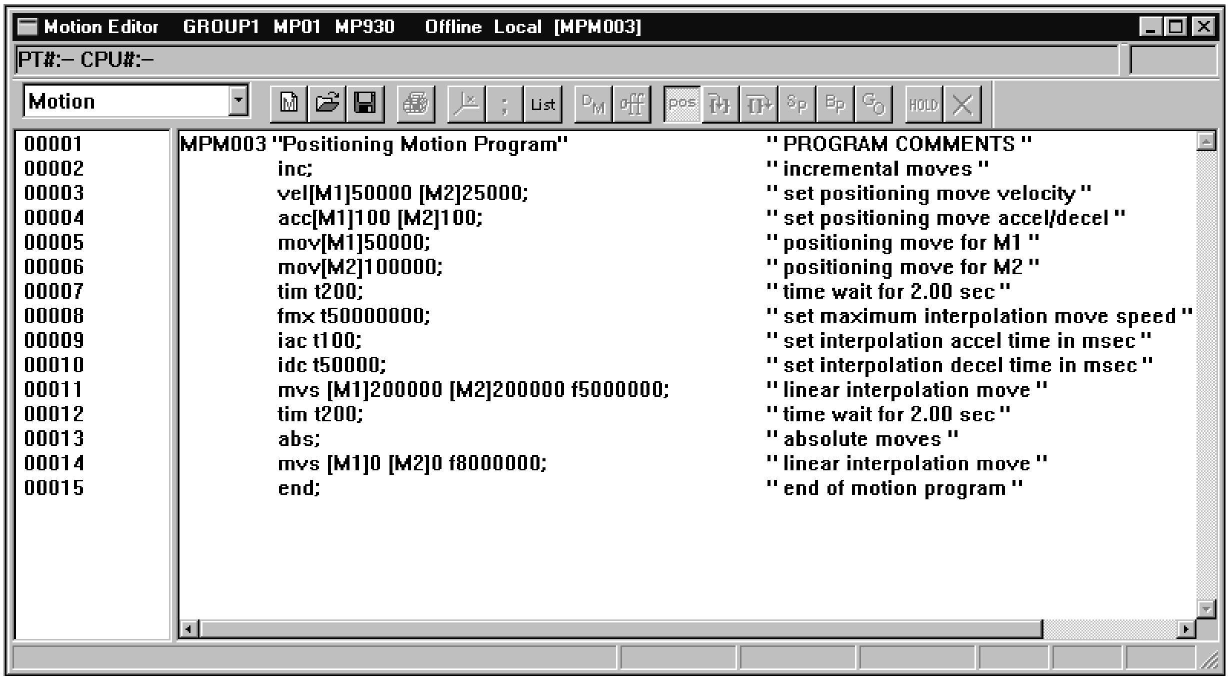
YASKAWA

A World of Automation Solutions™



```
MPM003 "Positioning Motion Program"              " PROGRAM COMMENTS "
          inc;                                    " incremental moves "
          vel[M1]50000 [M2]25000;                 " set positioning move velocity "
          acc[M1]100 [M2]100;                     " set positioning move accel/decel "
          mov[M1]50000;                           " positioning move for M1 "
          mov[M2]100000;                          " positioning move for M2 "
          tim t200;                               " time wait for 2.00 sec "
          fmx t50000000;                          " set maximum interpolation move speed "
          iac t100;                               " set interpolation accel time in msec "
          idc t50000;                             " set interpolation decel time in msec "
          mvs [M1]200000 [M2]200000 f5000000;     " linear interpolation move "
          tim t200;                               " time wait for 2.00 sec "
          abs;                                    " absolute moves "
          mvs [M1]0 [M2]0 f8000000;               " linear interpolation move "
          end;                                    " end of motion program "
```

MotionSuite™ Series Machine Controller Programming Manual

Outline of Manual

- This manual is a collection of data regarding the design and maintenance of the MotionSuite™ series machine controller. The following items are included in this manual:
  - Product outline, specifications and programming methods
  - Basic programming
  - Advanced programming
- Read this manual thoroughly so as to ensure proper use of the controller. Furthermore, store this manual properly so that it can be referenced whenever necessary.

Related Manuals

- Related manuals are shown in the following table.
- Use this product with full knowledge of the product specifications, usage limits, etc.

Document Name	Document Number	Content
MP930 Machine Controller Hardware Manual	• YEA-SIA-C887-1.1B	Describes in detail the functions, specifications and usage methods of the MP930 • Functions/Specifications • Setup procedures, etc.
MP930 Machine Controller Ladder Programming	• SIEZ-C887-1.2	Describes in detail the operation commands used in MP930 ladder programming
MotionSuite™ Series Machine Controller Software Manual	• YEA-SIA-C887-1.4B	Describes in detail the process control commands used in MotionWorks™ software

Using this And Manual

■ Users of this manual

This manual is to be used by the following personnel:

- Persons designing MotionSuite™ systems
- Persons writing MotionSuite™ motion programs

■ Abbreviations

The following abbreviations are used in this manual:

- MC Unit: MC unit used is the MotionSuite™ motion controller
- I/O Unit: I/O unit used is the I/O expansion module (model: JEPMC-IO350)
- PP: Programming Panel
- PC: Personal Computer

Safety Notes

This chapter deals with the cautionary items regarding the safe and proper use of this device. Be sure to thoroughly read the directions in this manual and all associated materials prior to mounting, running, storage/inspection, and then execute the contents of these manuals correctly. Use the MotionSuite™ series machine controller after thorough study of all device data, safety information, and cautionary items.

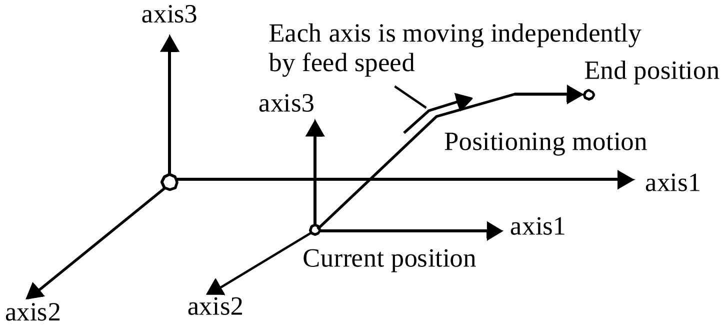
■ Cautions on Usage

⚠ CAUTION

- **During programming of the following axis motion commands, be sure to check the move path to make sure that the tool does not interfere with the work.**

  Commands requiring such checks:
  - Positioning (MOV) commands
  - Linear Interpolation (MVS) commands
  - Circular Interpolation (MCC, MCW) commands
  - Helical Interpolation (MCC, MCW) commands
  - Time Designated Positioning (MVT) commands
  - Skip (SKP) commands

Example



Move Path Based on the MOV Command

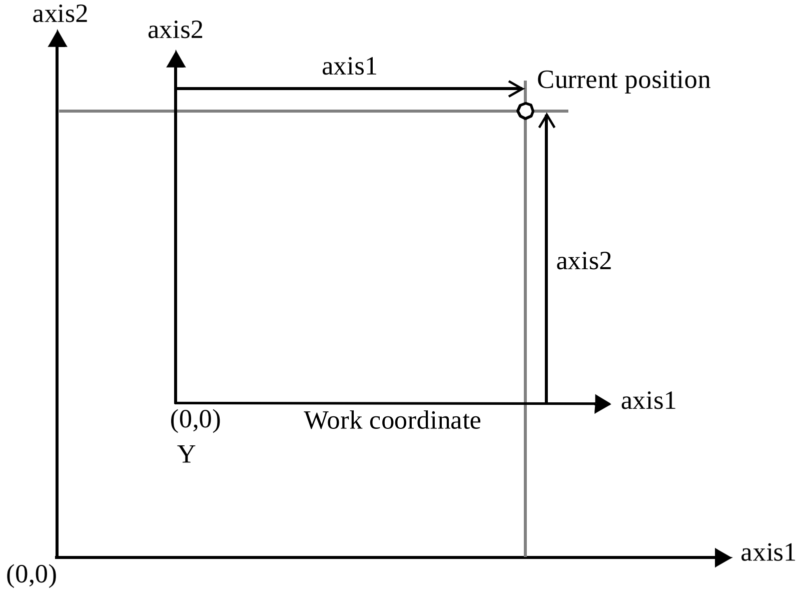
Forgetting this check may result in tool damage, or bodily injury.

⚠ CAUTION

- **If the following commands are erroneously designated, the subsequent motion operation will be completely incorrect. Verify prior to running that these commands have been correctly designated**
  The following commands require such checks:

  - Absolute mode (ABS)
  - Incremental mode (INC)
  - Current value change (POS)
  - Machine coordinate designation (MVM)

Example



POS Command (current value change)

Forgetting this step may result in tool damage, or bodily injury.

■ General Cautionary Items

Cautions During Use

- The MotionSuite™ was neither designed nor manufactured for use in devices or systems under such critical conditions as follow: Transportation systems, medical devices, aerospace, nuclear power control, submarine relay devices, etc. Please contact Yaskawa when considering any such special application.
- Although the MotionSuite™ is manufactured under rigorous quality control, put in place safety apparatus so that a major accident cannot occur when applying the MotionSuite™ in an installation where the occurrence of major facilities damage or serious, life-threatening injury is anticipated due to the failure of the MotionSuite™.
- The pictures and diagrams in this manual are representative examples, and may differ from the product received.
- These manuals may be changed as needed due to product improvements, specification change, or for improvement in ease of use of the manual.
- These changes are made following updating of the document number of the manual, and its issuance as a revised edition. The publication number of the revised edition is written on the manual cover.
- When ordering new manuals due to damage or loss, contact a Yaskawa dealer or the nearest Yaskawa corporate office listed on the cover, and give the document number.
- If the nameplate mounted on the product becomes illegible or damaged, order another nameplate from a Yaskawa dealer or Yaskawa corporate office listed on the cover.
- Any product modified by the customer fall beyond Yaskawa’s product warranty. Yaskawa shoulders no responsibility for any injury or damage resulting from modified products.

1 Motion Programming Outline

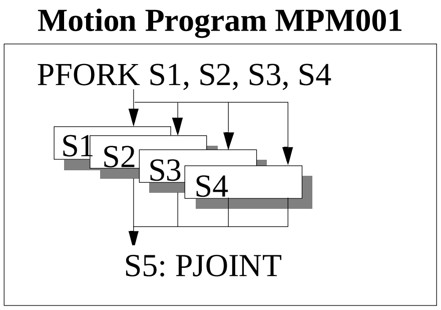
This chapter deals with the methods of creating motion programs. Motion programs are created using MotionWorks™ (Programming Unit); the programs are executed after transfer to the MotionSuite™ series machine controller.

1.1 What is a Motion Program?

A general description of motion programming is presented in this chapter. Be sure to read this section before doing any programming.

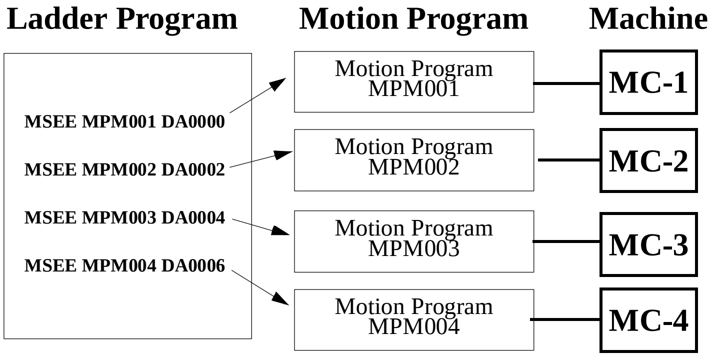
1.1.1 Capabilities of Motion Programs

Using MotionSuite™, it is possible to program the specific motions necessary for industrial machines. The main characteristics of motion programs are provided below for reference:



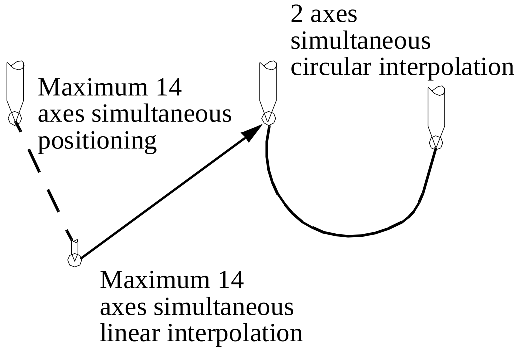
Parallel Operation 1

- A maximum of four programs can be operated in parallel using the PFORK command within a single motion program.
- Axes may be freely combined in up to four groups.



Simultaneous Control of Multiple Machines

- Several programs can be operated in parallel.
- A maximum of 14 axes can be controlled.



Enhanced Motion Commands

• Positioning	14 axes maximum
• Linear Interpolation	14 axes maximum
• Circular Interpolation	2 axes
• Helical Interpolation	3 axes

Example of Operation Commands:

```
MW81D=AB01H*SIN(100)+50;
MW5=BCD (1W0001);
IF CF476  <> DF897;
IF MW0001 > 7;
IF 0B0 == 1;
0B0 = (IB0 | IB2 | IB3) & 1B1;
MW6 = SQT (MW809);
```

Calculations are Flexible and Dependable

- Integer arithmetic operations
- Real number arithmetic operations
- Logical operations
- Trigonometric operations
- Exponents
- Logarithms
- etc.

Conditional Branching Commands

```
    IF <Conditional>
          •
          •     Processed when conditions established
          •
          •
    ELSE
          •
          •     Processed when conditions not established
          •
          •
    IEND
```

Repeat Commands

```
    WHILE <Conditional>
          •
          •     Processing
          •
          •
    WEND
```

Control Commands

- Conditional branching commands (IF/ELSE)
- Repeat commands (WHILE)
- Timer commands (TIM)
- Subroutines (MSEE)
- Parallel execution commands (PFORK)
- Selection execution commands (SFORK)

1.1.2 Basic Program Structure

a. Motion programs are written in a text format motion language. Up to 256 of these motion programs can be created separately.

b. Motion programs are of the following two types: Main programs (MPM□□□) which can be called out from DWG.H, and sub-programs (MPS□□□) which can be called out from the main program.

Important Point

Numbers of the MPM and MPS programs cannot be duplicated.

Motion Program Types

Classification	Designation Method	Characteristics	Number of Programs
Main Program	MPM□□□ 1~256	Can be called out from an H drawing	A maximum of 256 combined main programs and sub-programs can be created.
Sub-Program	MPS□□□ 1~256	Can be called out from a main program	

c. Motion Program Execution Processing Format
Always refer to the motion program from the H drawing by using the MSEE command. H drawings can be referenced from source drawings, sub-drawings, or sub-sub-drawings.

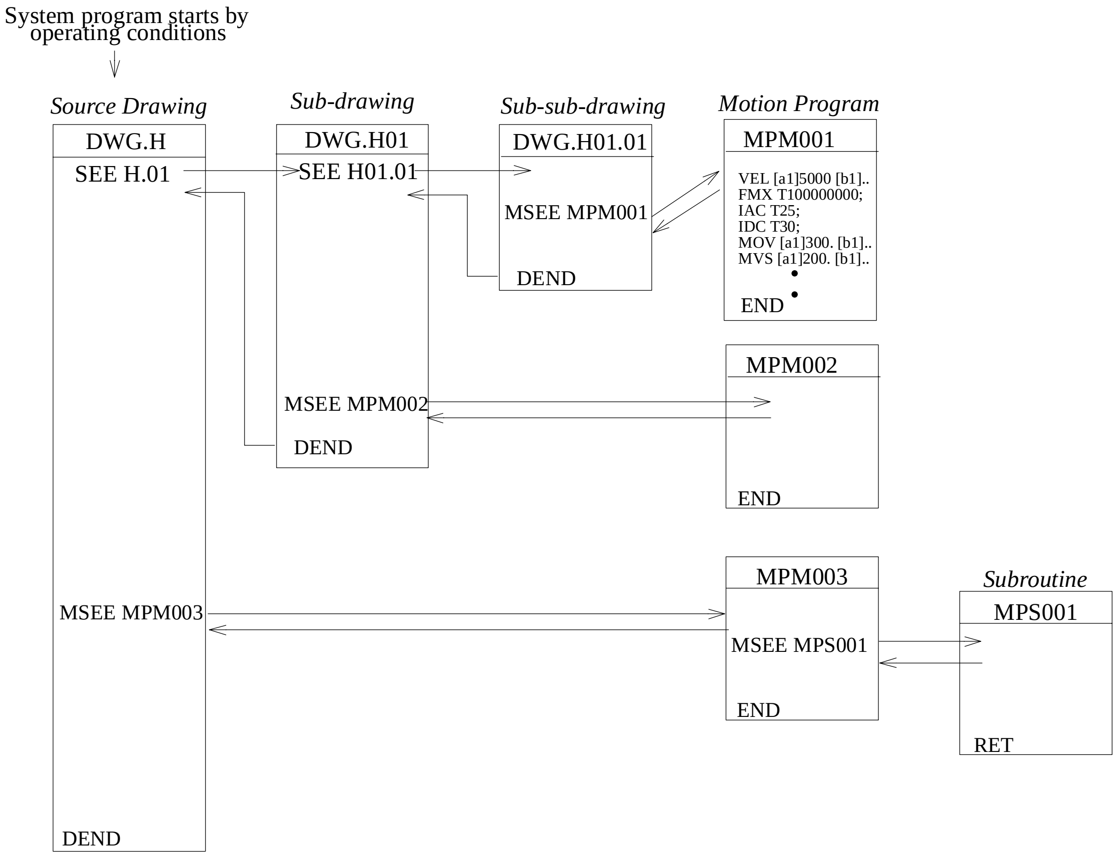


Figure 1.1: Motion Program Execution Processing Format

Supplement

See Section 3.4 “User Programs” in the MP930 Machine Controller Hardware Manual for details regarding this figure.

1.1.3 Function Performance List

The MP9xx motion program function specifications are as follows:

MP9xx Motion Control Function Specifications

Item		Specifications
Number of Control Axes		1~14 maximum
Control Specs	Position Control	Linear, rotational, unlimited, independent axis
	Interpolation	Linear: 14 axes, Circular: 2 axes, Helical: 3 axes
	Speed Control	None
	Torque Limit	Limited (torque limit set by parameters only)
Command Unit		mm, inch, deg, pulse
Minimum Command Setting Unit		1, 0.1, 0.01, 0.001, 0.0001, 0.00001
Maximum Command Value		-2147483648~+2147483647 (with 32-bit symbols)
Speed Command Unit		mm/min, inch/min, deg/min, pulse/min
Accel/Decel Type		Linear, S-curve, separate accel/decel
Override Functions		Positioning: 0.01~100.00% of axis unit Interpolation: 0.01~100.00% of group unit
Coordinates		Cartesian Coordinates
Zero-Point Return		4 Types: Torque+C-phase, zero-point LS, torque+zero-point LS, C-phase There is a zero-point setting function
Programs	Language	Dedicated motion language
	Number of Tasks	A maximum of four programs can be simultaneously executed in parallel.
	Number of Programs	256 maximum
	Program Volume	80 Kbytes (characters) (Adjustable by the volume of ladder program used: 100 Kbyte maximum)
Applied Servo Amplifier		SGD-□□□N/SGDB-□□AN
Encoder		Incremental/Absolute
Command Language		Axis Motion Commands : 8 types MOV, MVS, MCW, MCC, ZRN, SKP, MVT, EXM Basic Control Commands : 5 types ABS, INC, POS, PLN, MVM Speed/Accel/Decel Commands : 7 types ACC, SCC, VEL, IAC, IDC, IFP, FMX Advanced Control Commands : 4 types PFN, INP, SNG, UFC Control Commands : 9 types MSEE, TIM, IOW, END, RET, IF ELSE IEND, WHILE WEND, PFORK JOINTO PJOINT, SFORK JOINTO SJOINT Operation/Sequence Control Commands : 36 types =, +, -, *, /, MOD, |, ^, &, !, (), S{}, R{}, SIN, COS, TAN, ASN, ACS, ATN, SQRT, BIN, BCD, ==, <>, >, <, >=, <=, TON, TOF, SFR, SFL, PON, NON, BLK, CLR

1.1.4 Motion Program Start

The motion program is started from an “H” drawing ladder program. Start is initiated by motion program start commands (MSEE) from within the ladder program and a low to high transition of the program run start request bit. Motion programs are designated directly by the program number and indirectly by the register number containing the program number.

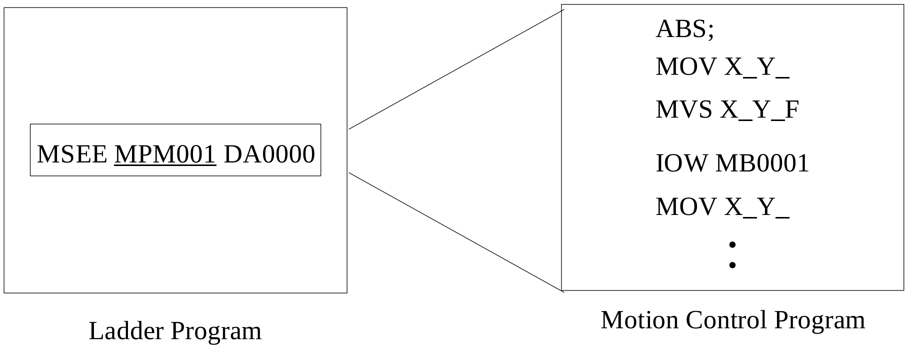


Figure 1.2: Motion Program Start by Direct Designation

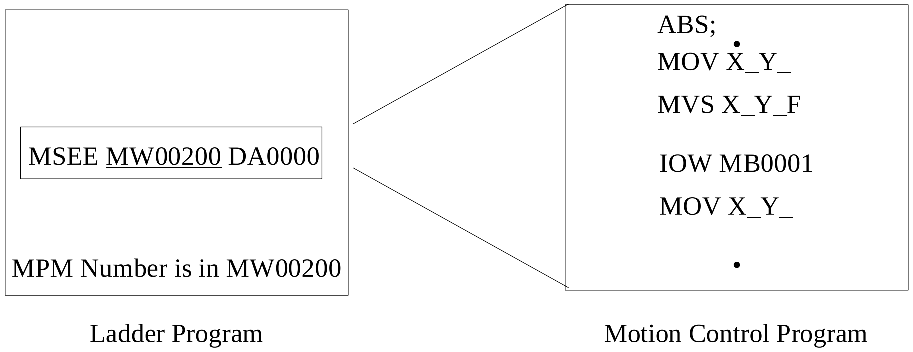
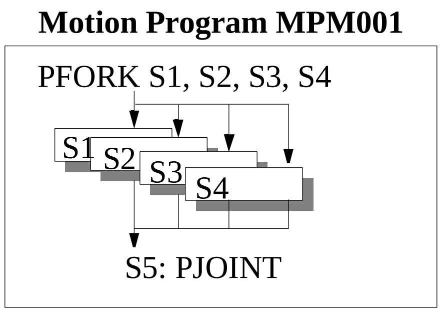


Figure 1.3: Motion Program Start by Indirect Designation

1.1.5 Parallel Program Operation

With MP9xx, it is possible to program freely with various machine activities, due to capabilities for parallel running that make complex motion control possible. Parallel running of motion programs is available in the two following forms.

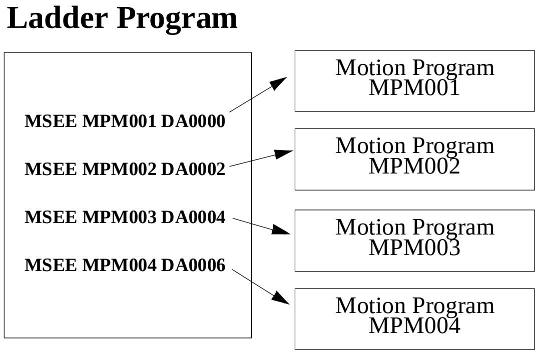
a. With motion program PFORK commands, parallel running of a maximum of 4 programs within 1 program is possible.



Parallel Operation 1

- A maximum of four (4) programs can be operated in parallel using the PFORK command within a single motion program. The PFORK command must be ended with the PJOINT command.

b. With ladder program MSEE commands, parallel running of multiple motion programs is possible. (When automatically generated with MotionWorks™ group setting display, parallel running of a maximum of 4 programs is possible.)



Parallel Operation 2

- Several programs can be operated in parallel by using the ladder program MSEE command.

1.1.6 Program Editor

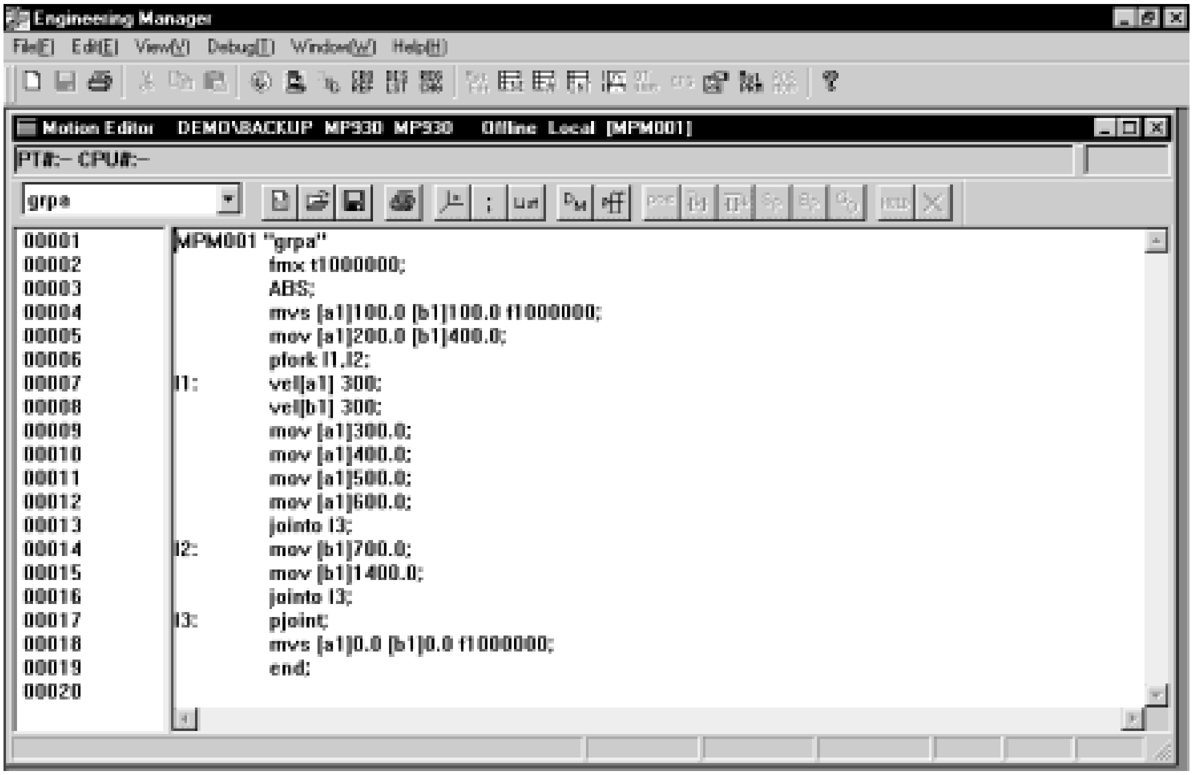
The motion program editor is generated on the MotionWorks™ (programming device) motion program editor display. The editing display contains the following functions.

a. The same functions as the text editor, such as cut & paste, look-up, replace, and jump

b. Special functions such as debugging operations and program instruction monitoring

c. Function for importing and reading text editor files

d. Function for writing and exporting motion program files as text files.



Engineering Manager
File(F) Edit(E) View(V) Debug(T) Window(W) Help(H)
Motion Editor DEMO\BACKUP MP930 MP930 Offline Local [MPM001]
PT#:- CPU#:-
grpa

```
MPM001 "grpa"
            fmx t1000000;
            ABS;
            mvs [a1]100.0 [b1]100.0 f1000000;
            mov [a1]200.0 [b1]400.0;
            pfork l1,l2;
l1:         vel[a1] 300;
            vel[b1] 300;
            mov [a1]300.0;
            mov [a1]400.0;
            mov [a1]500.0;
            mov [a1]600.0;
            jointo l3;
l2:         mov [b1]700.0;
            mov [b1]1400.0;
            jointo l3;
l3:         pjoint;
            mvs [a1]0.0 [b1]0.0 f1000000;
            end;
```

Figure 1.4: MotionWorks™ Motion Program Editor Display

Supplement

For motion program editor function details, please refer to the MotionSuite™ Series Machine Controller Software Manual.

1.2 Motion Programming Method

This section deals with the basic rules for creating motion programs. Read this section thoroughly prior to program execution.

1.2.1 Input Format

■ Motion Program Sample

Motion programs are created in variable-length block format.

Program Sample

Block Number	Program	
00001	MPM001"sample”	...Program number and comment
00002	FMX=T1000000;	...Interpolation feed high-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 command
00007	MOV [X1]100.[Y1]150.[Z1]200.;	...Fast feed
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 operation command
00016	LA01: INC;	...Label
00017	MOV [X1]1000.;	...
00018	JOINTO LA05;	...
00019	LA02: INC;	...
00020	MOV [X1]2000.;	...
00021	JOINTO LA05;	...
00022	LA03: ABS;	...
00023	MV S[Z1]1500 F50000;	...
00024	MW1000=12345;	...
00025	JOINTO LA05;	...
00026	LA04: MW1100=1000;	...
00027	IOW IB101==1;	...
00028	JOINTO LA05;	...
00029	LA05:PJOINT	...Closes parallel operation command
00030	END;	...Closes program

■ Input Format

The variable length input formats are as given in the following table:

Variable Input Format List

Item	Input Format
Program Number	MPM□□□ □□□ = 1~256
Label	8 characters maximum
Motion Commands	3 alphabetical characters (some commands are other than 3 letters)
Coordinate Language	[abcd] ± 123,456 A B C A: Axis Name B: Pos/Neg designation possible C: See Item 1.2.2 “Control Axes” for details on coordinates
Interpolation Feed Speed	F3000000 Changes according to the number of places below the decimal point. (set parameter) 3000.000mm/min when number of places below decimal point =3 30.00000mm/min when number of places below decimal point =5
Wait Time	TIM T1000 10 msec units (no fractions)
Sub-program Number	MPS□□□ □□□ = 1~256
P Designation	P100; Interpolation feed speed ratio setting 1~100
Close Block	;

■ Leading Zero

Numbers following the characters, including program numbers and register (variable) numbers, can omit the leading zero.

Example

```
[X1]00123                                  ⇒ [X1]123
[X1]MW00010                                ⇒ [X1]MW100
MPS002                                     ⇒ MPS2
```

■ +/- Symbol

Although the plus sign may be omitted from numbers, the negative sign may not be omitted.

Example

```
[X1]00123                                  ⇒ [X1]123
[X1]-123                                   ⇒ [X1]-123
```

Supplement

Decimal places cannot be used in the interpolation feed speed (Fxxxx) command. F30000.000 is not possible; enter it as F30000000.

■ Usable Characters

The usable characters and their meanings are given in the following table.

Usable Character List

Character	Meaning
C	C register
D	D register
I	I register
M	M register
O	O register
S	S register
F	Interpolation feed speed
P	Interpolation feed speed override
R	Circular radius
SS	Step signal number
T	Timer value, number of circle turns, FMX, IAC, IDC
U	Circular midpoint coordinate 1 (horizontal)
V	Circular midpoint coordinate 2 (vertical)
MPS	Sub-program number

■ Function Characters

The function characters and their meanings are given in the following table.

Function Character List

Character	Meaning
SP	Space
TAB	Tab
;	End of block
ENTER	Changes row
0~9	numbers
A~Z	Alphabet
.	Decimal point
+	Operation
-	Operation
*	Operation
/	Operation
|	Operation
^	Operation
&	Operation
!	Operation
=	Operation
()	Operation
==	Operation
>	Operation
<	Operation
<>	Operation
>=	Operation
<=	Operation
S {}	Operation
R {}	Operation
—	—

Program Number Handling

The program number is a number for the purpose of program discrimination. There are two kinds of programs: Main programs and sub-programs. The numbers 1~256 may be applied to each.

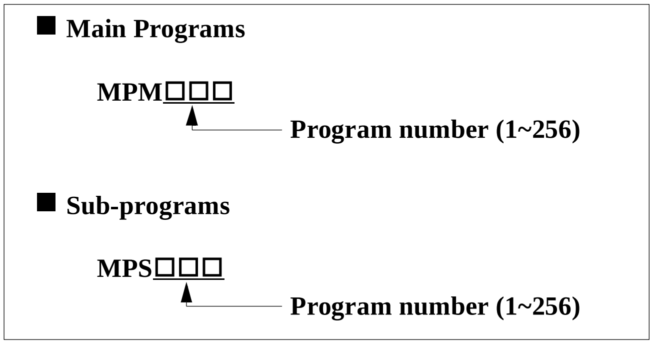


Figure 1.5: Motion Program File Names

Supplemental Item

(1) The same number cannot be designated for both main programs and sub-programs.
(2) Up to 256 programs of the Main program and Subprogram combined can be created.

Comment Writing

It is possible to write comments within the program. These comments are saved within the controller. There are two ways to create a comment, as follows:

1. Surrounding a comment statement with quotation marks.
   “Character string”

Example

```
ZRN [AXIS1]0 [AXIS2]0 [AXIS3]0; “Zero return all axes”
MVS [AXIS1]100.0 [AXIS2]200.0 [AXIS3]300.0; “Three axis linear interpolation”
```

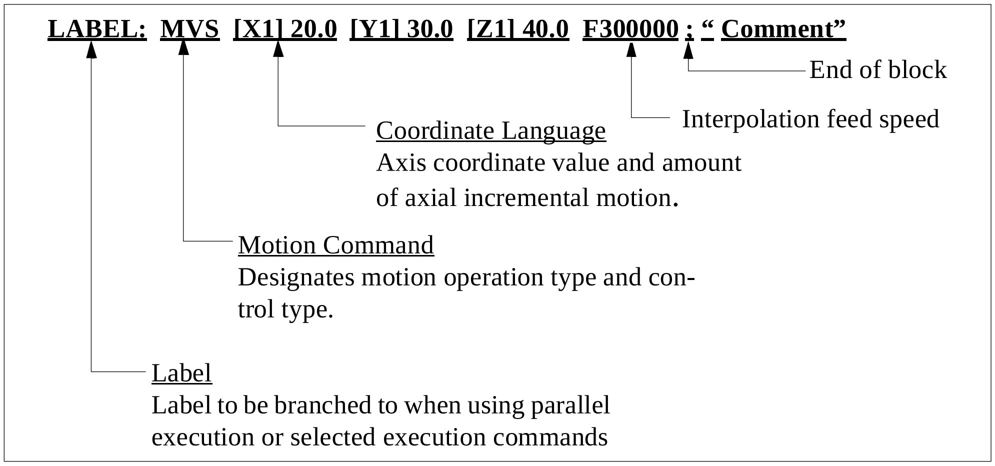
2. All characters following the first quotation mark in a line become comments without surrounding the line in quotation marks if **ENTER** is pressed.
   “Character string **ENTER**

Example

```
“Move to the wait machine position by linear interpolation after all axis zero point return
ZRN [AXIS1]0 [AXIS2]0 [AXIS3]0;
MVS [AXIS1]100.0 [AXIS2]200.0 [AXIS3]300.0;
```

■ Creation of One-Block Commands

a. A one-block command is made according to the input format list. A representative example of a single block is shown below.


LABEL: MVS [X1] 20.0 [Y1] 30.0 [Z1] 40.0 F300000 ; " Comment"
End of block
Interpolation feed speed
Coordinate Language
Axis coordinate value and amount of axial incremental motion.
Motion Command
Designates motion operation type and control type.
Label
Label to be branched to when using parallel execution or selected execution commands

b. Always be sure to insert a space [SP] between the motion command and the coordinate language.

c. Although there is no restriction on the number of characters in a single line of a single block, we recommend that the number of characters be kept to within a range that can be displayed on-screen for the sake of program viewability.

d. A [;] is needed any time that a block is completed.

e. The label is used as the target block for the parallel execution command (PFORK) or selection execution command (SFORK).

■ Label

A label must be used for the parallel execution command (PFORK) or selection execution command (SFORK). Attach a colon [:] to the end of a 1~8 character string of alphanumeric characters or symbols. The characters that can be used in a label are shown below. The first character in a label must be alphabetical.

Characters	Numbers	0~9
	Letters	A~Z, a~z
	Symbols	$, %, ¥, @,—, _, .

Example

```
        PFORK LAB1, LAB2
LAB1:   ZRN [AXIS1]0 [AXIS2]0 [AXIS3]0;
        JOINTO LAB3
LAB2:   MVS [AXIS1]100.0 [AXIS2]200.0 [AXIS3]300.0;
        JOINTO LAB3
LAB3:   PJOINT
```

Important Point

1. The error “Duplicate Label Defined” results if the same label is used multiple times within a program.
2. An error results in the number of PFORK branches if the number of labels differs.

1.2.2 Control Axes

■ Axis Names

It is possible to set a desired axis name of up to eight characters. The axis names are set in the Group Definitions Screen in MotionWorks™. The characters that can be used in a name, as well as the default axis names, are given below.

Usable Characters	0~9, A~Z, a~z
Axis Name Examples	[AXIS1] [X1] [CONV1]
Default Axis Names	If four axes are designated: [A1] [B1] [C1] [D1] If eight axes are designated: [A1] [B1] [C1] [D1] [E1] [F1] [G1] [H1]

Supplement

1. Always be sure to enclose axis names written in 1 ~ 8 alphanumeric characters within [ ].
2. The same axis name cannot be set into multiple axes.
3. An error results if an axis name is designated in the motion program different from the axis name set in the Group Definition Screen.

■ Coordinate Language List

The motion amount and coordinate values attached to the axis name are called "coordinate language" in this book. The meanings of the coordinate language used in this system are shown in the table below.

Coordinate Language Designation Method		Meaning
Axis Name	[X1], [Y1], [AXIS]	Designates axis to be moved
Motion Amount or Coordinate Value	Direct Designation: 123.456 Variable Designation: MW0100	Coordinate value of designated axis, or incremental motion range
Auxiliary Data for Circular Interpolation and Helical Interpolation	R	Circular interpolation radius (set by the increment)
	U V	Circular interpolation center coordinate (Horizontal) Circular interpolation center coordinate (Vertical)
Amount of External Positioning Motion	D	Distance of motion after external signal input (set by the increment)

Important Points

1. The 32-bit integer data type is used when the motion amount or coordinate value is designated by a variable.
   (Ex.) ML0100

2. When there are fractions, insert zeroes for the number of places following the decimal point.
   (Ex.) Use 300000 to designate 300.000 when the number of decimal places = 3.
   However, the servo parameter area (IWCxxx, OWCxxx) in the I, O registers cannot be used for variables of the motion amount or coordinate value.

■ Number of Simultaneously Controlled Axes

The number of simultaneously controlled axes designated from the motion program is shown in the following chart.

Number of Simultaneously Controlled Axes List

Command Language	Number of Simultaneously Controlled Axes
Positioning (MOV)	14 axes maximum
Linear Interpolation (MVS)	14 axes maximum
Circular Interpolation (MCW/MCC)	2 axes
Helical Interpolation (MCW/MCC)	3 axes
Skip Command (SKP)	14 axes maximum
External Positioning (EXM)	1 axis
Time Designated Positioning (MVT)	14 axes maximum

■ Command Units

The programmable command units are in accordance with the settings of b0~b3 “Command Unit Selection” of set-up parameter 17 “Servo Module Function Selection Flag” and set-up parameter 18 “Number of Places Below Decimal Point.”

Command Unit List

Parameter Setting	Command Unit Pulse	Command Unit mm	Command Unit deg	Command Unit inch
# of Places Below Decimal=0	1 pulse	1mm	1º	1”
# of Places Below Decimal=1	1 pulse	0.1mm	0.1º	0.1”
# of Places Below Decimal=2	1 pulse	0.01mm	0.01º	0.01”
# of Places Below Decimal=3	1 pulse	0.001mm	0.001º	0.001”
# of Places Below Decimal=4	1 pulse	0.0001mm	0.0001º	0.0001”
# of Places Below Decimal=5	1 pulse	0.00001mm	0.00001º	0.00001”

Supplement

The number of places below the decimal point are disabled if command unit = pulse. The decimal points in the motion program input and position monitor display are also meaningless.

■ Maximum Command Value

The maximum values of single motion commands are given in the table below.

	# of Decimal Places	Command Unit Pulse	Command Unit mm	Command Unit deg	Command Unit inch
Limited Lengths	0	-2147483648 ~2147483647	-2147483648 ~2147483647	0~ 35999999	-2147483648 ~2147483647
	1	-2147483648 ~2147483647	-2147483648 ~214748364.7	0~ 3599999.9	-2147483648 ~214748364.7
	2	-2147483648 ~2147483647	-2147483648 ~21474836.47	0~ 359999.99	-2147483648 ~21474836.47
	3	-2147483648 ~2147483647	-2147483648 ~2147483.647	0~ 35999.999	-2147483648 ~2147483.647
	4	-2147483648 ~2147483647	-2147483648 ~214748.3647	0~ 3599.9999	-2147483648 ~214748.3647
	5	-2147483648 ~2147483647	-2147483648 ~21474.83647	0~ 359.99999	-2147483648 ~21474.83647

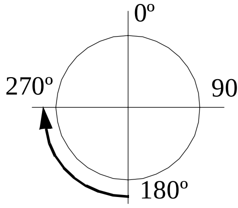
■ Designation of “Absolute Mode” in a Rotary Axis

When an absolute value command (designation mode in a range of 0~359.999º) is used in a rotary axis, the commanded +/- sign shows the rotation direction, and the command value signifies the absolute position.

Example

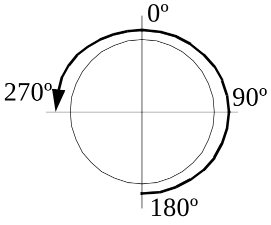
When designating a position 180º from the current position:

Example of Designating the Rotary Axis Absolute Mode



```
ZRN [X1]0;
INC MOV [X1]180.0;
ABS MOV [X1]270.0; moves 90º clockwise
```

Example of Designating the Rotary Axis Absolute Mode



```
ZRN [X1]0;
INC MOV [X1]180.0;
ABS MOV [X1]-270.0; moves 270º counter-clockwise
```

Supplement

When moving to the 0º position by designating the absolute mode in a rotary axis, -0.0 cannot be designated in a counter-clockwise motion. In this case, designate -360.0.

■ Number and Variable Tabulation Method

Numbers used in motion programs are of two types: parameters and variables. The setting method for these numbers is given below.

a. Parameters

Tabulation of Numbers that Can be Designated

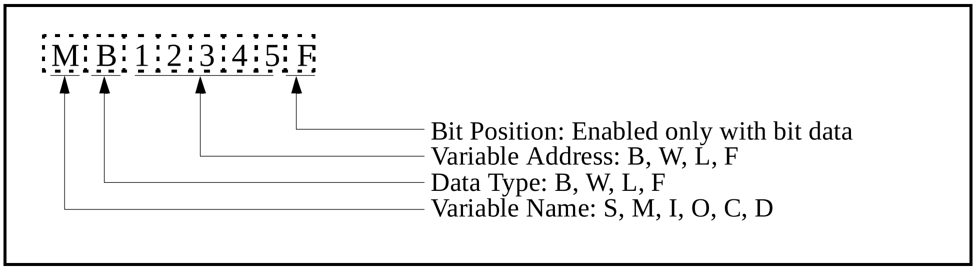
Type	Range	Notation Example
Decimal Integers	-2147483648~2147483647	0, 734, +823, -2493
Decimal Fractions	-2147483.648~2147483.647 Changes according to number of decimal places	763., +824.2, -234.56 -321.12345
Hexadecimal Integers	0~FFFFFFFFH	FFFABCDEH, 2345H, FH
Real Numbers		

b. Variables

The following types of variables exist for use in motion programs. Use them according to the application.

Types of Variables and Tabulation Method

Type	Variable Type	Data Type			
		BIT	WORD	LONG	FLOAT
Global Variables	S Register	SB	SW	SL	SF
	M Register	MB	MW	ML	MF
	I Register	IB	IW	IL	IF
	O Register	OB	OW	OL	OF
	C Register	CB	CW	CL	CF
Local Variables	D Register	DB	DW	DL	DF


M B 1 2 3 4 5 F
Bit Position: Enabled only with bit data
Variable Address: B, W, L, F
Data Type: B, W, L, F
Variable Name: S, M, I, O, C, D

```
(Ex.) MB001001=1;
      MW00100=1234;
      ML00100=12345678;
      MF00100=1234.5678;
```

■ Calculations and Functions

Calculations can combine global variables, local variables and constants with operators and functions. The result can be substituted by a variable. Calculation and functions use the following commands.

Type	Command	Name	Command Format
Value Calculations	=	Substitution	MW– =MW–;
	+	Addition	MW– =MW– +MW–;
	–	Subtraction	MW– =MW– –MW
	*	Multiplication	MW– =MW–*MW–;
	/	Division	MW– =MW–/MW–;
	MOD	Remainder	MW– =MOD;
Logical Calculations	|	OR (Logical OR)	MB– =MB– | MB–;
	^	XOR (Exclusive OR)	MB– =MB– ^MB–
	&	AND (Logical AND)	MB– =MB– &MB–;
	!	NOT (Invert)	MB– =MB– !MB–;
Value Comparisons	= =	Same	IF MW– = =MW–;
	< >	Not same	IF MW– < >MW–;
	>	Greater	IF MW– >MW–;
	<	Less	IF MW– <MW–;
	>=	Greater or equal	IF MW– > =MW–;
	<=	Less or equal	IF MW– < =MW–;
Data Operations	SFR	Right-shift	SFR MB– N– W–;
	SFL	Left-shift	SFL MB– N– W–;
	BLK	Block transfer	BLK MW– MW– W–;
	CLR	Clear	CLR MB– W–;
Basic Functions	SIN	Sine	SIN (MW–);
	COS	Cosine	COS (MW–);
	TAN	Tangent	TAN (MF–);
	ASN	ARC sine	ASN (MF–);
	ACS	ARC cosine	ACS (MF–);
	ATN	ARC tangent	ATN (MW–);
	SQT	Square root	SQT (MW–);
	BIN	BCD→BIN	BIN (MW–);
	BCD	BIN→BCD	BCD (MW–);
	S{}	Designated bit ON	S{MB–}=MB– &MB–;
	R{}	Designated bit OFF	R{MB–}=MB– &MB–;

1.2.3 Feed Speed

■ Fast Feed Speed

a. Fast feed speed is used in the following axis motions:
- Positioning (MOV) commands
- JOG run (JOG) operation
- Step run (STEP) operation

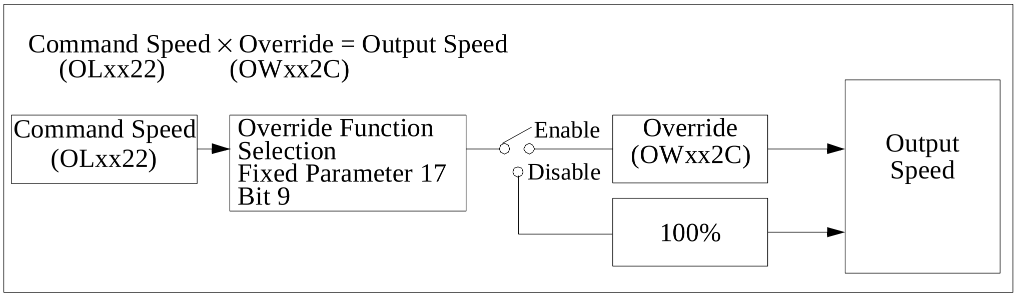
b. Set the fast feed speed in the setup parameters “Fast Feed Speed (OLxx22) or in the motion program “Feed Speed Change Command (VEL)”.
- Fast Feed Speed Parameters (set in setup parameters)

Parameter Number	Name	Setting Range	Unit
OLxx22	Fast Feed Speed	0~2^{31}-1	By command unit

- Methods for setting Fast Feed Speed in the motion program

1. Direct Setting Method for Fast Feed Speed Parameters OLC022=6000 ; Sets fast feed speed for the first axis OLC062=5000 ; Sets fast feed speed for the second axis OLC0A2=7000 ; Sets fast feed speed for the third axis
2. Setting Method Using the Feed Speed Change Command (VEL) VEL [X1] 6000 [Y2] 5000 [Z1] 7000

c. The fast feed can be switched to override within a range of 0~327.67%. This can be set for each axis using the setup parameter “Override (OWxx2C).” There are three override setting methods: in the motion program, in the ladder program, and in the setup parameter.


Command Speed × Override = Output Speed
(OLxx22)
(OWxx2C)
Command Speed (OLxx22)
Override Function Selection Fixed Parameter 17 Bit 9
Enable
Disable
Override (OWxx2C)
100%
Output Speed

Supplement

1. Override is normally enabled during run. Ladder programs and motion programs can be modified by parameter setting during axis motion.

2. When the output speed from the override setting data is outside of operable range, the Parameter Setting Error results.

■ Interpolation Feed Speed

a. The feed speed for the interpolation feed command is set by the number following Character (F). It is sometimes referred to as the F command.

b. The F command for linear interpolation and circular interpolation sets the tangential speed.

Example

If INC MVS [X]200 [Y]500 F500;

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

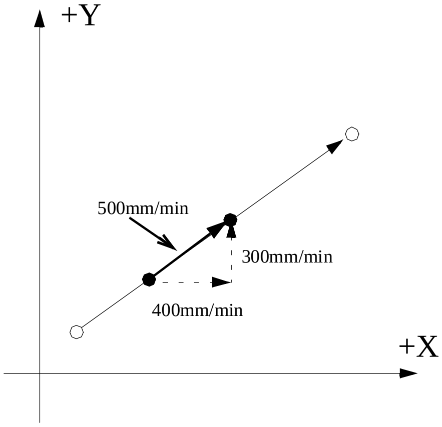


Figure 1.6: Tangential Speed of Two Axis Linear Interpolation

If MCC [X]--- [Y]--- I--- J--- F200;

F=200= $\sqrt{Vx^2 + Vy^2}$ [mm/min]

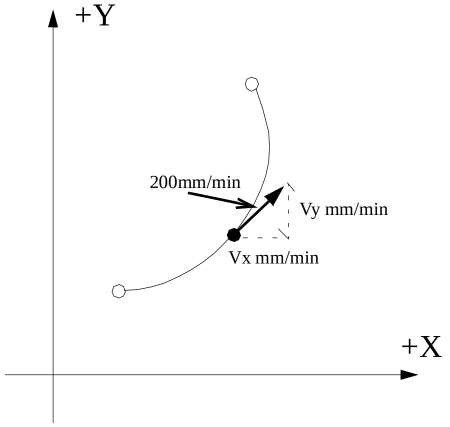


Figure 1.7: Tangential Speed of Circular Interpolation

Example

If INC MVS [X]100 [Y]100 [Z]100 F400;

F=400= $\sqrt{Vx^2 + Vy^2 + Vz^2}$ [mm/min]

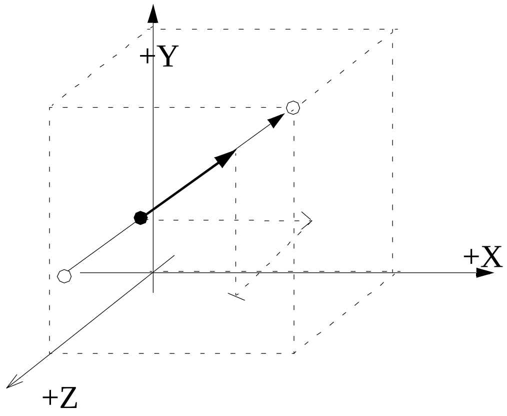


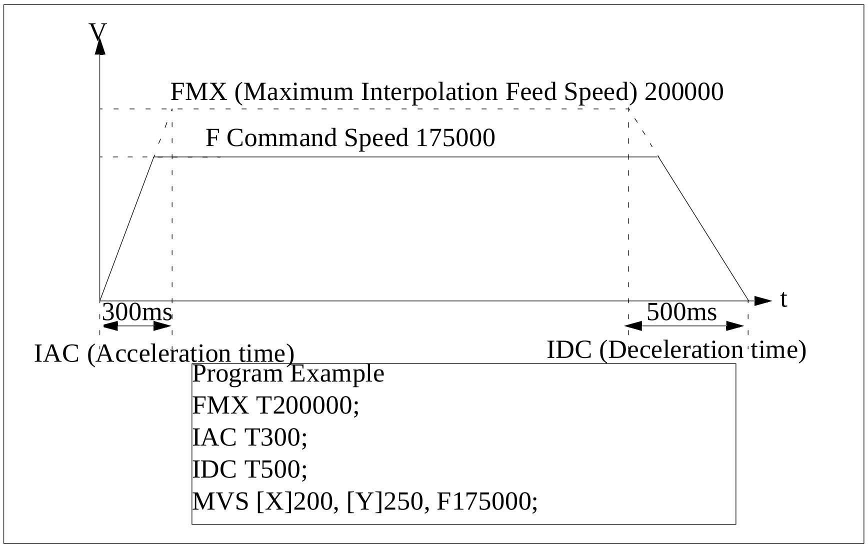
Figure 1.8: Tangential Speed of Three-Axis Linear Interpolation

If INC MVS [X]-- [Y]-- [Z]-- [S]-- F600;

F=600= $\sqrt{Vx^2 + Vy^2 + Vz^2 + Vs^2}$ [mm/min]

(***Tangential Speed of Four-Axis Linear Interpolation)***

c. The feed speed upper limit is restricted by machine and servo performance. Set the upper limit of the feed speed by the following motion commands.



```
Program Example
FMX T200000;
IAC T300;
IDC T500;
MVS [X]200, [Y]250, F175000;
```

Figure 1.9: Interpolation Feed Speed Limit Setting Command

An alarm results if the value of the F command exceeds the maximum interpolation feed speed.

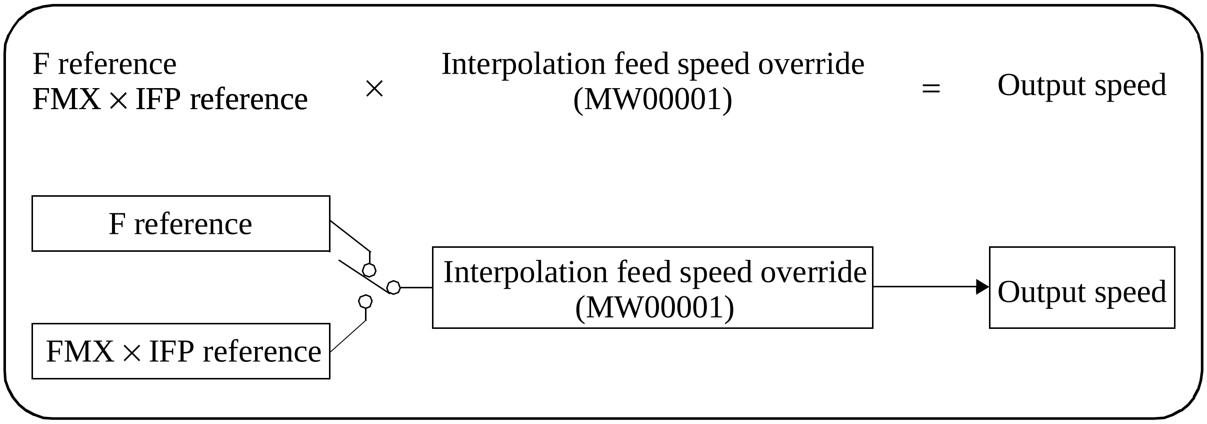
Important Point

If interpolation commands are to be used, an FMX command must be used at the start of a motion program.

d. F Command Units
Decimal places cannot be used in F command values.

F2000000

e. It is possible to switch the interpolation feed speed override within a range of 0~32,767%. Set the override setting in the register (default = MW00001) defined in the Group Definitions Screen. There are three override setting methods: motion program, ladder program, and setup parameter screen.



Motion commands regarding the interpolation feed speed are shown as follows:

• F reference:	[F reference in the interpolation command]
• IFP command:	[Interpolation feed speed ratio setting]
• FMX command:	[Maximum interpolation feed speed]
• IAC command:	[Interpolation acceleration change]
• IDC command:	[Interpolation deceleration change]
• SCC command:	[S-curve setting value change]

Supplement

1. Override is always enabled during running.
2. An FMX clamp results when the output speed from the override setting data exceeds the range.

Important Point

1. The motion speed of the machine does not reach the tangential speed as per the F command when the rotating axis is included in the axes of the interpolation command.
2. A program error results if [F0] is designated in the F command.
3. Do not use the negative F command [F-□□□]. An alarm results.

1.2.4 Motion Command List

A list of the motion commands is given in the table below:

Type	Command	Name	Command Format	Function/Meaning
Axis Motion Commands	MOV	Positioning	MOV [axis1]— [axis2]— ... ; (Up to 14 axes may be designated)	Positioning can be simultaneously executed by fast feed for up to 14 axes.
	MVS	Linear Interpolation	MVS [axis1]— [axis2]— ... F—; (Up to 14 axes may be designated)	Linear motion can be simultaneously executed at interpolation feed speed F for up to 14 axes.
	MCW MCC	Circular Interpolation (clockwise) (counter-clockwise)	MCW [axis1]— [axis2]— R— F—; MCC [axis1]— [axis2]—U—V—T—F—;	Simultaneously executes interpolation for 2 axes at tangential speed F according to a circle of radius R (or a designated mid-point coordinate). It is possible to designate multiple circles in T— during mid-point coordinate designation (T— may also be omitted).
	MCW MCC	Helical Interpolation (clockwise) (counter-clockwise)	MCW [axis1]— [axis2]— U—V—[axis3]—T—F—; MCC [axis1]— [axis2]—R—[axis3]—F—;	Simultaneously moves 3 axes by combining circular interpolation and linear interpolation outside the circular interpolation plane. It is possible to designate multiple circles in T— during mid-point coordinate designation (T— may also be omitted).
	ZRN	Zero Point Return	ZRN [axis1]— [axis2]— ...; (Up to 14 axes may be designated)	Returns each axis to the zero point after [mid-position] positioning. Zero return is executed immediately without mid-positioning after the first time power is turned ON.
	SKP	Skip Command	SKP[axis1]— [axis2]— ...SS—; (Up to 14 axes may be designated)	When the skip signal is turned ON during linear interpolation, the remaining motion is skipped, and the system proceeds to the next block. The machine records the position at which the skip signal went ON.
	MVT	Time Designation Positioning	MVT [axis1]— [axis2]— ...T—; (Up to 14 axes may be designated)	Executes positioning after clamping feed speed so that positioning is completed in a designated time.
	EXM	External Positioning	EXM[axis] - A~B (one axis only)	Moves to designated position (A) if signal is not input. Moves the incremental amount (B) from when the signal is input.

Type	Command	Name	Command Format	Function/Meaning
Basic Control Commands	ABS	Absolute Mode	ABS;	Handles the subsequent coordinate language as absolute values.
	INC	Incremental Mode	INC;	Handles the subsequent coordinate language as incremental values.
	POS	Current Value Change	POS [axis1]— [axis2]— ...;	Simultaneously changes current values for up to 14 axes to a desired coordinate value. Subsequent motion commands move based on this new coordinate system.
	PLN	Coordinate Plane Designation	PLN [axis1] [axis2]	Designates a coordinate plane to be used in commands requiring a coordinate plane.
	MVM	Machine Coordinate Command	MVM MOV [axis1]— [axis2]— ; or MVM MVS [axis1]— [axis2]— ;	Designated when motion on a machine coordinate plane is desired. At zero point return, the automatically set coordinate system is called the machine coordinate. These coordinates are not effected by POS commands.
Speed/Accel/Decel Commands	ACC	Acceleration Time Change	ACC [axis1]— [axis2]—...;	Simultaneously sets the accel/decel time for the linear accel/decel of up to 14 axes.
	SCC	S-Curve Time Constant Change	SCC [axis1]— [axis2]—...;	Simultaneously sets s-curve time constant for the average accel/decel for the motion of up to 14 axes.
	VEL	Feed Speed Change	VEL [axis1]— [axis2]—...;	Sets feed speed for up to 14 axes.
	IAC	Interpolation Acceleration Time Change	IAC T—;	Sets the acceleration time for linear accel/decel during interpolation motion.
	IDC	Interpolation Deceleration Time Change	IDC T—;	Sets the deceleration time for linear accel/decel during interpolation motion.
	IPF	Interpolation Feed Speed Ratio Setting	IFP P—;	Executes speed designation during interpolation feed by designating a maximum speed %.
	FMX	Interpolation Feed Maximum Speed Setting	FMX T—;	Sets the maximum speed during interpolation feed. The interpolation acceleration time is the time to reach this speed from 0.

Type	Command	Name	Command Format	Function/Meaning
Advanced Control Commands	PFN	In-Position Check	MVS [axis1]— [axis2]—... PFN; or PFN [axis1] [axis2];	Proceeds to next block after an interpolation motion command in the same block or previous block enters the positioning completion area (parameter setting).
	INP	2nd In-Position Check	INP [axis1]— [axis2]—... PFN;	Proceeds to next block after the subsequently designated interpolation motion command enters the 2nd positioning completion area.
	SNG	Ignore Single Block	SNG MVS [axis]100. [axis2]200.F1000;	Ignores a block containing this command, and continues running even in the Single Block Operation Mode.
	UFC	User Function Call-out	UFC user function name input data, input address, output address	Calls out functions created by the user.
Sequence Commands	=	Equals	(Result) = (Operation)	Introduces the result of an operation. Operations flow from left to right (regardless of order of priority).
	+	Add	MW— =MW— +MW—; MW— =MW— +123456; MW— =123456 + MW—	Executes addition of integers and real numbers. Calculations are done in real number form if real numbers and integers are mixed.
	-	Subtract	MW— =MW— -MW—; MW— =MW— -123456; MW— =123456 - MW—	Executes subtraction of integers and real numbers. Calculations are done in real number form if real numbers and integers are mixed.
	*	Multiply	MW— =MW— *MW—; MW— =MW— *123456; MW— =123456 * MW—	Executes multiplication of integers and real numbers. Calculations are done in real number form if real numbers and integers are mixed.
	/	Division	MW— =MW— /MW—; MW— =MW— /123456; MW— =123456 / MW—	Executes division of integers and real numbers. Calculations are done in real number form if real numbers and integers are mixed.
	MOD	Remainder	MW— =MW— /MW—; MW— =MOD;	Saves MOD as a remainder to a designated register when commanded in the next block following division.

	|	OR (logical OR)	MB— =MB— |MB—;
			MB— =MB—|1;
			MW— =MW— |MW—;
			MW— =MW— |H00FF;
				Creates a bit/integer logical OR.

Type	Command	Name	Command Format	Function/Meaning

Sequence Commands (continued)	^	XOR (exclusive OR)	MB— =MB—^MB—;
			MB— =MB—^1;
			MW— =MW—^MW—;
			MW— =MW— ^H00FF;
				Creates a bit/integer exclusive OR.

	&	AND (logical AND)	MB— =MB—&MB—;
			MB— =MB—&1;
			MW— =MW—&MW—;
			MW— =MW— &H00FF;
				Creates a bit/integer logical AND.

	!	NOT (invert)	MB— =MB—!MB—;
			MB— =MB—!1;
			MW— =MW—!MW—;
			MW— =MW— !H00FF;
				Creates a bit with a reverse value.

	()	Parentheses	MW— =MW—& (MW—|MW—);	Executes operation of the logical operation within parentheses first.
	S{}	Designated Bit ON	S{MB—} = MB— &MB—;	The designated bit goes ON if the result of the logical operation is true. The designated bit does not go OFF even if the result of the logical operation is false.
	R{}	Designated Bit OFF	R{MB—} = MB— &MB—;	The designated bit goes OFF if the result of the logical operation is true. The designated bit does not go ON even if the result of the logical operation is false.

	SIN	Sine	SIN(MW—);
			SIN(90);
				Calls the sine of an integer/real number (deg), and returns a real value.

	COS	Cosine	COS(MW—);
			COS(90);
				Calls the cosine of an integer/real number (deg), and returns a real value.

	TAN	Tangent	TAN(MW—);
			TAN(45);
				Calls the tangent of an integer/ real number (deg), and returns a real value.

	ASN	Arc Sine	ASN(MW—);
			ASN(90);
				Calls the arc sine of an integer/ real number (deg), and returns a real value.

	ACS	Arc Cosine	ACS(MW—);
			ACS(90);
				Calls the arc cosine of an integer/ real number (deg), and returns a real value.

Type	Command	Name	Command Format	Function/Meaning

Sequence Commands (continued)	ATN	Arc Tangent	ATN(MW—);
			ATN(45);
				Calls the arc tangent of an integer/real number (deg), and returns a real value.

	SQRT	Square Root	SQT(MW—);
			SQT(100);
				Calls the square root of an integer/real number (deg), and returns a real value.

	BIN	BCD→BIN	BIN (MW—);	Converts BCD data to BIN data.
	BCD	BIN→BCD	BCD (MW—);	Converts BIN data to BCD data.
	==	Same	IF MW— ==MW—; WHILE MW— ==MW—;	Used with IF or WHILE conditions. Is true if the left and right are the same.
	<>	Not Same	IF MW— <>MW—; WHILE MW— <>MW—;	Used with IF or WHILE conditions. Is true if the left and right are not the same.
	>	Greater	IF MW— >MW—; WHILE MW— >MW—;	Used with IF or WHILE conditions. Is true if the left is greater than the right.
	<	Less	IF MW— <MW—; WHILE MW— <MW—;	Used with IF or WHILE conditions. Is true if the left is less than the right.
	>=	Equal or Greater	IF MW— >=MW—; WHILE MW— >=MW—;	Used with IF or WHILE conditions. Is true if the left is equal to or greater than the right.
	<=	Equal or Less	IF MW— <=MW—; WHILE MW— <=MW—;	Used with IF or WHILE conditions. Is true if the left is equal to or less than the right.
	TON	Time Limit ON Timer	MB— &TON(5.00 MW—);	The basic timer clock is 10msec. Timing occurs while the bit variable is ON (waits while bit is OFF). The designated bit on the left goes ON upon reaching the set time (parameter * clock). Time values (clock numbers) are stored in the word variable).
	TOF	Time Limit OFF Timer	MB— &TOF(5.00 MW—);	The basic timer clock is 10msec. Timing occurs while the bit variable is ON (waits while bit is OFF). The designated bit on the left goes OFF upon reaching the set time (parameter * clock). Time values (clock numbers) are stored in the word variable).

Type	Command	Name	Command Format	Function/Meaning
Sequence Commands (continued)	SFR	Shift Right	SFR MB— N— W—;	Shifts the word variable a designated amount to the right.
	SFL	Shift Left	SFL MB— N— W—;	Shifts the word variable a designated amount to the left.
	PON	Start-up Detection	PON (MB— MB—);	A designated bit goes ON upon start-up detection.
	NON	Shutdown Detection	NON (MB— MB—);	A designated bit goes ON upon shutdown detection.
	BLK	Block Transmission	BLK MW— MW— MW—;	Transmits a range of blocks (parameter designation) as the start of a designated bit (word) variable.
	CLR	Clear	CLR MB— W—;	Sets a group of variables to the designated parameter OFF (0) as the start of a designated bit (word) variable.
Control Commands	MSEE	Sub-program Call-out	MSEE MPS— 0;	Executes an MPS sub-program.
	TIM	Timed Wait	TIM T—;	Waits for the time designated in T only, then proceeds to the next block.
	IOW	I/O Variable Wait	IOW MB— == ***;	Stops motion program execution until the I/O variables fulfill conditions.
	END	Program END	END;	Ends the motion program.
	RET	Sub-program END	RET;	Ends a sub-program.

	IF
	ELSE
	IEND
		Branch Commands	IF (conditional);
			(process 1)
			ELSE;
			(process 2)
			IEND;
				Executes (process 1) if the conditions are satisfied, and executes (process 2) if the conditions are not satisfied.

	WHILE
	WEND
		Repeat Commands	WHILE (conditional);
			...
			WEND;
				Repeatedly executes WHILE~WEND processing while certain conditions continue to be satisfied.

	PFORK JOINTO PJOINT	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;	Parallel executes a block designated by a label. No more than two labels can be designated in the case of sub-programs. Furthermore, motion program commands cannot be used in a block selected by two labels. END and RET cannot be used during parallel processing execution.

Type	Command	Name	Command Format	Function/Meaning
Control Commands	SFORK JOINTO SJOINT	Selected Execution Commands	SFORK Conditional1 ? Label 1, Conditional 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 conditional1 is satisfied, and executes (process 2) if conditional2 is satisfied.

2 Motion Commands

Programing of axial motion commands and control commands are explained in this chapter.

2.1 Axial Motion Commands

This section describes how to command axial motion, and gives program examples.

2.1.1 Positioning (MOV)

> **⚠ CAUTION**
>
> The move path based on the positioning (MOV) command is not like the straight line in the linear interpolation. When programing, the move path must be checked to avoid tools interfering with the workpiece.
>
> Forgetting this check carries a risk of tool damage, as well as bodily injury due to interference.

■ Outline

The positioning command (MOV) makes each axis move independently from the current position to the end position by fast feed speed (the speed set up in each axis's parameter). Up to 14 axes can be moved simultaneously. An axis that is not designated does not move. The move path based on the MOV command does not move along the line designated by the linear interpolation command mentioned in Item 2.1.2.

■ Detailed Explanation

The designated method of the MOV command is shown as follows:

```
MOV    [axis1]—[axis2]—•••;
       ____________________
       Designated position
```

The move path based on the MOV command is illustrated in the following figure:

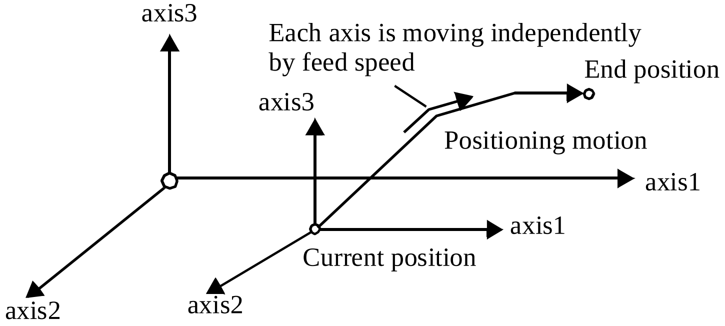


Figure 2.1: Move Path Based on the MOV Command

- The designated position is set up as either absolute or incremental in previously commanded ABS/INC* mode:

Absolute (ABS) mode:	Target position
Incremental (INC) mode:	Incremental amount from the current position

* **ABS/INC command**: Command uses either absolute value or incremental value to handle the coordinate term, which is called “modal group command.” Once designated, it is enabled until the command is switched.

- For the axial motion based on the MOV command, execute an in-position check that verifies whether the axial motion based on the MOV command has entered the positioning completion range. After the in-position check, execute the next moving command block. The following figure illustrates the motion of the in-position check.

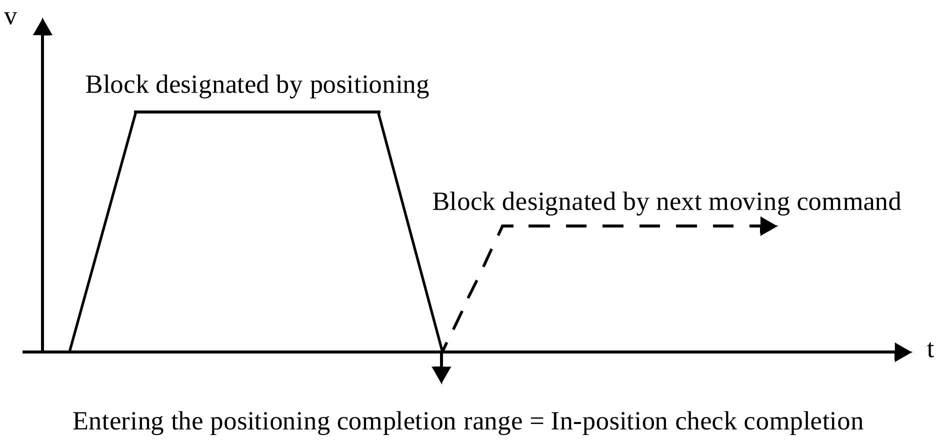
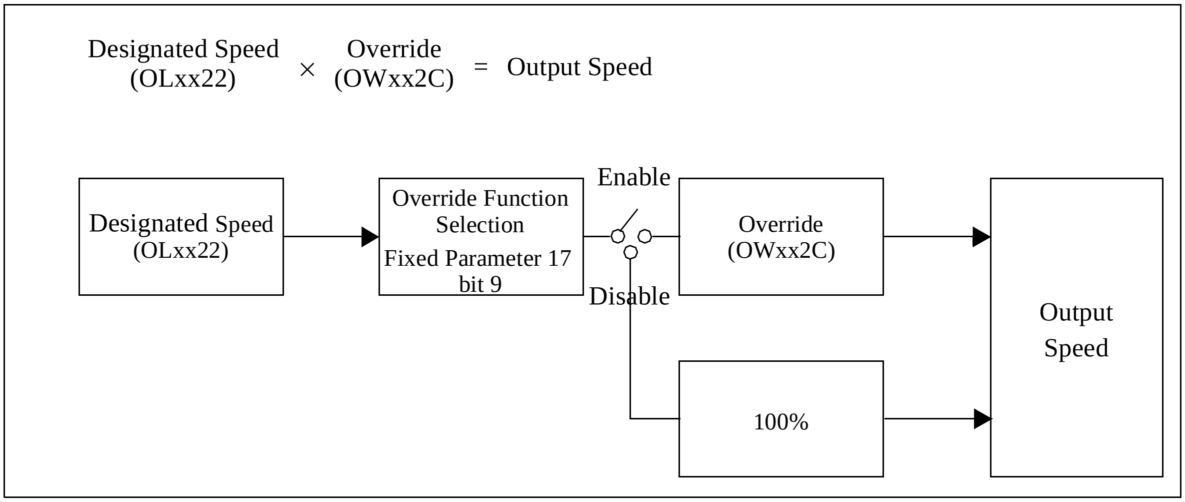

v
Block designated by positioning
Block designated by next moving command
t
Entering the positioning completion range = In-position check completion

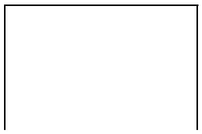
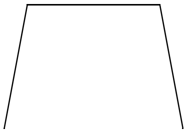
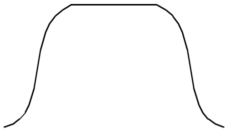
Figure 2.2: In-position Check Motion

- Fast feed speed is set up in parameter 30 “Fast feed speed (OLxx22)” of each axis. Override can be set in a range of 0~327.67% to the fast feed speed. Use parameter 35 “Override (OWxx2C)” to set up the override in each axis.


Designated Speed (OLxx22) × Override (OWxx2C) = Output Speed
Designated Speed (OLxx22)
Override Function Selection
Fixed Parameter 17 bit 9
Enable
Disable
Override (OWxx2C)
100%
Output Speed

- For automatic accel/decel control based on the MOV command, linear accel/decel, or S-curve accel/decel can be selected in the parameter settings.
- Automatic accel/decel (in positioning) related parameters and motion commands are shown as follows:
  - Setup parameter 11
    "Linear acceleration time setting (OWxx0C)"
  - Setup parameter 18
    "S-curve accel time (OWxx14)"
  - Setup parameter 29
    "Servo command flag (OWxx21: b4~b7): Filter type selection"
  - Motion command
    "Acceleration time change (ACC)"
  - Motion command
    "S-curve time constant change (SCC)"

Various accel/decel patterns can be set using a combination of the above parameters and motion commands.

No.	Automatic Accel/Decel Type	Related Parameter Setting	Note
1	No accel/decel	OWxx0C=0 OWxx21b4~b7=0	
2	Linear	OWxx0C≠0 OWxx21b4~b7=0	
3	S-curve	OWx0C≠0 OWxx21b4~b7=2 OWxx14≠0	

Supplement

The setup parameter 18 “S-curve accel time (OWxx14)” is automatically transferred to the servo amplifier by the “S-curve time constant change (SCC)” motion command.

■ Program Example

The program example of the MOV command in the ABS mode is shown as follows:

```
ABS;
MOV [axis1]4000 [axis2]3000 [axis3]2000;
Start at current position: axis1 = axis2 = axis3 = 0
```

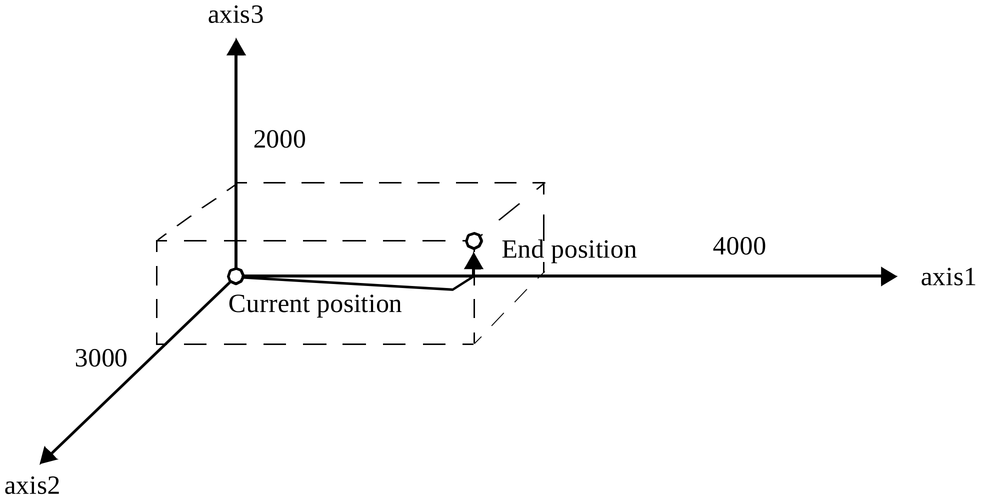


Figure 2.3: Program Example of the MOV Command

2.1.2 Linear Interpolation (MVS)

⚠ CAUTION

The axis that executes the linear interpolation (MVS) command can be either a linear or rotating axis. However, if the rotating axis is included, the move path of the linear interpolation is not a straight line. When programing, the move path must be checked to avoid tools interfering with the workpiece.

Forgetting this check carries a risk of tool damage, as well as bodily injury due to interference.

■ Outline

The linear interpolation (MVS) command is a command that makes each axis move along a straight line, from the current position to the end position, by interpolation feed speed. Up to 14 axes can be moved simultaneously. An axis not designated does not move.

■ Detailed Explanation

The designating method of the MVS command is shown as follows:

MVS $\underbrace{\text{[axis1]—[axis2]—•••;}}_{\text{Designated position}}$ $\underbrace{\text{F—;}}_{\text{Interpolation feed speed}}$

The move path is illustrated in the following figure:

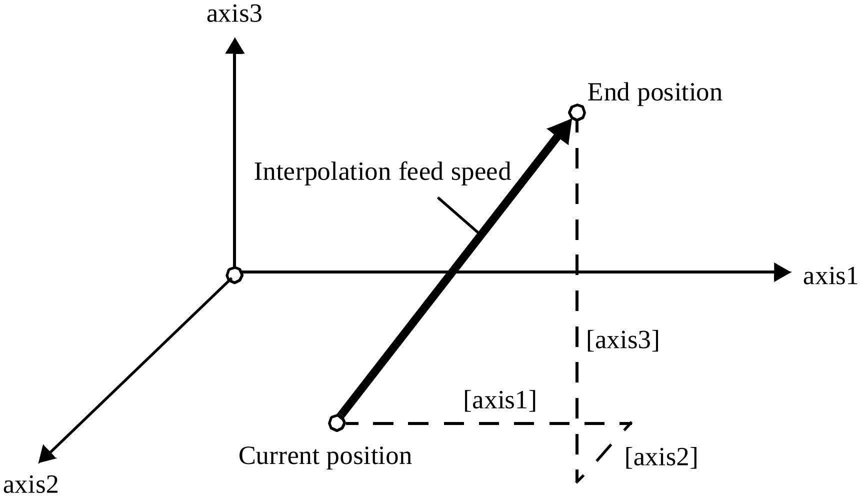


Figure 2.4: Move Path Based on the MVS Command

- The designated position is set up in the ABS/INC mode which was previously set.
- The interpolation feed speed is also called F command. It is designated by the speed designation (F) or speed command (%) (IFP). The final F command designated in the previous block is enabled. When the power supply is connected, an alarm occurs if the interpolation command has not been designated by the F command.
- When designating an F command that exceeds the limited value set up in the maximum interpolation feed speed, an alarm occurs.

- When 2 axes are designated (axis1 and axis2):

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

- When 3 axes are designated (axis1, axis2 and axis 3):

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

- When 4 axes are designated (axis1, axis2, axis3 and axis4:

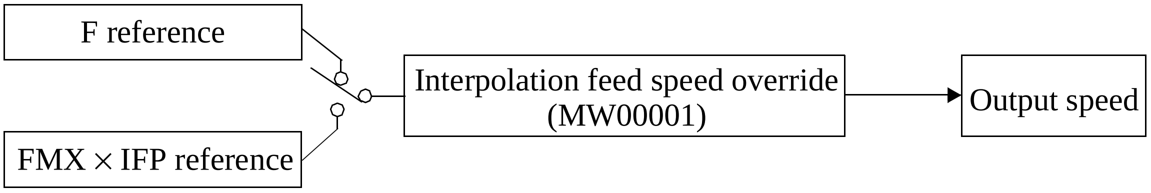
$$F = \sqrt{V_{axis1}^2 + V_{axis2}^2 + V_{axis3}^2 + V_{axis4}^2}$$

Important Point

When creating the motion program that uses the interpolation command, designate the maximum interpolation feed speed (FMX) in the beginning of the program, to avoid an alarm.

- In actual program running, an override in a range of 0~327.67% can be set up to the F command value. The override is enabled immediately. It is set up in the register (default = MW0001) fixed in the group definition window.

F reference
FMX × IFP reference × Interpolation feed speed override (MW00001) = Output speed



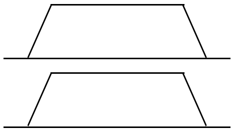
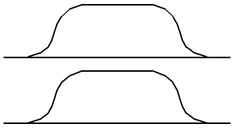
- The in-position check is not executed for the axial motion based on the linear interpolation (MVS) command. Execute the next block when the pulse distribution of the designated block is finished. When attempting to execute the next block after the in-position check, designate the (PFN) in either the same block or the next block.

- The following control methods can be selected from the parameter and the IAC, IDC command settings in the automatic accel/decel control during movement, based on the interpolation command:
    - Linear accel/decel
    - Separate accel/decel
    - S-curve accel/decel

- Parameters and motion commands regarding automatic accel/decel of the interpolation feed are shown as follows:

• Setup parameter 18:	[S-curve accel time (OWxx14)]
• Setup parameter 29:	[Servo command flag (OWxx21 b4~b7) Filter type selection]
• Motion command:	[Maximum interpolation feed speed (FMX)]
• F reference in interpolation feed:	[Interpolation feed speed]
• Motion command:	[Interpolation feed speed ratio (IFP)]
• Motion command:	[Interpolation acceleration time change (IAC)]
• Motion command:	[Interpolation deceleration time change (IDC)]
• Motion command:	[S-curve time constant change (SCC)]

Various accel/decel patterns can be set by combining the above parameters and motion commands.

No.	Auto. Accel/Decel Type	Relate Parameter and Command	Note
1	No accel/decel	Interpolation accel time change (IAC)=0 Interpolation decel time change (IDC)=0 OWxx21 b4~b7=0	
2	Linear Interpolation	Interpolation accel time change (IAC)≠0 Interpolation decel time change (IDC)≠0 OWxx21 b4~b7=0	
3	S-curve	Interpolation accel time change (IAC)≠0 Interpolation decel time change (IDC)≠0 OWxx21 b4~b7=2 OWxx14≠0	

- Use the IAC or IDC command to set up the interpolation command-based accel/decel time of the automatic accel/decel control.

■ Program Example

The program example of the MVS command in the ABS mode is shown as follows:

```
FMX T30000000;
ABS;
MVS [axis1]4000 [axis2]3000 [axis3]2000 F1000;
Start at current position: axis1 = axis2 = axis3 = 0
```

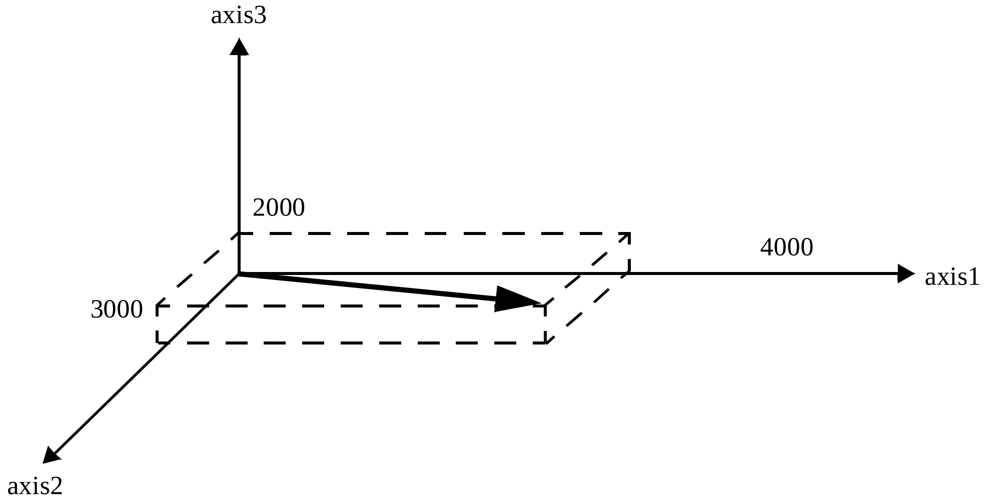


Figure 2.5: Program Example of the MVS Command

Supplement

(1) The speed designation (F) can only be set in the same block as the interpolation command.

(2) The speed command (%) (IFP) is set separately; it cannot be set in the same block as the interpolation command.

(3) When the speed override to the F command value exceeds the maximum interpolation feed speed (FMX), it is limited by the FMX speed.

2.1.3 Circular Interpolation (MCW, MCC)

■ Outline

The circular interpolation (MCW, MCC) command simultaneously moves 2 axes on the designated plane, from the current position to the end position, along a circular arc determined by the central position (U-V) or the radius value (R), by the interpolation feed speed.

■ Detailed Explanation

An example of the designating method is shown as follows:

$$\text{MCW}\quad \underset{\text{A}}{\underline{\text{[axis1]—[axis2]—}}}\quad \underset{\text{B}}{\underline{\text{U—V—}}}\quad \underset{\text{C}}{\underline{\text{T—}}}\quad \underset{\text{D}}{\underline{\text{F—;}}}$$

A: End position
B: Central position
C: Turn number
D: Interpolation feed speed
Or,

$$\text{MCC}\quad \underset{\text{A}}{\underline{\text{[axis1]—[axis2]—}}}\quad \underset{\text{B}}{\underline{\text{R—}}}\quad \underset{\text{C}}{\underline{\text{F—;}}}$$

A: End
B: Radius
C: Interpolation feed speed

Note: When the central position is designated, multiple circular arcs can be designated. (Omission is also possible.)

The rotational direction of the circular interpolation command is shown as follows:

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

Important Points

- Before executing the circular interpolation command, designate the plane of the circular interpolation by the coordinate plane designation (PLN) command. For the circular interpolation command, the rotational direction (MCW or MCC) of the circular arc must be designated. Designate the end position and circular arc center of the horizontal axis and the vertical axis on the designated plane using axis1 and axis2.
- Designate the end position and circular arc center in the order corresponding with the horizontal axis and vertical axis names which are designated by the PLN command.

The designating method of the circular interpolation is shown as follows:

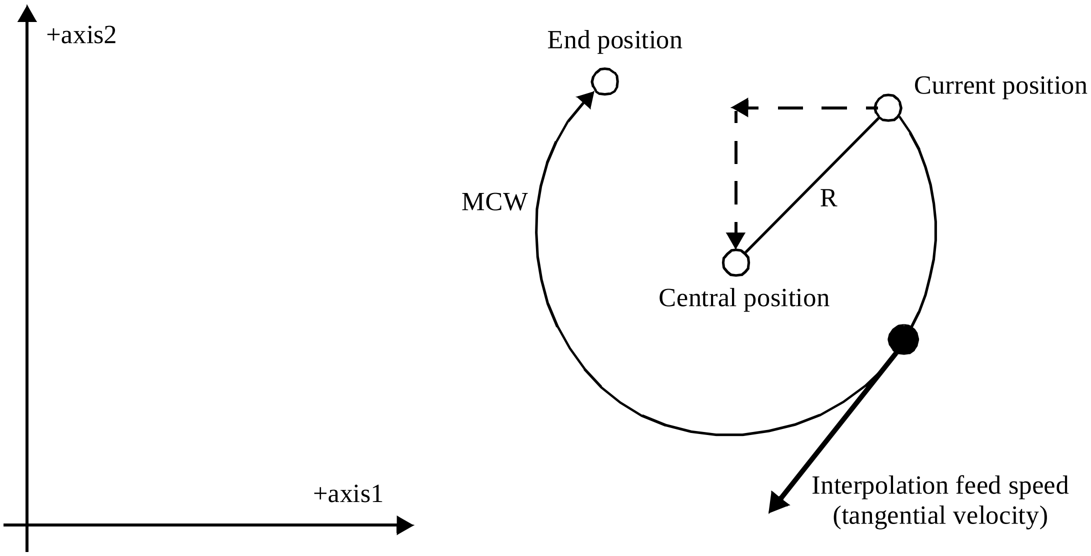


Figure 2.6: Designating Method of the Circular Interpolation (MCW, MCC) Command

- The designation of the end and central positions is executed in either the ABS or INC mode, whichever was last designated.

■ Program Example

a. The following is a program example in the ABS mode.

```
ABS;
PLN [X] [Y];
MCC [X]1500 [Y]4000 U2500 V1000 F150;
```

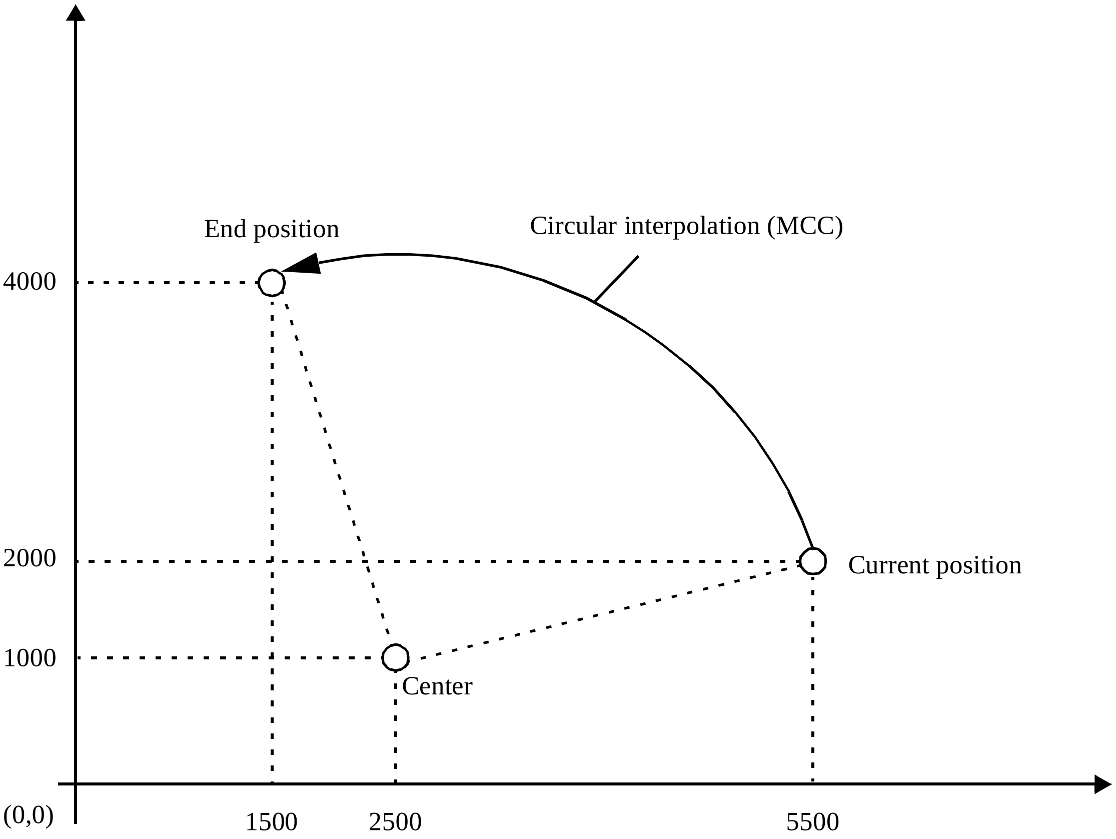


Figure 2.7: Program Example in the ABS Mode

b. The following is a program example in the INC mode.

```
INC;
PLN [X] [Y];
MCC [X]-4000 [Y]2000 U-3000 V-1000 F150;
```

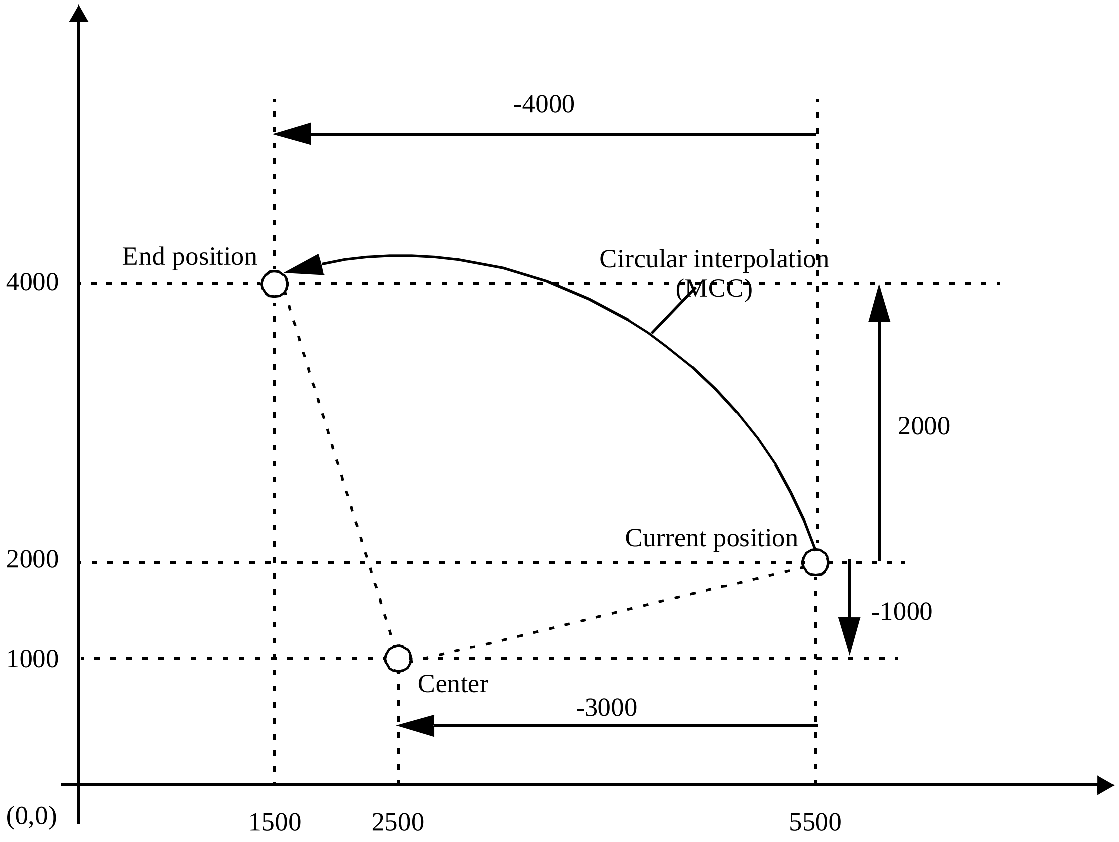


Figure 2.8: Program Example in the INC Mode

c. Below is a program example of multiple circular arcs.

```
ABS;
PLN [X] [Y]
MCC [X] 4000 [Y]2000 U2000 V2000 T2 F150;
```

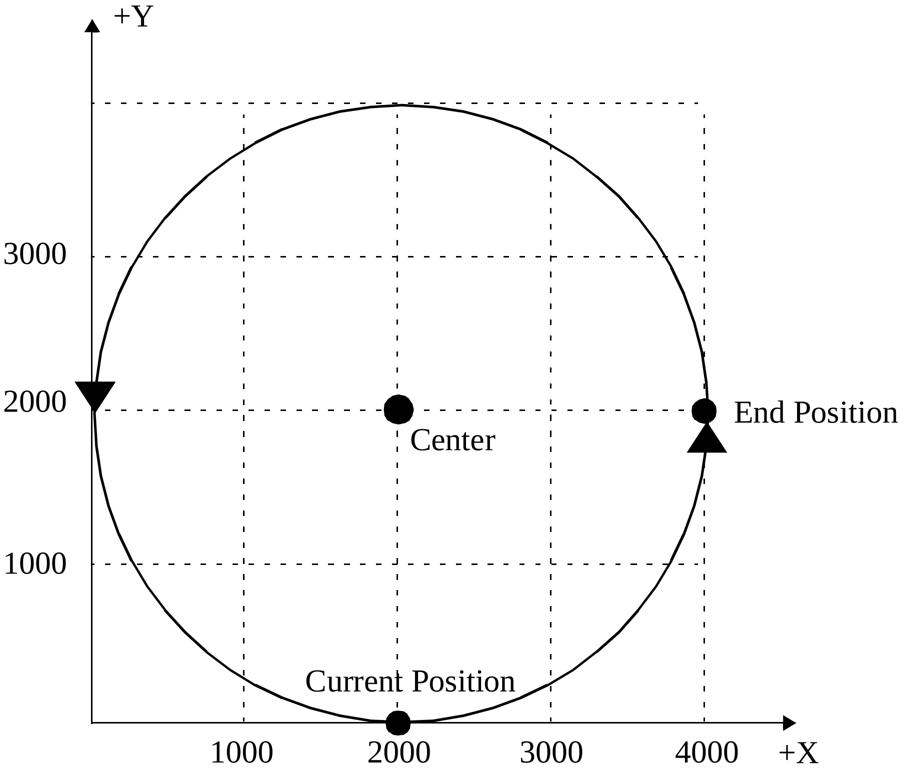


Figure 2.9: Program Example of Multiple Circular Arcs

In the above case, there are 2¼ multiple circular arcs.

Supplement

When the Current Position = End Position: MCC [X]2000 [Y]0 U2000 V2000 T2 F150, there are 3 multiple circular arcs.

- The circular arc can be designated by the radius value (R) instead of the center position. The circular interpolation at this point is shown in Figure 9.

  In the command of MCW [axis1]—[axis2]—R—;

  When R>0, circular interpolation is smaller than circular angle 180°
  When R<0, circular interpolation is greater than circular angle 180°

  Note: When R=0, an alarm occurs.

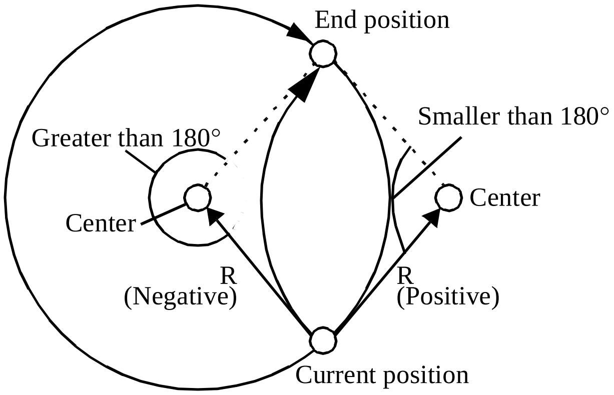


Figure 2.10: Circular Interpolation

- Before designating the circular interpolation, a plane must be designated to avoid an alarm. If there is no new designation, the last plane designated is used.
- The interpolation feed speed is also called F command. It is designated by the speed designation (F) or speed command (%) (IFP). The F command designated in the previous block is used. An alarm occurs if the interpolation command is not designated by the F command.
- The tangential velocity speed of the designated circular arc is equal to the F command value. However, an alarm occurs when designating an F command that exceeds the limited value set up in the maximum interpolation feed speed.

The maximum interpolation feed speed is illustrated in the following figure:

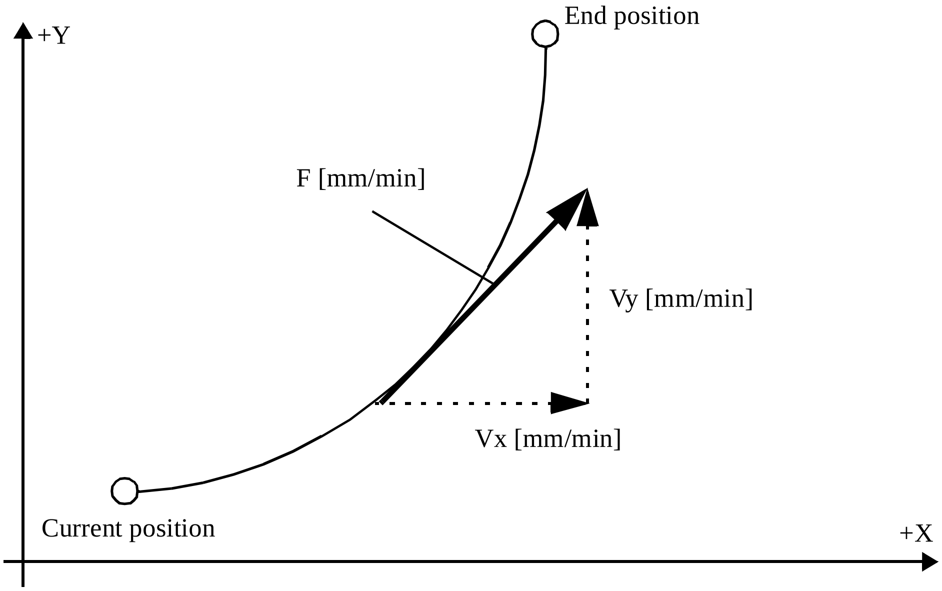


Figure 2.11: Maximum Interpolation Feed Speed

- When the program is running, an override in the range of 0~327.67% can be set up to the F command value. The override is enabled immediately.

Important Points

When creating a motion program that uses the interpolation command, designate the maximum interpolation feed speed (FMX) in the beginning of the program, to avoid an alarm.

- Regardless of the moving command, when attempting to execute the next block after in-position check, designate it in the same block or the next block.
- In the automatic accel/decel control during movement based on the interpolation command, the following control methods can be selected from the parameter and the IAC, IDC command settings:
  - Linear accel/decel
  - Separate accel/decel
  - S-curve accel/decel
- When designating central position, a completely closed circular arc can be designated within 1 block, by making the current and end position at one point. By turn number (T) designation, multiple circular arcs also can be designated within 1 block. However, 1 circle and multiple circular arcs cannot be designated if the radius (R) is designated.

Supplement

The designating methods of the speed reference, accel/decel, override, etc. are the same as those of linear interpolation. Refer to item 2.1.2, Linear Interpolation (MVS) for detailed information.

■ Program Example

The program example of central position designation in the ABS mode is shown as follows:

```
ABS;
MOV [X] 0 [Y] 0;
PLN [X] [Y];
MCW [X]0 [Y]0 U1000 V0;
```

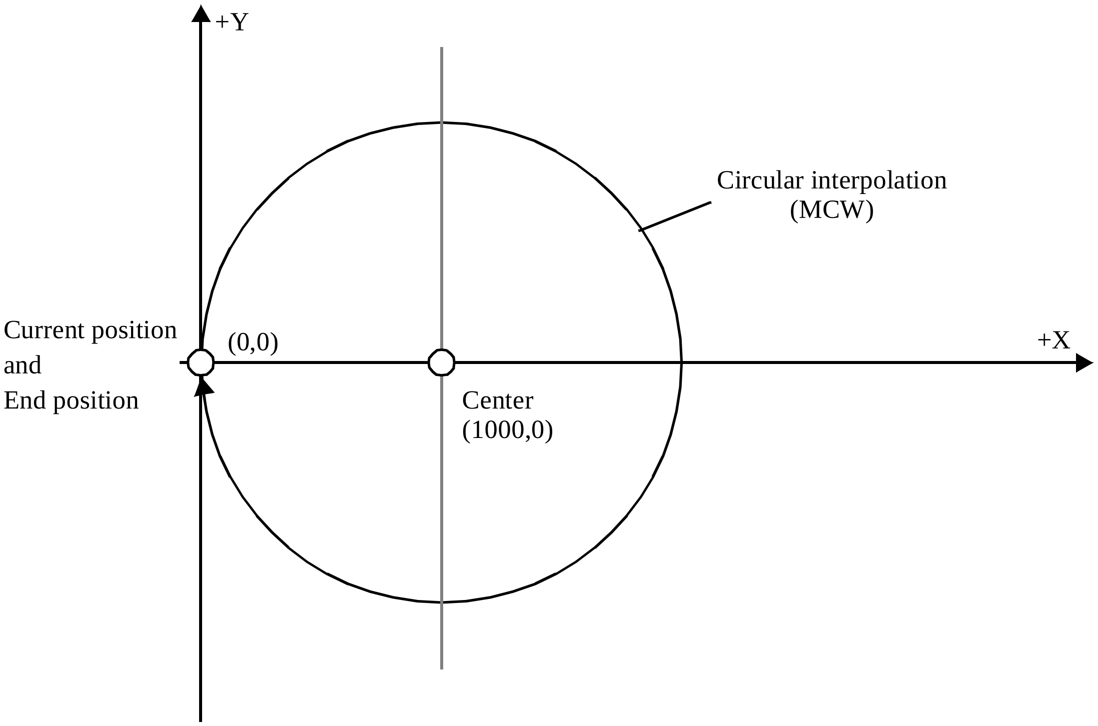


Figure 2.12: Program Example of Central Position Designation

Supplement

(1) The speed designation (F) can only be set in the same block as the interpolation command.
(2) The speed command (%) (IFP) is set separately; it cannot be set in the same block as the interpolation command.
(3) When the speed override to the F command value exceeds the maximum interpolation feed speed (FMX), it is limited by the FMX speed.

2.1.4 Helical Interpolation (MCW, MCC)

⚠ CAUTION

In the helical interpolation (MCW, MCC) command, the axis that executes the linear interpolation can be either the linear axis or the rotating axis. However, the axis chosen in linear interpolation does not shape the move path helically. When programming, the move path must be checked to avoid tools interfering with the workpiece.

Forgetting this check carries a risk of tool damage, as well as bodily injury due to interference.

■ Outline

Helical interpolation (MCW, MCC) is a command that extends circular interpolation. It executes the linear interpolation movement of each axis simultaneously with circular interpolation along the circular arc determined by the set central position or radius (R) value.
During movement, the tangential velocity of the circular interpolation is the interpolation feed speed (F command). This movement is called helical interpolation. Designation regulation for circular interpolation is based on the regulation of the circular interpolation (the description regarding it is omitted here).

■ Detailed Explanation

The designating method of the helical interpolation is shown as follows:

$$\text{MCW}\quad \underset{\text{A}}{\underline{\text{[axis1]—[axis2]—}}}\quad \underset{\text{B}}{\underline{\text{U—V—}}}\quad \underset{\text{C}}{\underline{\text{[axis3]—}}}\quad \underset{\text{D}}{\underline{\text{F—;}}}$$

A: End position
B: Central position
C: End position of the linear interpolation
D: Interpolation feed speed
Or,

$$\text{MCC}\quad \underset{\text{A}}{\underline{\text{[axis1]—[axis2]—}}}\quad \underset{\text{B}}{\underline{\text{R—}}}\quad \underset{\text{C}}{\underline{\text{[axis3]—}}}\quad \underset{\text{D}}{\underline{\text{F—;}}}$$

A: End
B: Central position
C: End position of the linear interpolation
D: Interpolation feed speed

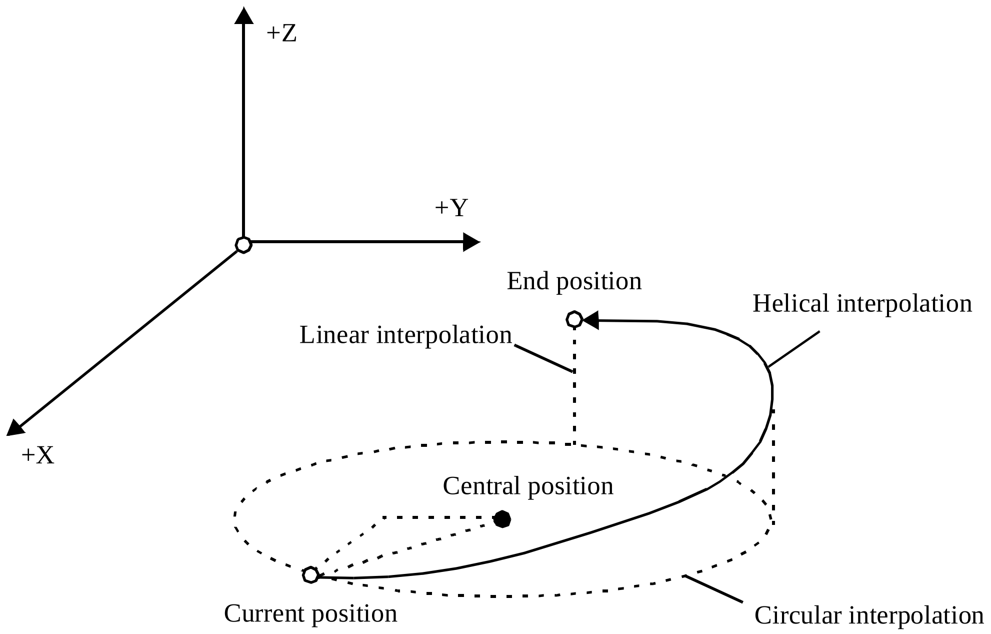


Figure 2.13: Helical Interpolation Command

- In the axis character of linear interpolation, the axis that is not designated by plane designation can be designated. It is not necessary to designate a right angle in the interpolation plane.
- Before designating the helical interpolation, designate the plane by the coordinate plane designation (PLN) command.
- Designate the end position and circular arc center in the order that corresponds to the horizontal axis and vertical axis names which are designated by the PLN command.
- The interpolation feed speed is also called F command. It is designated by the speed designation (F) or speed command (%) (IFP).
- The F command which is designated in the previous block is used. An alarm occurs if the interpolation command is not designated by the F command.
- The feed speed F indicates the tangential velocity of the circular arc in the circular plane. Therefore, the speed (F’) of the linear axis is:

  F’ = F× (Length of the linear axis) / (Length of the circular arc).

Supplement

The designating methods of the speed reference, accel/decel, override, etc. are the same as those of linear interpolation. Refer to item 2.1.2, Linear Interpolation (MVS) for detailed information.

■ Program Example

The program example of the helical interpolation command in the ABS mode is shown as follows:

```
ABS;
MOV [X]1000 [Y]0 [Z]0;
PLN [X] [Y];
MCC [X]0 [Y]1000, U0 V0 Z500;
```

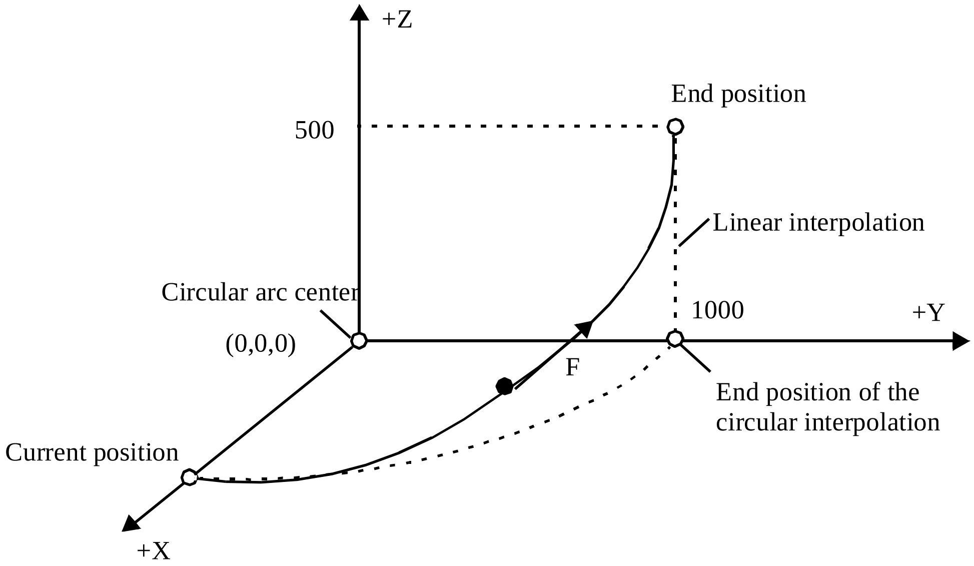


Figure 2.14: Program Example of the Helical Interpolation Command

Supplement

(1) The speed designation (F) can only be set in the same block as the interpolation command.

(2) The speed command (%) (IFP) is set separately; it cannot be set in the same block as the interpolation command.

(3) When the speed override to the F command value exceeds the maximum interpolation feed speed (FMX), it is limited by the FMX speed.

2.1.5 Zero-point Return (ZRN)

■ Outline

The zero-point return (ZRN) command is a command that designates the zero-point return motion. Up to 14 axes can be designated. The point at which the motion stops is set up as the zero-point of the machine coordinate.
The move function advances to the next block after all designated axes have finished the zero-point return motion.

■ Detailed Explanation

The designating method of the ZRN command and the move path are shown as follows:

```
ZRN     [axis1]—[axis2]—•••;
        ____________________
        Axis designation + 0 (the position is always 0)
```

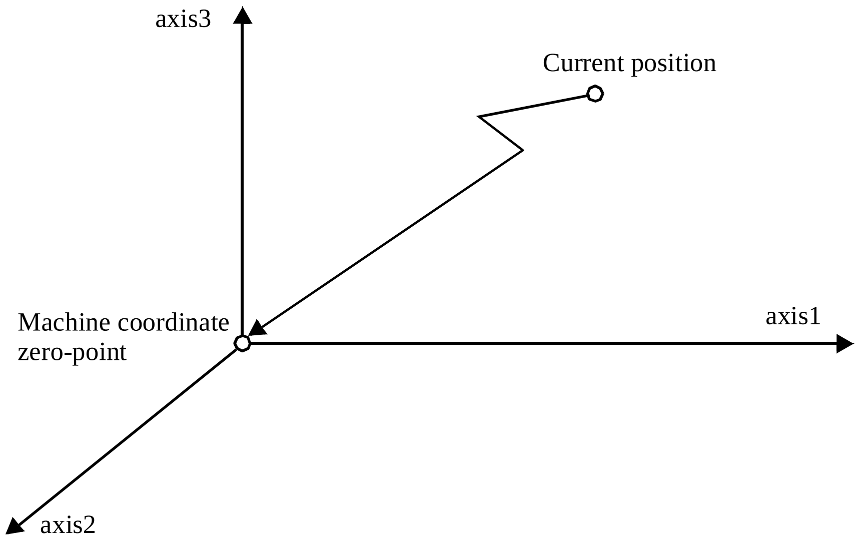


Figure 2.15: Move Path of the Zero-point Return Motion

- When the ZRN command is executed, the position to which the axis returns is set up as the zero-point of the machine coordinate. At the same time, the work coordinate set up previously by the current value change (POS) command is cancelled.
- When the ZRN command is executed, the machine coordinate and work coordinate become the same. Even though the machine coordinate (MVW) is designated, it is disabled until the next current value change (POS) command is executed.

- Zero-point return motion types are shown in the following table. They can be selected from the parameters.

Fixed Parameter 26 [Zero-point return format]	Name	Content
0	Zero-point return motion 1	3-stair deceleration format based on deceleration LS and C-phase pulse
1	Zero-point return motion 2	Zero-point return format based on zero-point LS
2	Zero-point return motion 3	3-stair deceleration format based on deceleration LS and zero-point LS
3	Zero-point return motion 4	Zero-point return format based on C-phase pulse

a. Zero-point Return Motion 1
Execute the zero-point return motion which is shown as follows. The position to which the axes return is the zero-point of the machine coordinate.

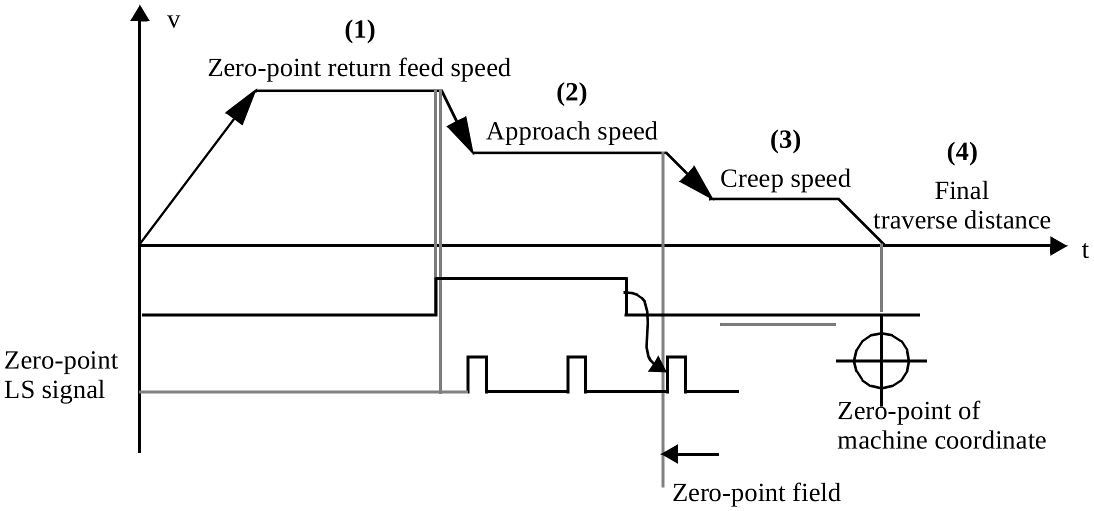


Figure 2.16: Motion Chart of the Zero-point Return Motion 1

(1) Starts moving in the direction designated in the “Zero-point return direction” parameter. The speed at this point is the value set up in the “Feed speed” parameter.

- Setup parameter 1 (OWxx00): [Motion mode b9: Zero-point return direction]
- Setup parameter 30 (OWxx22): [Fast feed speed]

(2) When the switch set up for deceleration turns the accel/decel LS signal ON, the feed speed decelerates to the value set up in the “Approach speed” parameter.

- Servo amplifier user constant (Cn-0022): [Zero-point approach speed 1]

(3) After the switch departs from the deceleration LS, the speed is changed to the value set up in the “Zero-point return traverse distance” parameter at the previous C-phase position.

- Servo amplifier user constant (Cn-0023): [Zero-point approach speed 2]

(4) By creep speed, the position to which the axes move (from the position at which the C-phase pulse is detected) only the distance set up by the “Zero-point return traverse distance,” is set up as the machine coordinate zero-point.

- Servo amplifier user constant (Cn-0028): [Zero-point return final traverse distance]

b. Zero-point Return Motion 2
Execute the zero-point return motion which is shown as follows. The position to which the axes return is the zero-point of the machine coordinate.

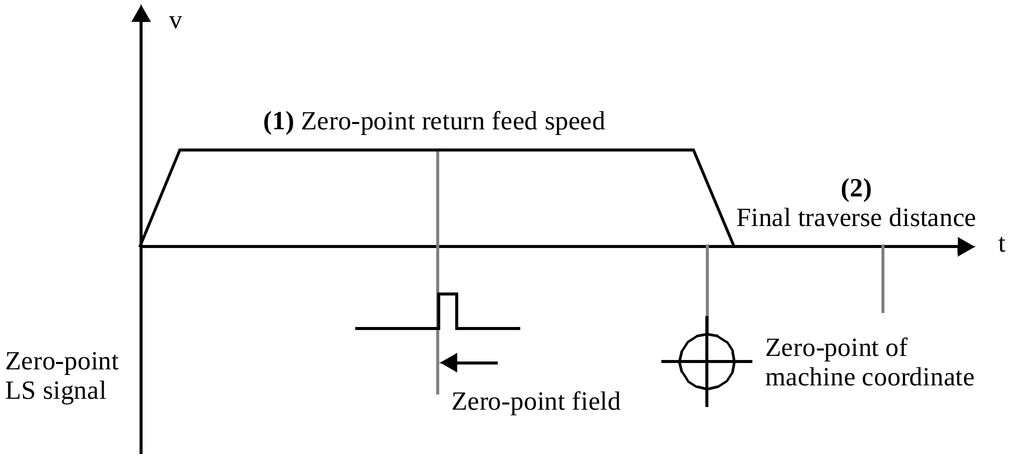


Figure 2.17: Motion Chart of the Zero-point Return Motion 2

(1) Starts moving in the direction designated in the “Zero-point return direction” parameter. The speed at this point is the value set in the parameter “Fast feed speed”.

- Setup parameter 1 (OWxx00): [Motion mode b9: Zero-point return direction]
- Servo amplifier user constant (Cn-0023): [Zero-point approach speed 2]

(2) Axes only move (from the position at which the zero-point LS is ON) the distance set up by the “Zero-point return traverse distance,” then stop. This position is set up as the zero-point of the machine coordinate. An axis which is not designated has neither the zero-point setting nor movement.

- Servo amplifier user constant (Cn-0028): [Zero-point return final traverse distance]

c. Zero-point Return Motion 3
The pulse signal based on another set zero-point LS can be used instead of the C-phase pulse in the zero-point return motion 1.

d. Zero-point Return Motion 4
The C-phase pulse can be used instead of the pulse signal based on the zero-point LS in the zero-point return motion 2.

■ Program Example

The program example of the zero-point return motion 4 in the ABS mode is shown as follows:

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

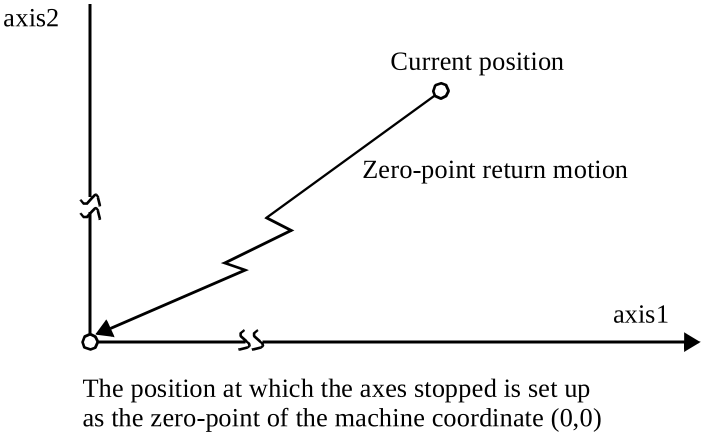

axis2
Current position
Zero-point return motion
axis1
The position at which the axes stopped is set up
as the zero-point of the machine coordinate (0,0)

Figure 2.18: Program Example of the Zero-point Return Motion 4

Parameters and motion commands regarding automatic accel/decel of the interpolation feed are shown as follows:

• Zero-point return direction:	Setup parameter [Run mode setting, (OWxx00b9) zero-point return direction]
• Zero-point feed speed:	Setup parameter 30 [Fast feed speed (OLxx22)]
• Approach speed:	Cn-0022 [Zero-point return approach speed 1]
• Creep speed:	Cn-0023 [Zero-point return approach speed 2]
• Final traverse distance:	Cn-0028 [Zero-point return final traverse distance]

Supplement

(1) The designated position after the axis must always be zero to avoid an alarm.

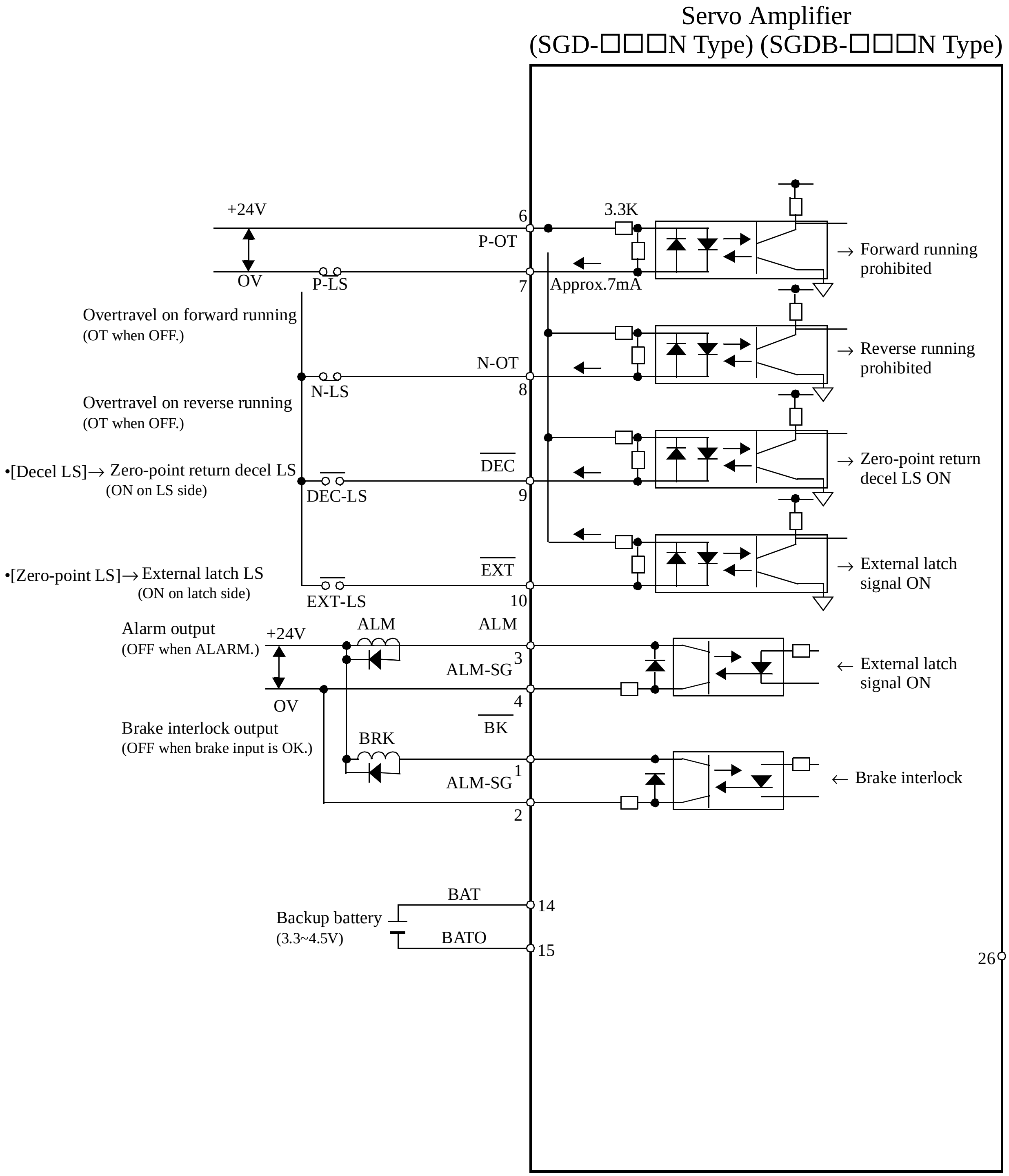
(2) The deceleration limit switch and the zero point limit 1 switch are both inputs on connector 1CN of the servo amplifier for each axis.

■ Zero-point Return Signal Connection

The “Deceleration LS” and “Zero-point LS” used with the zero-point return are connected to the 1CN of the servo amplifier.

- The “Deceleration LS”: 1CN, 9pin, Zero-point deceleration LS (/DEC)
- The “Zero-point LS”: 1CN, 10pin, External latch input (/EXT)

The connection is illustrated as follows:


Servo Amplifier
(SGD-□□□N Type) (SGDB-□□□N Type)
+24V
OV
P-LS
6
P-OT
7
3.3K
Approx.7mA
→ Forward running prohibited
Overtravel on forward running
(OT when OFF.)
N-OT
N-LS
8
→ Reverse running prohibited
Overtravel on reverse running
(OT when OFF.)
•[Decel LS]→ Zero-point return decel LS
(ON on LS side)
DEC-LS
DEC
9
→ Zero-point return decel LS ON
•[Zero-point LS]→ External latch LS
(ON on latch side)
EXT-LS
EXT
10
→ External latch signal ON
Alarm output
(OFF when ALARM.)
+24V
ALM
ALM
ALM-SG
3
OV
4
← External latch signal ON
Brake interlock output
(OFF when brake input is OK.)
BRK
BK
ALM-SG
1
2
← Brake interlock
Backup battery
(3.3~4.5V)
BAT
14
BATO
15
26

2.1.6 Skip Command (SKP)

■ Outline

The skip command (SKP) is a command that moves up to 14 axes simultaneously (by the interpolation feed speed (F)) from the current position to the end position with the linear interpolation motion. Once the skip signal is ON during movement, the axes which are moving decelerate to stop, and the remaining moving amount is cancelled.
If the skip input signal is ON, the axial motion decelerates to stop, and the remainder of the moving amount in the block is cancelled while the axes are moving in the block in which the SKP command is designated. The motion control that corresponds with the external conditions can be programmed by the SKP command.

■ Detailed Explanation

The designating method of the SKP command is shown as follows:

$$\text{SKP}\quad \underbrace{[\text{axis1}]\text{—}[\text{axis2}]\text{—}\bullet\bullet\bullet}_{\text{A}}\quad \underbrace{\text{F—}}_{\text{B}}\quad \underbrace{\text{SS—;}}_{\text{C}}$$

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

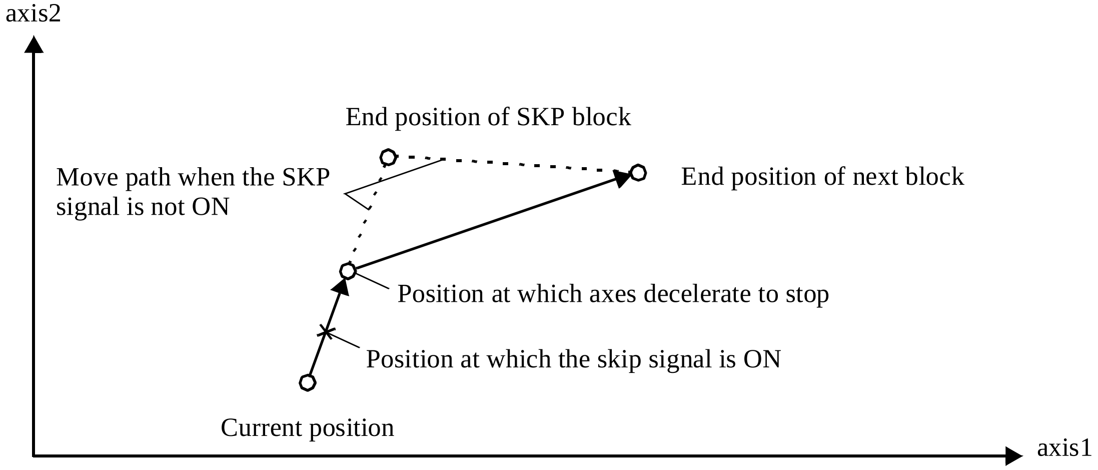

axis2
End position of SKP block
Move path when the SKP signal is not ON
End position of next block
Position at which axes decelerate to stop
Position at which the skip signal is ON
Current position
axis1

Figure 2.19: Move Path of the SKP Command

- The SKP input signal can control up to 14 axes that are divided into 4 groups. Each group can be assigned 2 points.
- The SKP input signal can be selected (skip selection SS) by writing either number 1 or 2. The number selected corresponds with the SKP input signal previously assigned in the group definition window.

2.1.7 Time Designation Positioning (MVT)

■ Outline

The time designation positioning (MVT) command limits the axis feed speed so that each axis moves from the current position to the end position with the positioning (MOV) motion, and completes the positioning in the designated time. Because this command is not the interpolation motion command, there is no guarantee that all designated axes can complete the positioning simultaneously. A time lag exists, depending on the accel/decel setting. Up to 14 axes can be designated simultaneously. An axis which is not designated does not move.

■ Detailed Explanation

The designating method of the MVT command is shown as follows:

```
MVT  [axis1]—[axis2]—•••        T—;
     ___________________   _______________________
     Designated position   Positioning time (msec)
```

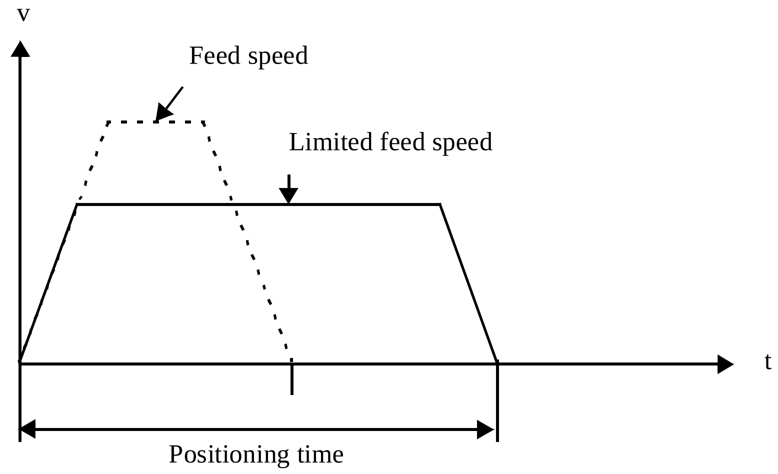


Figure 2.20: Time Designation Positioning (MVT)

- Positioning cannot be completed in the designated time when using override.
- When using filter, positioning time delays the amount of filter time constant.

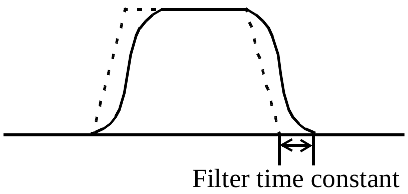


Figure 2.21: Positioning Time Delay When Using Filter

Important Points

- T0: Positioning time=0.
- If there is 0 axis, an alarm occurs to indicate the moving amount.

Supplement

After executing the time designation positioning (MVT), the value of parameter 30 "Fast feed speed (OLxx22)" of each axis used by the MVT command, is rewritten. Therefore, the speed set up by the VEL command "Feed speed change" is also rewritten. The feed speed must be set up again by the VEL command after executing the MVT command.

■ Program Example

The program example of the time designation positioning (MVT) in the ABS mode is shown as follows:

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

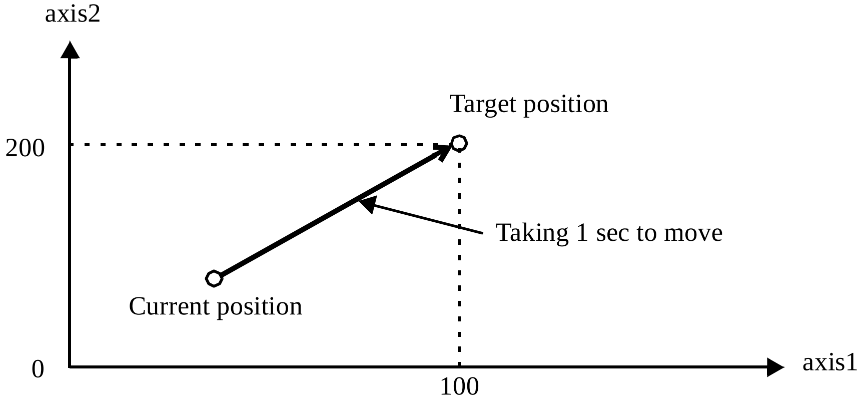


Figure 2.22: Program Example of the Time Designation Positioning (MVT)

2.1.8 External Positioning (EXM)

■ Outline

When executing the positioning command, once the external positioning (EXM) signal is input, the next block is executed after moving the designated distance.

- The EXM command is only enabled for one axis
- Move the designated distance if the signal is not input.
- The external positioning signal is connected to the external latch input (connector 1CN, Pin 10) on the servo amplifier of the designated axis.

■ Detailed Explanation

The designating method of the EXM command is shown as follows:

```
EXM   [axis]—       D—;
      ─────────    ─────
          A          B

       A: Designated distance
       B: Moving amount (incremental amount only)
          after input of the external positioning signal
```

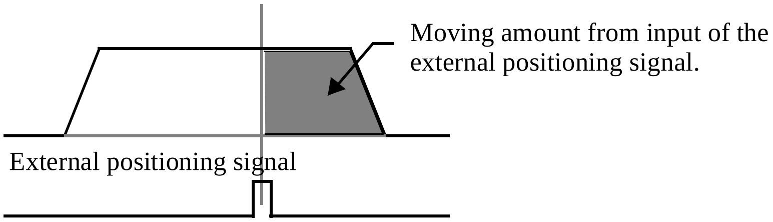

Moving amount from input of the external positioning signal.
External positioning signal

Figure 2.23: External Positioning (EXM) Signal

If a negative value is designated as the moving amount, the axis moves in a negative direction after decelerating to a stop.

- Only the designated distance amount is executed during machine lock, ignoring the external input signal.
- Movement continues for the designated distance if the signal is not input.

Important Points

(1) The external positioning signal is 1CN 10pin of the servo amplifier; connect it to the external latch (/EXT).

(2) The external latch (/EXT) input signal is also used for the zero-point return using the “Zero-point LS”. Therefore, be careful when using.

2.2 Control Command

The control commands are the commands that control axis motion. This section explains how to program basic control commands.

2.2.1 Absolute (ABS) Mode

⚠ CAUTION

Motion differs, depending on whether the coordinate terms are designated by the absolute mode or incremental mode.
Before operating, verify that the ABS/INC command is designated correctly.

Forgetting this verification carries a risk of tool damage, as well as bodily injury due to interference.

■ Outline

The absolute (ABS) mode commands axial motion coordinate terms as an absolute value. Once the ABS mode command is designated, it remains until the next incremental (INC) mode command is designated. When the power supply is connected, it defaults the ABS mode command.

■ Detailed Explanation

The designating method of the ABS mode command is shown as follows:

```
ABS;
Or,
ABS MOV [axis1]—;
```

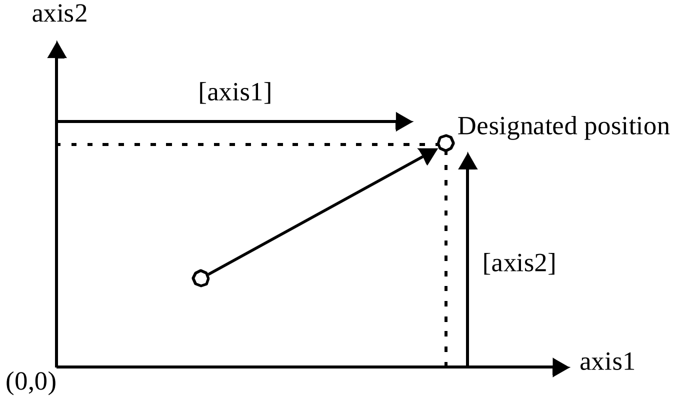


Figure 2.24: Absolute Value on the Work Coordinate

■ Program Example

The program example of the ABS mode command is shown as follows:

```
ABS MOV [axis1]100 [axis2]200;
MOV [axis1]50 [axis2]100;
MOV [axis1]100;
MOV [axis2]50;
```

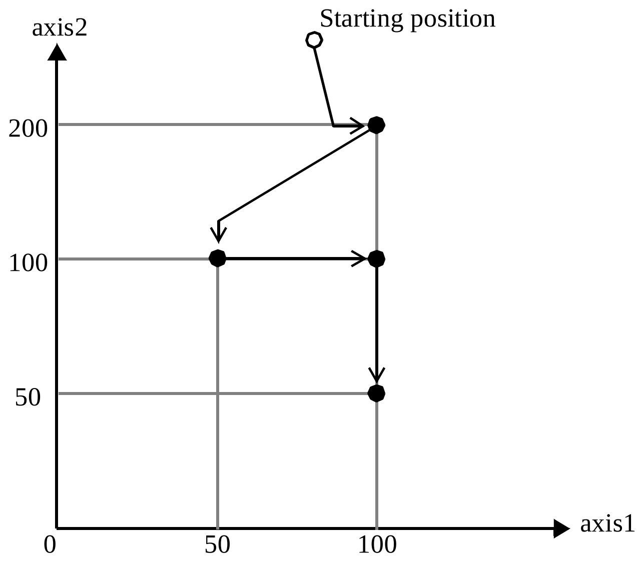


Figure 2.25: Program Example of the ABS Mode Command

2.2.2 Incremental (INC) Mode

■ Outline

The incremental (INC) mode commands the axial motion coordinate terms as the incremental value from the current position of the work coordinate. Once the INC mode command is designated, it remains until the next absolute (ABS) mode command is designated. When the power supply is connected, it defaults the ABS mode command.

■ Detailed Explanation

The designating method of the INC mode command is shown as follows:

```
INC;
Or,
INC MOV [axis1]—;
```

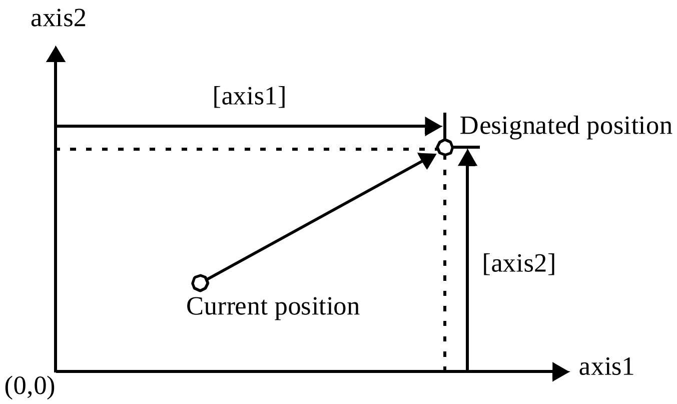


Figure 2.26: Incremental Value of the Work Coordinate

■ Program Example

The program example of the INC mode command is shown as follows:

```
INC MOV [axis1]100 [axis2]200;
MOV [axis1]50 [axis2]100;
MOV [axis1]100;
MOV[axis2]50;
```

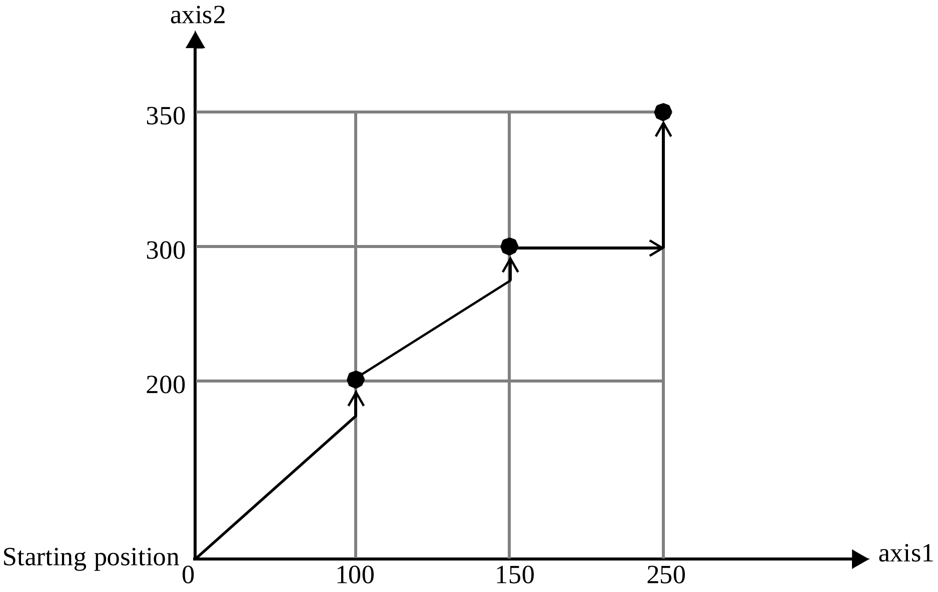


Figure 2.27: Program Example of the INC Mode Command

2.2.3 Current Value Change (POS)

⚠ CAUTION

Use caution when using the Current Value Change (POS) command.

The current value change (POS) command creates new “work coordinate values.” If the POS command is designated incorrectly, the moving actions will be totally different than expected. Therefore, before operation, **CHECK** that the correct “work coordinate values” are designated.

Forgetting this verification carries a risk of tool damage, as well as bodily injury due to interference.

■ Outline

The current value change (POS) command creates a new coordinate by rewriting the current position to the desired coordinate value. In this manual, this newly created coordinate is called the work coordinate. The moving command designated after the POS command moves on the work coordinate.

■ Detailed Explanation

The designating method of the POS command is shown as follows:

```
POS     [axis1]—[axis2]—•••;
        ────────────────────────
        Desired coordinate value
```

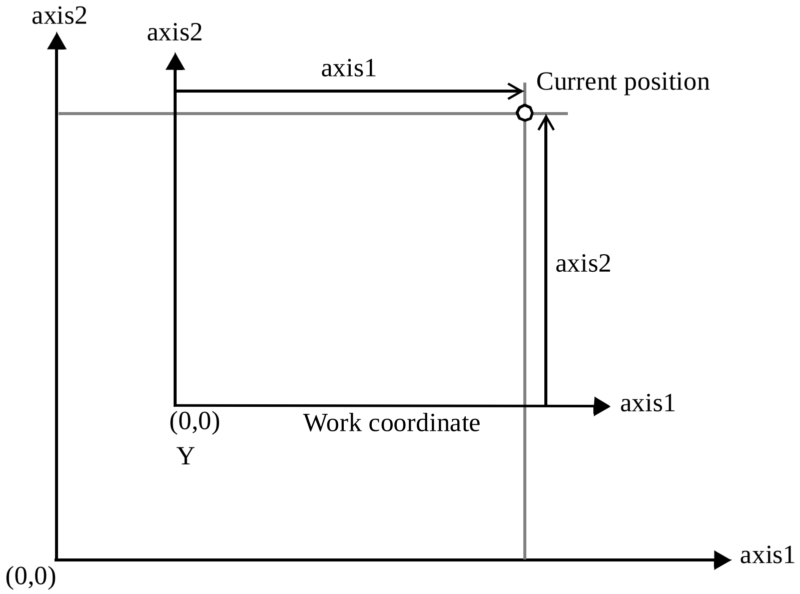


Figure 2.28: POS Command (current value change)

- The work coordinate based on the POS command can be switched as many times as desired. Set the machine coordinate first. The POS command does not affect the machine coordinate.
- Up to 14 axes can be designated by the POS command. An axis for which the designation is omitted cannot rewrite the current value.
- If the moving command on the work coordinate exceeds the maximum designation value after converting the machine coordinate, it cannot be designated.
- The setting status of the machine coordinate and work coordinate are shown in the following table.

Product Status	Incremental Detection System	Absolute Detection System
After the power supply is connected	Machine coordinate: Temporary setting *a Work coordinate: Cancel *c	Machine coordinate: Yes *b Work coordinate: Cancel
After execution of zero-point return (ZRN)	Machine coordinate: Setting Work coordinate: Cancel	Work coordinate: Cancel
After execution of the POS command	Work coordinate: Setting	Work coordinate: Setting
After operation of the zero-point setting	Work coordinate: Cancel	Machine coordinate: Setting Work coordinate: Cancel

a. Temporary setting
Sets up the machine coordinate whose zero-point is the current position after the power supply is connected. After this, the soft limit function is disabled if the zero-point return is not executed.

b. Yes
Creates the machine coordinate by the position data from absolute value detection encoder.

c. Cancel
Cancels the work coordinate which was previously set; it is equal to the machine coordinate.

2.2.4 Coordinate Plane Designation (PLN)

■ Outline

The coordinate plane designation command defines 2 logical axes set up in the parameters as the coordinate plane. Before executing the coordinate plane designation command, the coordinate plane must be designated.
Once the coordinate plane is designated, it is enabled until the next definition or the end of the program.

■ Detailed Explanation

The designating method of the PLN command is shown as follows:

```
        Horizontal axis     Vertical axis
PLN           [axis1]       [axis2];
              ______________________
              Designate 2 axes on
              the coordinate plane
```

■ Program Example

The program example of the PLN command is shown as follows:

PLN [A][B]; (designate the plane structured by A and B axis)
MCW [A]50 [B]50 R50 F1000;
Starting position: A = B = 0

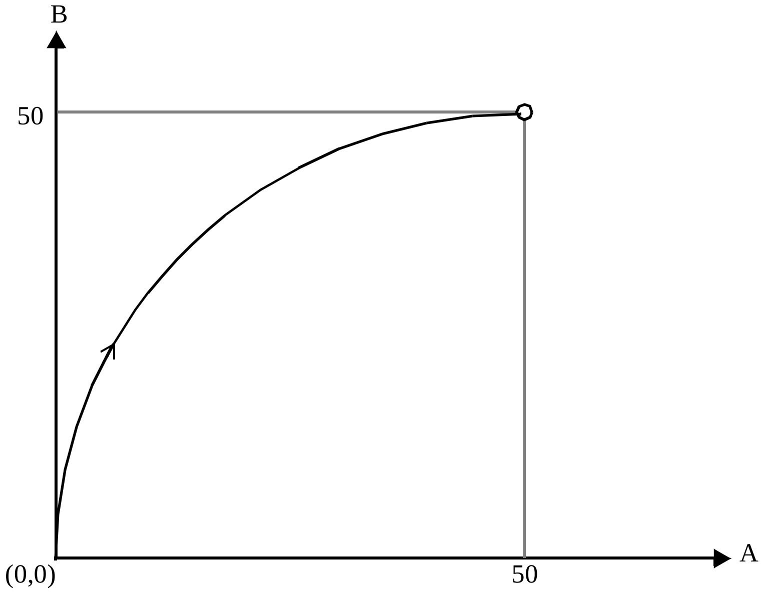


Figure 2.29: Program Example of the Coordinate Plane Designation (PLN)

2.2.5 Machine Coordinate Command (MVM)

⚠ CAUTION

The machine coordinate (MVM) command temporarily locates the coordinate position on the "machine coordinate". If the MVM command is designated without confirming the zero-point position of the "machine coordinate," unexpected movement will occur. Therefore, before operation, **CHECK** that the correct position on the "machine coordinate" is designated.

Forgetting this verification carries a risk of tool damage, as well as bodily injury due to interference.

■ Outline

The machine coordinate command (MVM) is used when attempting to temporarily move the machine coordinate, after the work coordinate has been set up differently than the machine coordinate by the current value change (POS) command.
From the next designation, move temporarily to the absolute coordinate position by the positioning (MOV) command or linear interpolation (MVS) command. This command operates in the ABS mode, regardless of ABS/INC mode designation.
The MVM command is only enabled in the designated block. For example, it sets up the movement on the work coordinate by the linear interpolation (MVS) command in the following block. Refer to the following figure.

■ Detailed Explanation

The designation method of the MVM command is shown as follows:

```
MVM MOV •••;
Or,
MVM MVS •••;
```

■ Program Example

The program example of the MVM command is shown as follows:

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

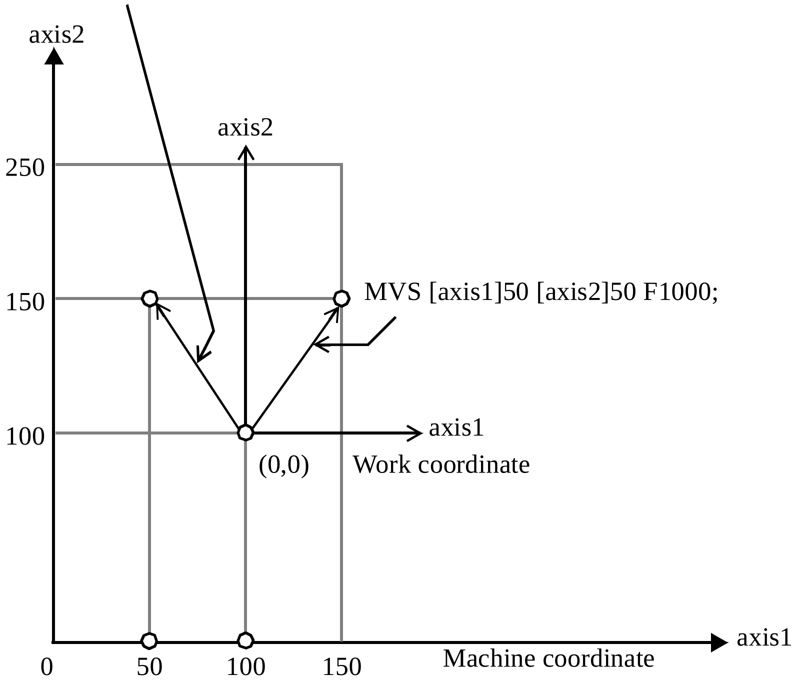


Figure 2.30: Program Example of the Machine Coordinate Command (MVM)

2.2.6 Program Current Position Update (PLD)

■ Outline

During the motion program operation, when an axis is moved by programs other than the motion program (such as JOG motion, STEP motion, or when an axis is moved by the user functions), the “program current position” does not move. In this case, if the program continues to execute the motion program, the axis moves only to the position at which the moving amount is manually shifted. In order to solve this problem, the PLD command is used to update the “program current position.”
This command does not correspond to the block operation. In other words, it is not a command which can be stopped by the block operation mode.

■ Designating Methods

```
PLD  [axis1] [axis2]... [axis];
```

■ Program Example

During manual operation of the motion program:

```
MPM001 “GROUP1”
MOV [axis1] 1000; ← Here, [axis1] is moved by JOG.
PLD [axis1]; ← the “program current position” is updated.
MOV [axis1] – 1000;
```

When the axis is moved by the motion program user functions:

```
MPM001 “GROUP1”
MOV [axis1] 1000;
UFC FNC10 MB00000 IW00100 MB00020
   ← [axis1] is moved by the user functions.
PLD [axis1]; ← the “program current position” is updated.
MOV [axis1] – 1000;
```

Supplement

The PLD command is executed depending on the user’s needs. Sometimes, even though manual movement applies during motion program operation, the PLD command is not used.

2.2.7 Timed Wait (TIM)

■ Outline

The timed wait (TIM) command is used to enter the next block after waiting the time designated by the character (T), from the next designation.
The TIM command cannot be designated simultaneously with other commands.

■ Detailed Explanation

The designating method of the TIM command is shown as follows:

```
TIM T—;
   Waiting time
```

The possible range that the (T) can designate is 0.01 ~ 599.99 [sec].

- When the time (T) is designated by the integer number, 1 = 0.01 [sec]. The position of the decimal does not affect the parameter setting.

■ Program Example

The program example of the TIM command is shown as follows:

```
MOV [axis1]100;
TIM T250;
    2.5sec
```

Execute the [TIM] command after the positioning is completed.

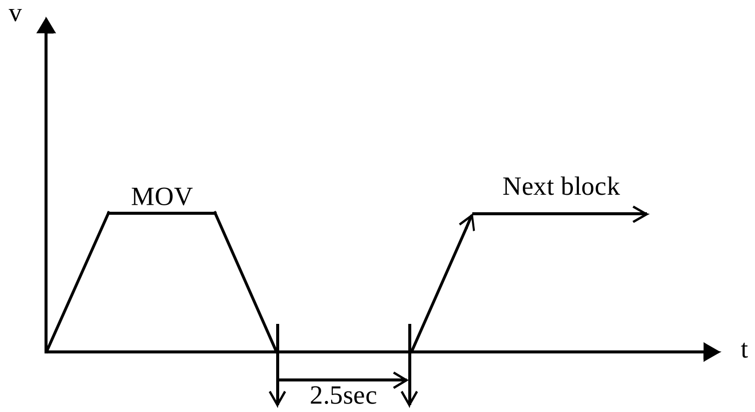


Figure 2.31: Program Example of the Timed Wait (TIM) Command

2.2.8 Program End (END)

■ Outline

The program end (END) command ends program operation. In this block, it cannot be designated simultaneously with other commands.

■ Detailed Explanation

The designating method of the END command is shown as follows:

END;

The program ends

According to this command, after the block execution is finished, the program operation is ended.

• When the previous block is designated by a moving command, the program operation is ended after the in-position check is finished.

3 Advanced Programming

This chapter explains how to program motion control commands including advanced control commands, and speed/acceleration commands.

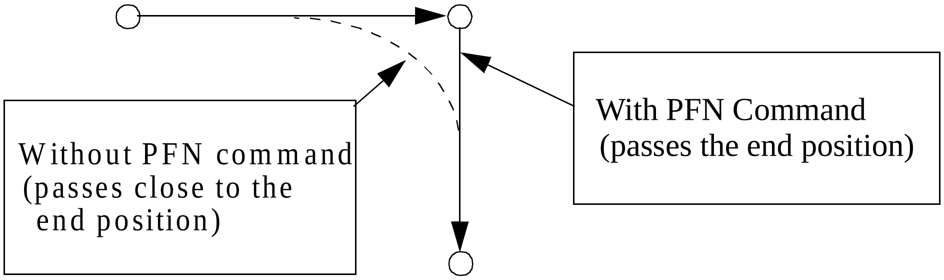
3.1 Advanced Control Commands

In this section, the details of programming methods for advanced commands within the motion control commands are explained. Since they are comparatively complicated, beginners should skip this section and go to item 1.2 Motion Programing Method.

3.1.1 In-Position Check (PFN) Command

■ Outline

During interpolation command movement, after an axis feed is complete, the In-position Check (PFN*) command waits until the axis has entered the In-position Check Range. When it enters the In-position Check Range, the next block is executed.
During interpolation command corner movement, this command is used to pass the designated end position.



■ Detailed Explanation

There are two designating methods for the PFN command.

a) When designating to the block simultaneously with the interpolation commands

```
MVS [axis1]100.[axis2]200.F1000PFN;
```

Execute the In-position Check for the axis designated by the MVS command, then proceed to the next block.

b) When designating independently

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

* **In-position Check (PFN)**: A function that detects if an axis has entered the positioning completion range after the designated block movement starts deceleration.

If axis1 and axis2 enter the In-position Check, the next block is executed.
According to the above command, the In-position Check is executed to verify if the axis movement in the block before the PFN command has entered the Positioning Completion Range, then the next block execution is started.

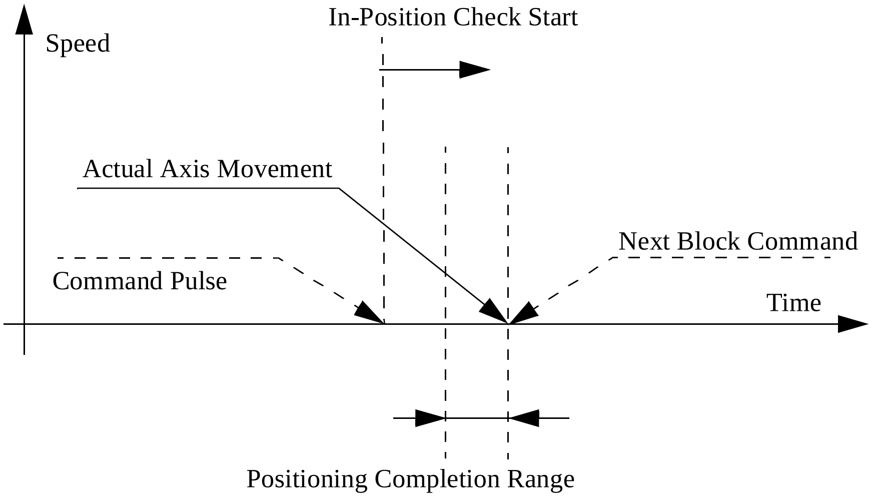


Figure 3.1: In-position Check Process

- The Positioning Completion Range is set up in each servopack user parameter.
  - Cn-001B Positioning Completion Width
- As indicated below, two methods are available for the PFN command: a command designated simultaneously with the interpolation commands, and a command designated independently to the block in which the In-position Check is executed.

	Designation Type	Designating Method	Notes
1	Designated for the same block as the interpolation commands	MVS[axis1]100. [axis2]200. F1000 PFN;	In-position check for the axis designated with the MVS commands, then proceeds to the next block.
2	Independent designation Axis Designation	PFN [axis1] [axis2]	In-position check for the designated axis (check if the axis can be used) then proceeds to the next block.

■ Program Examples

```
MVS [axis 1] 200. F100 PFN;     Linear Interpolation Movement Completion Wait
MOV [axis1] 400. [axis2] 400;   Movement according to the positioning
```

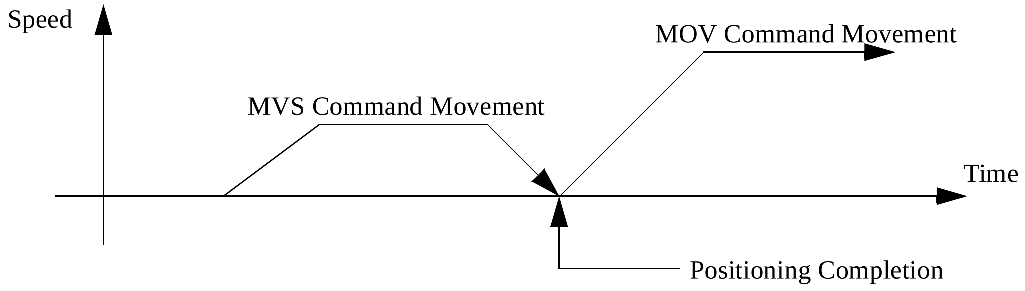


Figure 3.2: In-position Check (PFN)

Supplement

When the PFN command is not sent, next block execution is started after MVS command pulse distribution is complete.

3.1.2 Second In-Position Check (INP) Command

■ Outline

When moving according to the interpolation, the Second In-position Check (INP) command is sent to pass close to the end position.

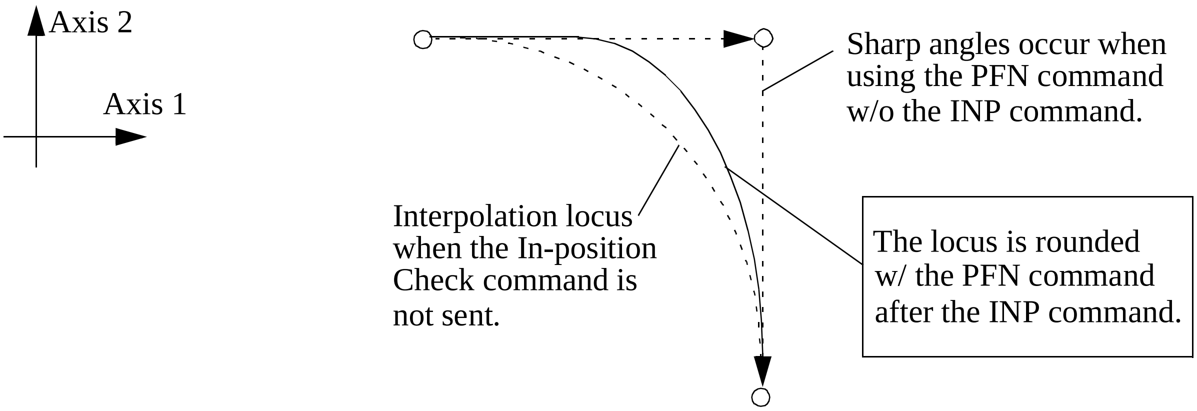

Axis 2
Axis 1
Sharp angles occur when using the PFN command w/o the INP command.
Interpolation locus when the In-position Check command is not sent.
The locus is rounded w/ the PFN command after the INP command.

Figure 3.3: Second In-position Check Command

■ Detailed Explanation

The INP command method is shown as follows:

```
INP    [axis1]--[axis2]--...;
        Positioning Completion Range
```

According to the above command, when the PFN-designated interpolation movements which are commanded after the INP command, and/or the independent PFN-designated interpolation movements are verified having entered the Second Positioning Completion Range, the next block execution is started. When there are multiple moving axes, the next block is started after each axis has entered the Second Positioning Completion Range.

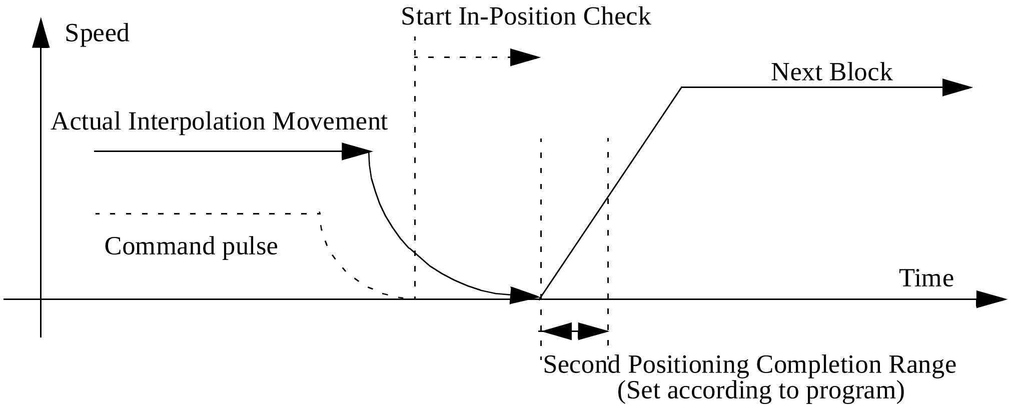

Speed
Start In-Position Check
Next Block
Actual Interpolation Movement
Command pulse
Time
Second Positioning Completion Range
(Set according to program)

Figure 3.4: Second In-Position Check Command Method

- A range for Second Positioning Completion Range setting is as indicated below:

1 ~ 65535[Command Units]

However, as indicated in the diagram above, the Second In-position Check is started at the time that the pulse distribution of the commanded block is complete. Therefore, if a very large value is set in the Second Positioning Completion Range, the Second In-position Check completes at the deceleration start point, and then proceeds to the next block.

- Once the INP command is set up, the Second In-position Check is valid for all interpolation commands until cancellation.
- For cancellation of the Second In-position Check, give the [0] command to the axis to be cancelled in the Second Positioning Completion Range. Only the axis receiving the [0] command will be cancelled.
- It is now possible to send the In-position Check (PFN) command. When the command is sent, the In-position Check will occur according to the parameter “Positioning Completion Range” in spite of the Second Positioning Completion Range.

■ Program Examples

```
ABS MOV [axis 1]0[axis2]0;     Zero-point positioning
INP [axis 1](a)[axis2](b);     Second Positioning Completion Range set-up
MVS [axis1]100) PFN;           X-axis direction / Linear interpolation
MVS [axis2]100 PFN;            Y axis direction / Linear interpolation
MVS [axis1]-100 PFN;           X axis direction / Linear interpolation
```

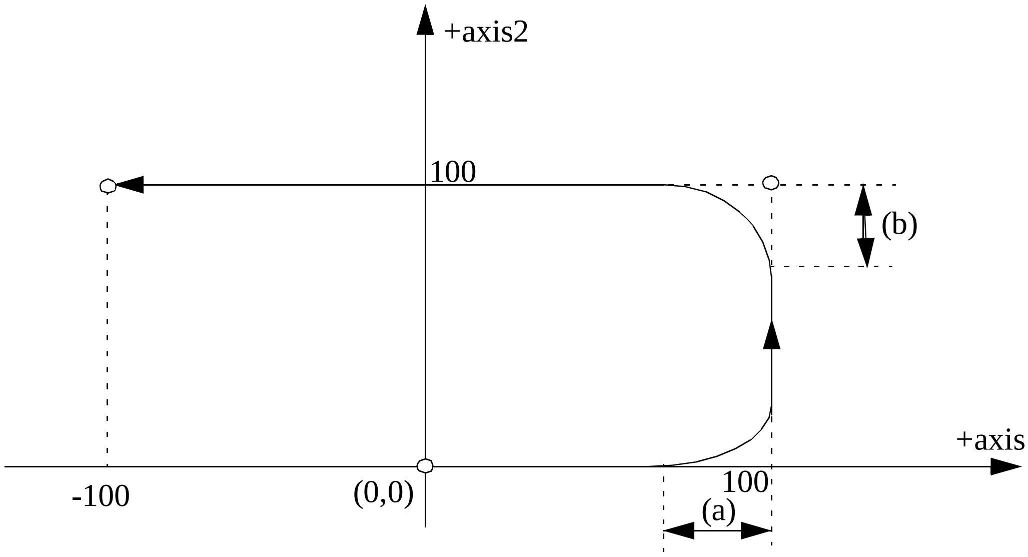


Figure 3.5: Program Example of the Second In-position Check

3.1.3 Ignore Single Block (SNG) Command

■ Outline

The Ignore Single Block (SNG) command is used when continuous operation is needed for a special block, during the operation of the Single Block Operating Mode*.

■ Detailed Explanation

The SNG command method is shown as follows:

```
SNG: Axis Movement Command
 and
SNG: No Axis Movement Command
```

According to the above command, even when the Single Block Operating Mode is in effect, continuous operation will take place within the block described by the SNG command without single block stoppage.

- For commands other than the axis movement and speed/acceleration, the single block operation cannot be executed. Therefore, it would be meaningless to use these commands with the SNG command.
- The commands which can be designated by the SNG command are shown as follows:

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

■ Program Examples

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

Even in the single block operating mode, continuous operation occurs.

* **Single Block Operating Mode**: With the Single Block Operating Mode, single block stoppage occurs in each block. However, the commands other than the axis movement and speed/acceleration commands cannot perform the single block stoppage. Moreover, use caution when operating, because each block stoppage in the single block operation occurring in parallel will change according to motion program content.

3.1.4 User Function Call-out (UFC) Command

■ Outline

The User Function Call-out (UFC) command calls up and executes the functions created by the user, by designating the function name.

■ Detailed Explanation

The UFC command method is shown as follows:

UFC Function Name, Input Data, Input Address, and Output Data

Function Name	:	ASCII 8 bytes
Input Data	:	Up to 16 bits
Input Address	:	Up to 1 addresses
Output Data	:	Up to 16 bits (1 bit is always needed)

Note: Input data and input address can be omitted.
The output data means that there is no input data. It always requires a minimum of 1.

According to the above command, the user functions are called out. When the execution of the user functions is finished, execution proceeds to the block following the UFC.

■ Program Examples

UFC KANSUU MB00000 IW0010 MB00020, MA00100 ,
Function Name Input Data Input Address
MB00001 MW00200 ML00201;
Output Data

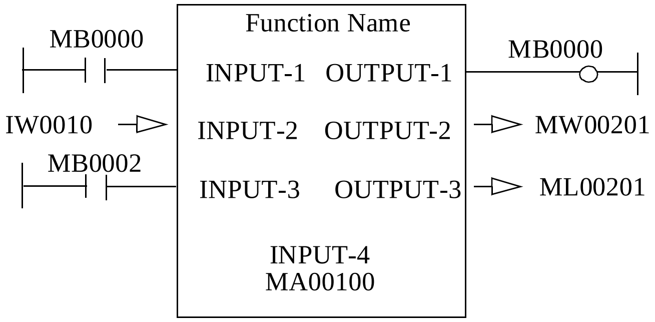
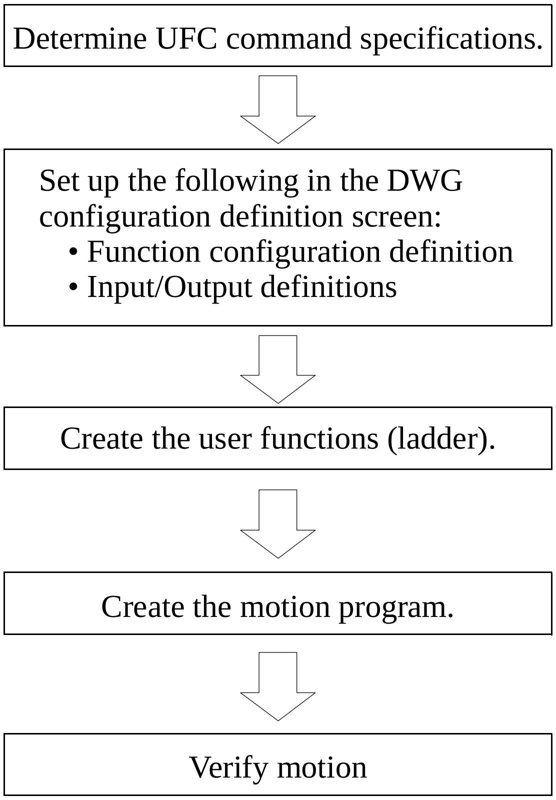


Figure 3.6: Program Example of the User Function Call-out (UFC)

■ UFC Command Creating Steps

UFC command creating steps are described below.



Determine UFC command specifications.

- Determine input/output number and data type.
- Determine function name.

Set up the following in the DWG configuration definition screen:
- Function configuration definition
- Input/Output definitions

Input by using the CP-717.

Create the user functions (ladder).

Use the same method as the DWG creation. However, the register type to be used varies.

Create the motion program.

Create with the format of the UFC:
Function name, Input data, Input address,
and Output data.

Verify motion

■ Register Type Used in the User Function

The types of the register are shown as follows:

• B-VAL →	Bit-type
• I-VAL →	For future use
• L-VAL →	For future use
• I-REG →	Integer number type
• L-REG →	32-bit integer or real number-type

■ Relationship Between the I/O Register and Function Register

The following illustration shows the interaction between the I/O register designated by the UFC command and the function register.

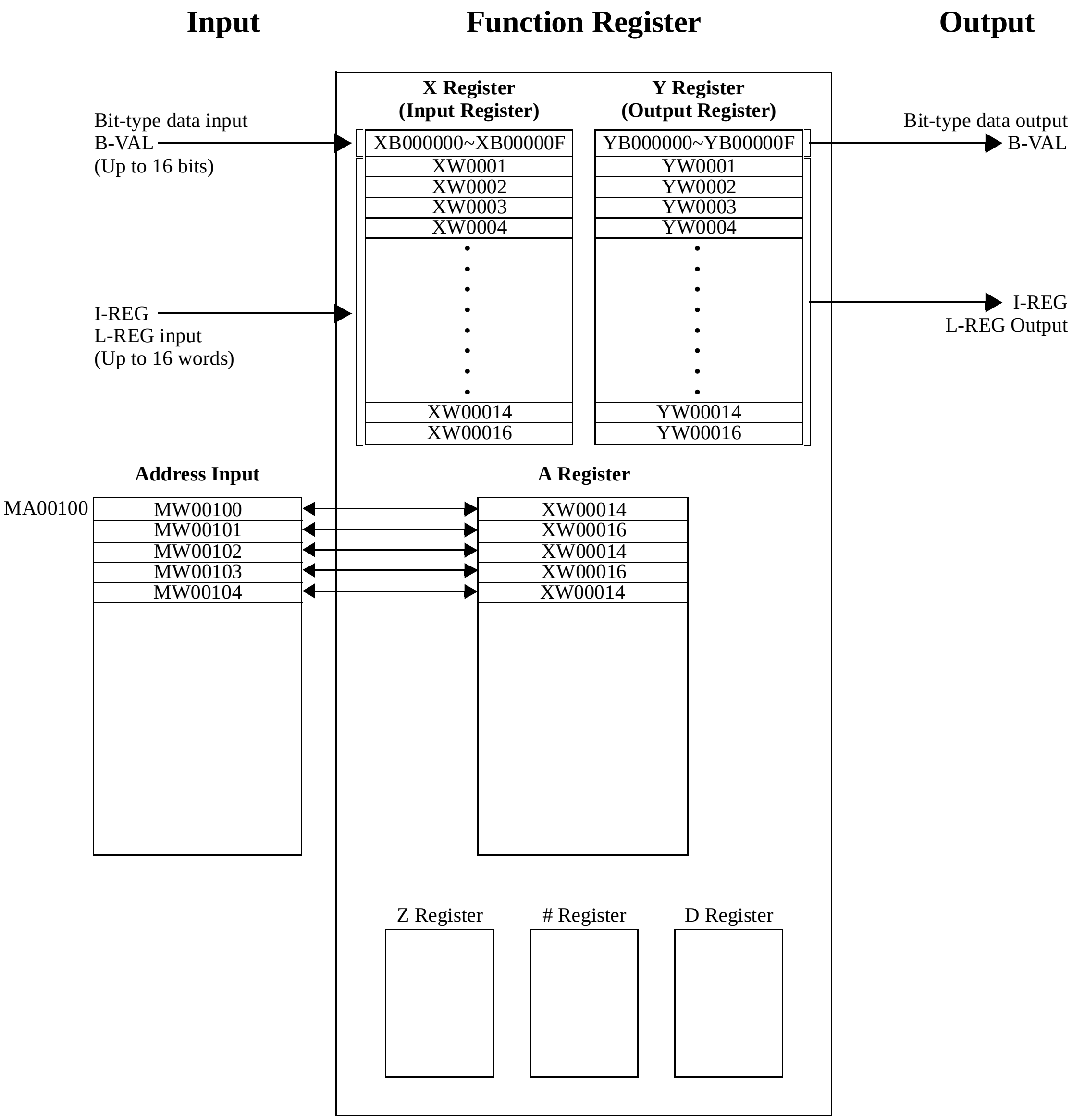


Figure 3.7: I/O Register and Function Register Relationship

Supplement

The S, M, I, O, and C registers are used in the same way as the DWG registers.

Each user function can use the 11 registers shown in the table below.

Function Register

Type	Name	Command Method	Content	Characteristic
X	Functional input register	XB, XW, XL, XFnnnnn	Function Input: • Bit input: XB000000~XB00000F • Integer input: XW000001~XW00016 • Double-length integer input: XL00001~XL00015 The register number nnnnn is displayed with a decimal.	Individual Functions
Y	Functional output register	YB, YW, YL, YFnnnnn	Function Input: • Bit input: YB000000~YB00000F • Integer input: YW000001~YW00016 • Double-length integer input: YL00001~YL00015 The register number nnnnn is displayed with a decimal.	
Z	Functional internal register	ZB, ZW, ZL, ZFnnnnn	Internal register owned by each function. It can be used as an internal process for the function. The register number nnnnn is displayed with a decimal	
A	Functional external register	AB, AW, AL, AFnnnnn	External register whose base address is address input value. It can be used for linking to (S, M, I, O, #, Dannnnn). The register number nnnnn is displayed with decimal.	
#	# Register	#B, #W, #L, #Fnnnnn (#Annnnn)	Register that only can be referenced in a program. Only the corresponding DWG can be referred. The actual range is designated in MotionWorks™ by the user.The register number nnnnn is displayed with a decimal.	
D	D Register	DB, DW, DL, DFnnnnn (DAnnnnn)	Register that is owned by each DWG. Only the corresponding DWG can be referred. The actual range is designated in Motion-Works™ by the user. The register number nnnnn is displayed with a decimal.	

Function Register (Continued)

S	System Register	SB, SW, SL, SFnnnnn (SAnnnnn)	Same as the DWG registers. Because these registers are the same as the DWG registers, use caution when referencing the same function from a DWG that has a different priority level.	Same as the DWG registers.
M	Data Register	MB, MW, ML, MFnnnnn (MAnnnnn)		
I	Input Register	IB, IW, IL, IFhhhh (IAhhhh)		
O	Output Register	OB, OW, OL, Ofhhhh (OAhhhh)		
C	Constant Register	CB, CW, CL, CFhhhhh (CAnnnnn)		

Note: Even within the functions, the SA, MA, IA, OA, DA, #A, CA can still be used.

An example of the input/output register delivery is shown as follows:

Motion program description

```
UFC TESTFUNC DB00000 DB000001 MW00030 MW00032 MA00100, DB0000002 MW00040
```

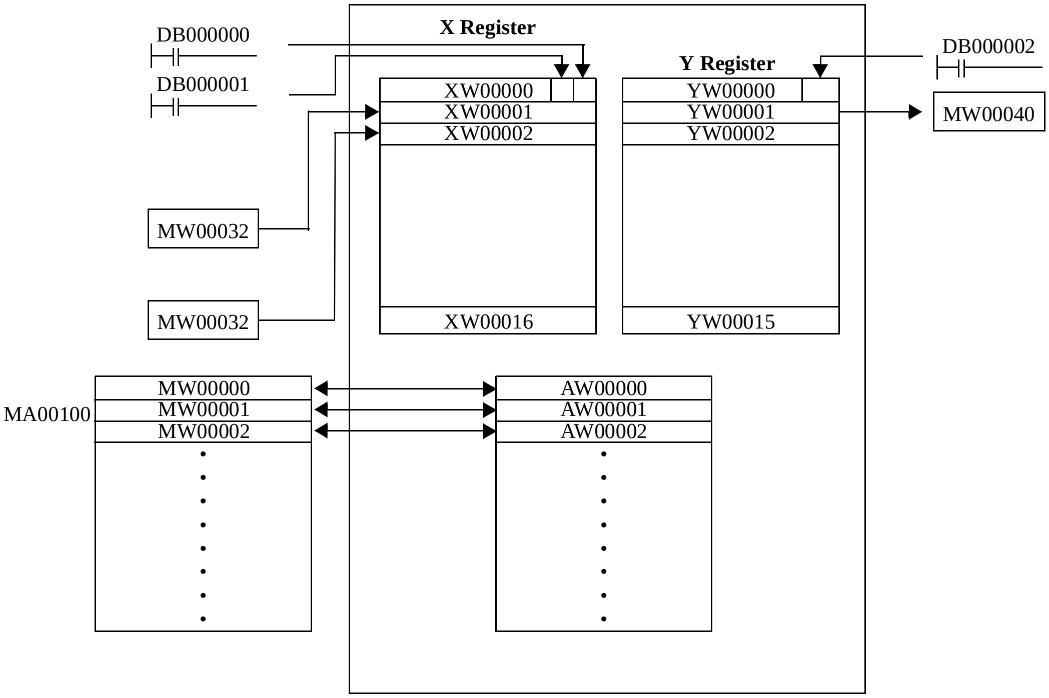


Figure 3.8: Motion Program Description

■ User Function Creation

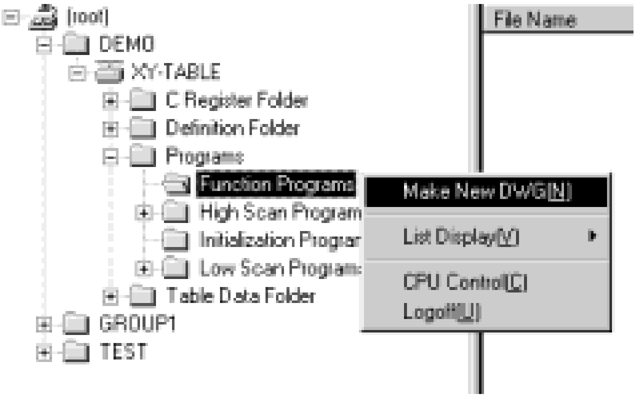
The steps for creating a user function are described with the example shown below.

Specifications	Designate servo axis number and speed data, then set them up in the parameter [Fast feed speed OLxx22].

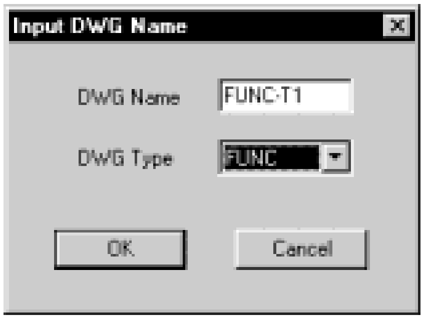
Motion program	MW00030 = Servo axis number (1 or 2)
	ML00032 = Fast feed speed
	UFL FUNC-T1 MW00030 ML00032, DB000001;

To create a user function, follow the steps below:

a. In the File Manager screen, open [Programs → Function Programs], right-click on Function Programs, then double-click on the [Make New DWG (N)].



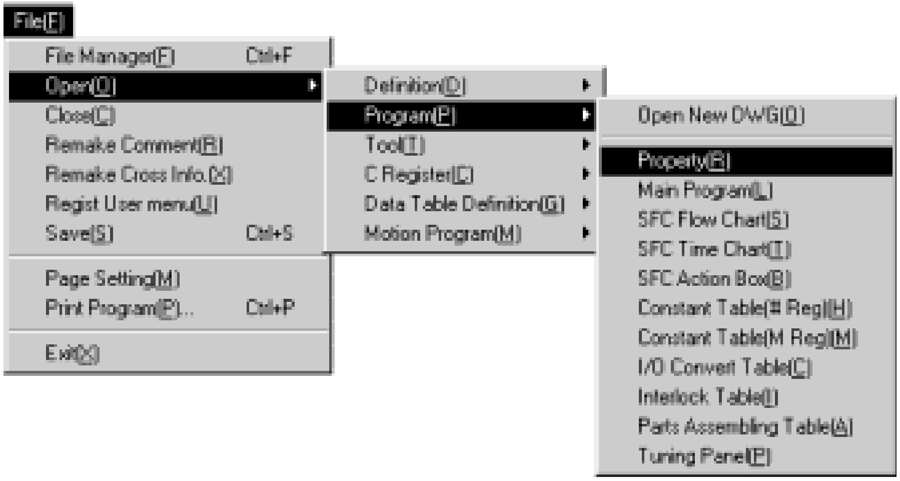
b. Set up DWG name and type in the message box [Input DWG Name], then click on the **OK** button.



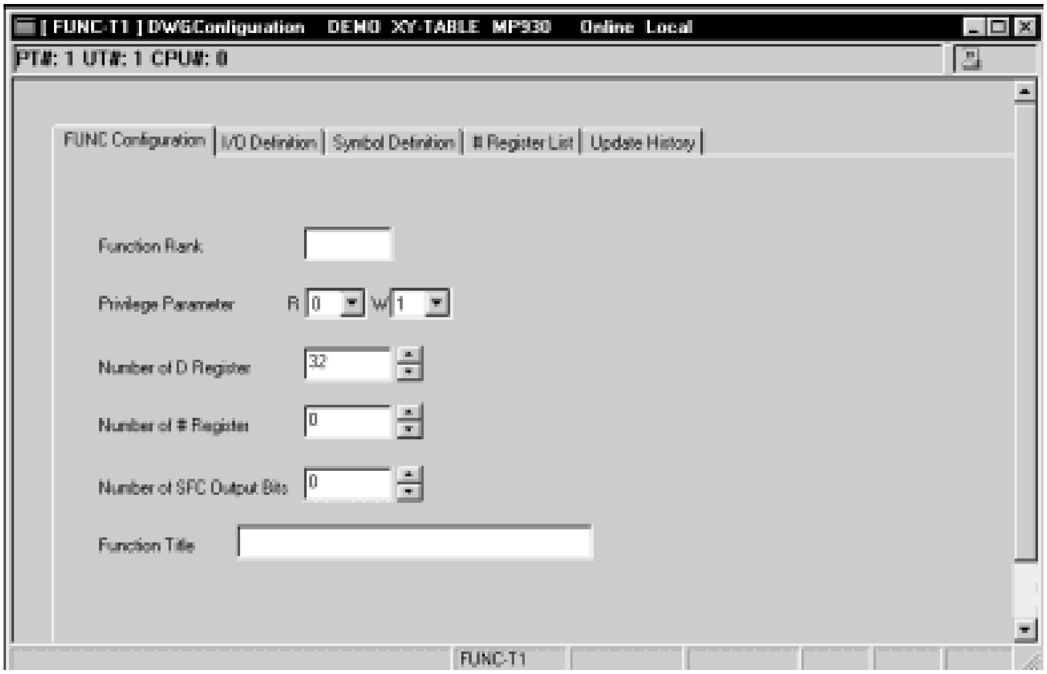
DWG Name FUNC-T1

DWG Type FUNC

c. The ladder screen (empty screen) appears. Select **File (F)** → **Open (O)** → **Program (P)** → **Property (R)**. The DWG Configuration screen is opened.



d. Set up the number of the D register in the Function Configuration of the DWG Configuration screen. (The default value is 32.)



[FUNC-T1] DWGConfiguration DEMO XY-TABLE MP930 Online Local

PT#: 1 UT#: 1 CPU#: 0

Function Rank

Privilege Parameter R 0 W 1

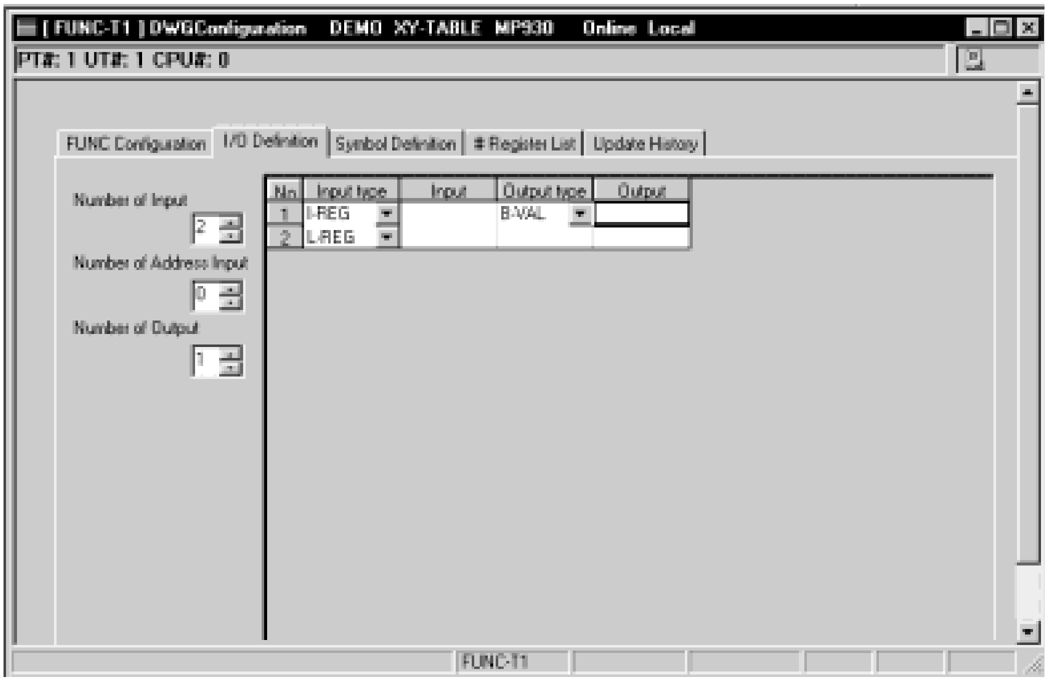
Number of D Register 32

Number of # Register 0

Number of SFC Output Bits 0

Function Title

e. Click on the I/O Definition tab, then set up the input/output number and data type of the function.
Example: In the case of UFC FUNC-T1 MW00030 ML00032,DB000001, the setting is as follows.



[FUNC-T1] DWGConfiguration DEMO XY-TABLE MP930 Online Local

PT#: 1 UT#: 1 CPU#: 0

Number of Input 2

Number of Address Input 0

Number of Output 1

No	Input type	Input	Output type	Output
1	I-REG		B-VAL	
2	L-REG			

f. Close the DWG Configuration screen. Enter the user function ladder program in the ladder screen.

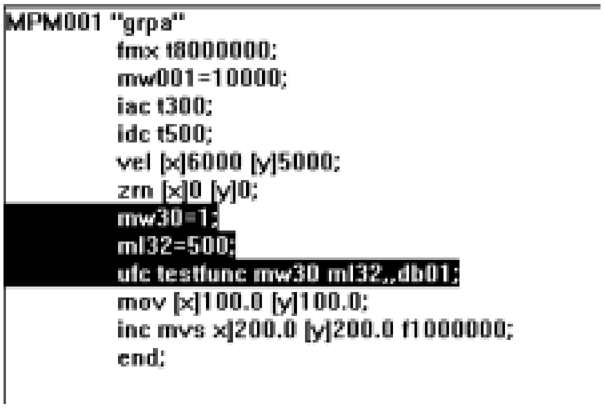
g. Select **Save (S)** from **File (F)** in the ladder menu.

h. The DWG/FUNC Save dialogue box appears. Click on the **Yes** button.

i. A dialogue box appears. It says “Save is complete.” Click on the **OK** button.

j. By completing the above steps, the user functions called from the motion program are created.

k. Create a program for calling up the user functions in the Motion Edit screen.



```
MPM001 "grpa"
        fmx t8000000;
        mw001=10000;
        iac t300;
        idc t500;
        vel [x]6000 [y]5000;
        zrn [x]0 [y]0;
        mw30=1;
        ml32=500;
        ufc testfunc mw30 ml32,,db01;
        mov [x]100.0 [y]100.0;
        inc mvs x]200.0 [y]200.0 f1000000;
        end;
```

l. Execute the motion program to verify motion.

3.1.5 I/O Variable Wait (IOW) Command

■ Outline

The I/O Variable Waiting (IOW) command waits until the conditional expression reaches its designated condition, then proceeds to the next block.

■ Detailed Explanation

The IOW command method is shown as follows:

```
IOW     IB000001&IB000002 = = |;
                  A            B

 A: Conditional expression
 B: Condition
```

According to the above command, execution waits until the conditional expression is formed. When this block is complete with a valid condition, it proceeds to the next block.

- The conditional expression can only be located on the left side.
- If it is bit-type, only the comparison command "= =" can be used. It cannot be used in the case shown below.

```
IOW     IB000001&IB000002 <>0;
```

- The ( ) can be used in the conditional expression.
- All comparison commands can be used in conditional expressions other than that of bit-type.

```
IOW     MW00100+MW00101>1000;

IOW     ML00100*3/100>20;

IOW     MW00100&MW00101^MW00102= =3355H;
```

■ Program Examples

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

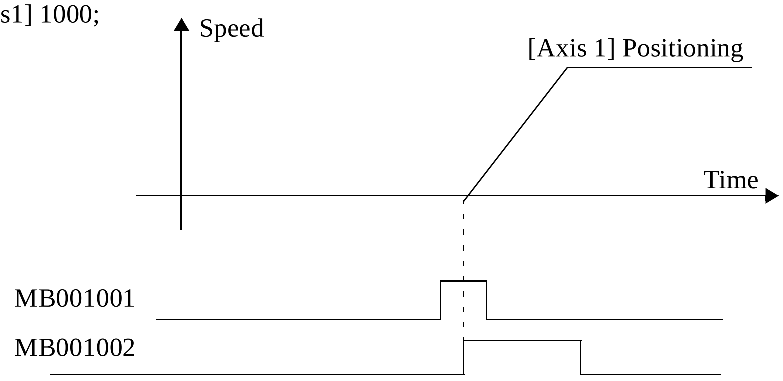


Figure 3.9: Program Example of the I/O Variable Wait (IOW)

3.1.6 Sub-program Call-out (MSEE) Command

■ Outline

The Sub-program Call-out (MSEE) command can call out a sub-program memorized in the motion program and execute it from the motion program.

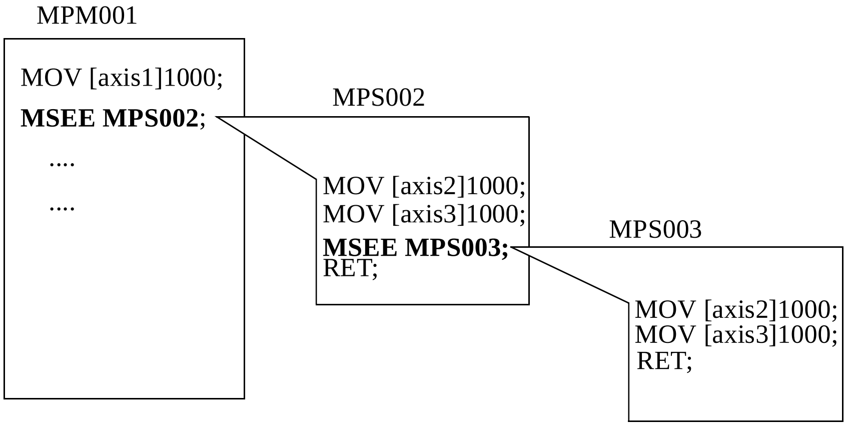
■ Detailed Explanation

The MSEE command method is shown as follows.

```
MSEE    MPS—;
            Number of the sub-program called out
```

According to the above command, the sub-program with a number designated by the MPS is executed.

• There are no restrictions on calling multiplex sub-programs from another sub-program.



```
MOV [axis1]1000;
MSEE MPS002;
   ....
   ....
```

```
MOV [axis2]1000;
MOV [axis3]1000;
MSEE MPS003;
RET;
```

```
MOV [axis2]1000;
MOV [axis3]1000;
RET;
```

Figure 3.10: Subprogram

• At the completion of the sub-program, the Subprogram End (RET) command must be designated.

Important Points

• Sub-routine Restrictions

Note that there are some restrictions for the motion program description within the sub-routine.

- Up to 2 of the PFORK commands can be run in parallel.
- When the PFORK command is used within the sub-routine, the axis movement commands can only be described on one side.
- Up to 256 programs of the combined Main program MPM □□□ and Sub-program MPS □□□ can be used. However, the same number cannot be used for both the MPM and MPS.
- When a main program number is called out by the MSEE command, it is not executed.

3.1.7 Sub-program End (RET) Command

■ Outline

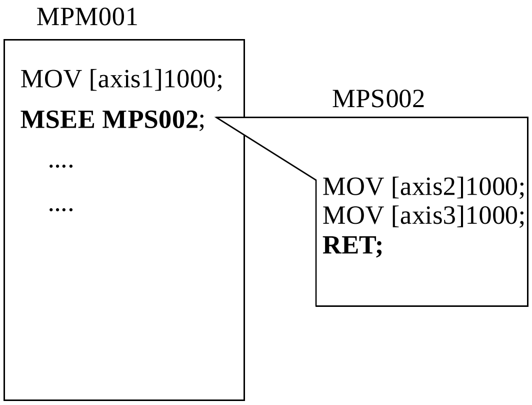
The Subprogram End (RET) command is designated at the end of the sub-program.

■ Detailed Explanation

The RET command method is shown as follows:

```
RET;
Sub-program End
```

According to the above command, the program proceeds to the block after the (MSEE) command of the program (main or sub-program) which called out this sub-program.



```
MOV [axis1]1000;
MSEE MPS002;
    ....
    ....
```

```
MOV [axis2]1000;
MOV [axis3]1000;
RET;
```

3.1.8 1-scan Wait (EOX) Command

■ Outline

Continuous sequence commands of the motion programs can be executed with 1 scan. The EOX command is used when the continuous sequence commands are executed with several scans. Moreover, this command also corresponds to the block operation. In other words, it is a command that can be stopped in the block operation mode.

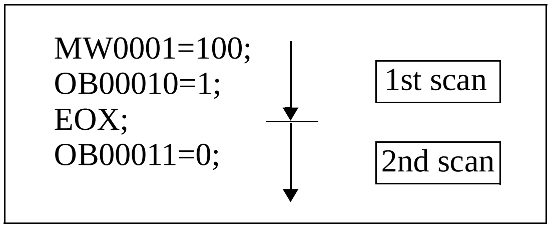
■ Command Methods

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

Blocks after the EOX command are executed in the next scan.

■ Program Example

In the case shown above:

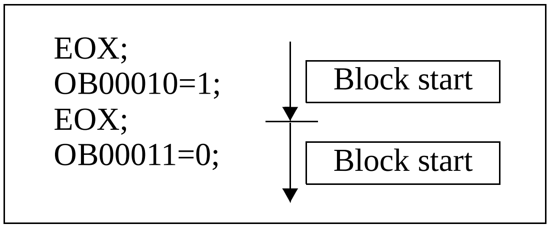


```
MW0001=100;
OB00010=1;
EOX;
OB00011=0;
```

When the WHILE command is used:

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

Debugging example of the sequence command:



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

Supplement

- Block Operation

  Block Operation verifies that the motion program is running normally after the program is started with a block unit. The block unit operation can be executed according to the block operation mode and block operation start signals. However, in the block operation, some commands can be stopped and some cannot be stopped.

  - Commands that can be stopped: The commands whose processes cannot be completed by 1 scan (such as the axis movement commands: MOV, MVS, etc., including the EOX).
  - Commands that cannot be stopped: The commands whose processes can be completed by 1 scan (such as the sequence commands: calculation command, etc., including the PLD).

3.1.9 Branch (IF ELSE IEND) Command

■ Outline

The Branch (IF ELSE IEND) command executes the block between [IF~ELSE] if a conditional expression is valid; and the block between [ELSE~IEND] if a conditional expression is invalid.

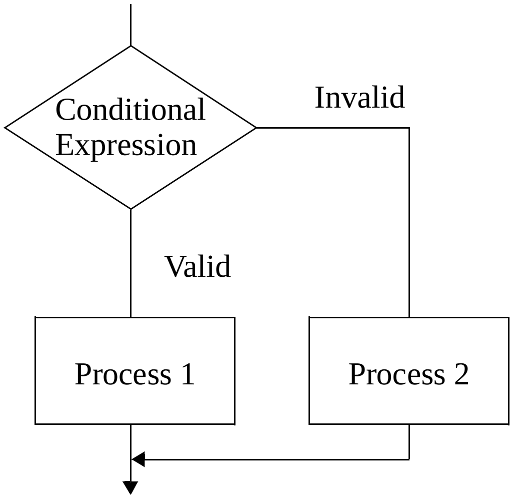
■ Detailed Explanation

The Branch (IF ELSE IEND) command method is shown as follows:

```
IF (Conditional expression)
    ... (Process 1)
ELSE;
    ... (Process 2)
IEND;
```

According to the above command, execute Process 1 if the conditional expression is valid; and Process 2 if the conditional expression is invalid.

- The ELSE can be omitted. At this point, if the conditional expression is invalid, the execution is continued from the next block followed by the IEND.



Important Point

The Branch (IF ELSE IEND) command can be nested up to 8 levels.

■ Program Example

```
IF MB– = =1;
MOV [axis1]10000; ← when MB— is ON, position the axis1.
ELSE;
MOV [axis2]10000; ← when MB— is OFF, position the axis2.
IEND;
```

3.1.10 Repeat (WHILE WEND) Command

■ Outline

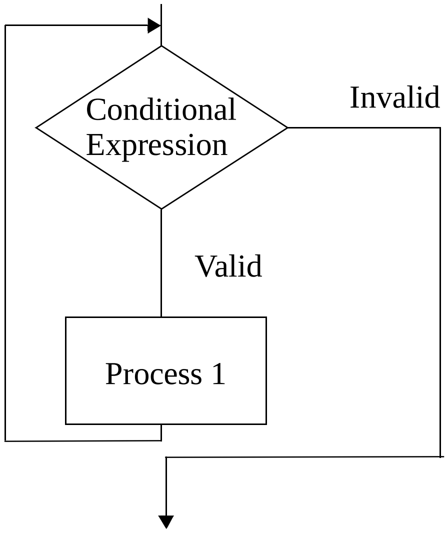
The Repeat (WHILE WEND) command repeats to execute the block within a designated range, when the conditional expression is being formed.

■ Detailed Explanation

The Repeat command method is shown as follows:

```
WHILE (Conditional expression);
...;
...; (Process)
...;
WEND; ← Repeat command ends.
```

According to the above command, the WHILE~WEND block is repeated when the conditional expression is being formed. If the conditional expression becomes invalid, it jumps to the block after WEND.



Important Points

(1) The WHILE WEND command can be nested up to 8 levels.

(2) If the repeated program section is created using only commands for which processing is completed in one scan, an infinite loop is created and a watchdog timer error (system error) is generated. Therefore, when using the 1-scan command, the EOX command (1-scan Wait command) must be set.

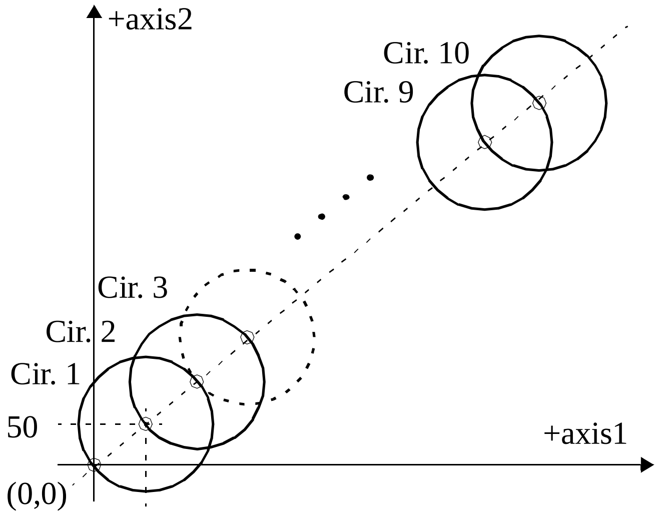
(3) Commands that cannot be ended by the 1-scan command:
MOV, MOS, MCW/MCC, ZRN, SKP, MVT, EXM, ACC, SCC, PFN, VEL, INP, TIM, IOW

(4) Commands that can be ended by the 1-scan command:
ABS, IWC, IFP, PLN, IAC, IDC, FMX, POS, PLD, MSEE, END, RET, IF, WHILE, PFORK, SFORK, and all of the sequence commands.

■ Program Example

In the following program example, ten circular arcs (each with a radius of 50) are drawn.

```
MOV [axis1]0 [axis2]0;                 ← Positioning
MW0001=1;                              ←Counter Preset
INC;                                   ← Incremental mode designation
PLN [axis1] [axis2];                   ← Coordinate plane designation
WHILE MW0001<=10;                      ← Repeat Command
MCW [axis1]0 [axis2]0 U50. V50. F8000; ← Circular Interpolation
MOV [axis1]50. [axis2]50.;             ← Positioning
MW0001=MW0001+1;                       ← Counter Preset
WEND;                                  ← Repeat Command End
```



3.1.11 Parallel Execution (PFORK, JOINTO, PJOINT) Command

■ Outline

The Parallel Execution (PFORK) command executes the block of designated labels in parallel. After each process is executed in parallel, they are merged into the label designated by the JOINTTO command. Up to 4 parallel executions can be designated.

■ Detailed Explanation

The Parallel Execution command method is shown as follows:

```
PFORK       Label1    Label2    Label3....
Lable1:     Process1
            JOINTO    LabelX
Label2:     Process2
            JOINTO    LabelX
Label3:     Process3
            JOINTO    LabelX
            •
            •
LabelX:     PJOINT
```

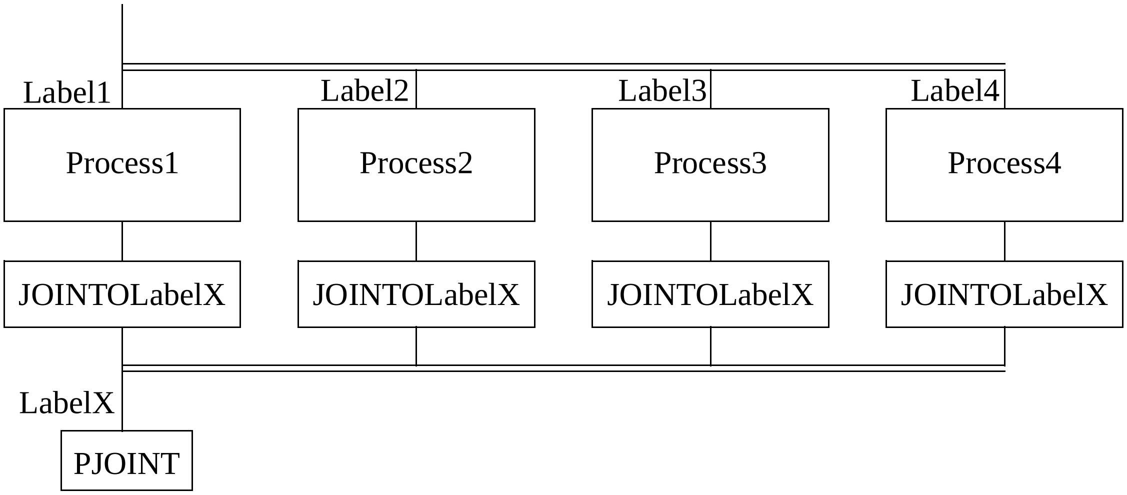


Figure 3.11: Parallel Execution Command (PFORK, JOINTO, PJOINT) Method

According to the above command, the blocks (Process1, 2, 3,....) of the PFORK command-designated label are executed. After each process is executed in parallel, they are merged into the label designated by the JOINTO command. Up to 4 parallel executions can be designated.
According to the command, parallel execution of the axis movement and sequence commands, or parallel execution among the axis movement commands can also be designated freely.

Commands designated before the PFORK command

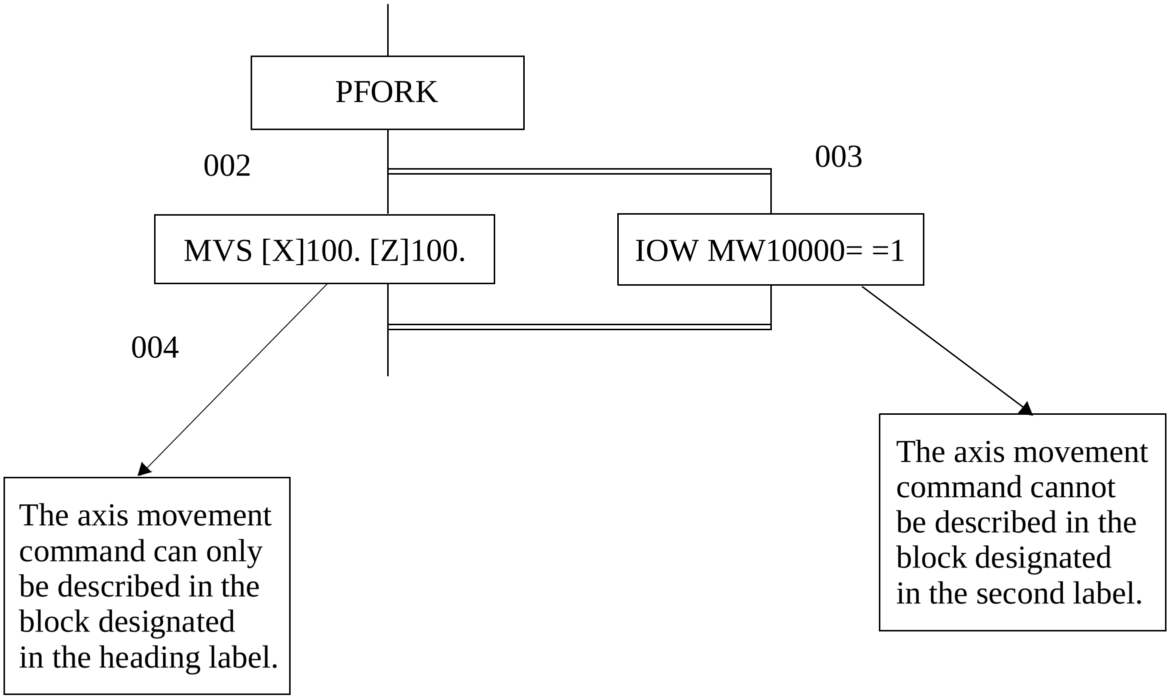
The values of the commands (FMX, ABS/INC, F reference, IFP, PLN, IAC/IDC...) designated before the PFORK command are converted to the processes executed by the Parallel Execution command. Moreover, these commands can be designated to each parallel. After merge, the value of the process at the far left is converted.

The Parallel Execution command within sub-routine

The Parallel Execution command within a subroutine has the following restrictions:

- Up to 2 parallel executions can be designated in a subroutine.
- The axis movement commands can only be described in the block designated in the heading label.

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



The axis movement command can only be described in the block designated in the heading label.

The axis movement command cannot be described in the block designated in the second label.

Figure 3.12: Parallel Execution Command Within Subroutine

Important Points

(1) When duplicate labels exist, an error: “The label is duplicated” occurs.

(2) An error occurs when a branch point of the PFORK command is different from the number of the label.

■ Program Example

Basic Pattern

A program example of the basic pattern is shown as follows:

```
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.;
```

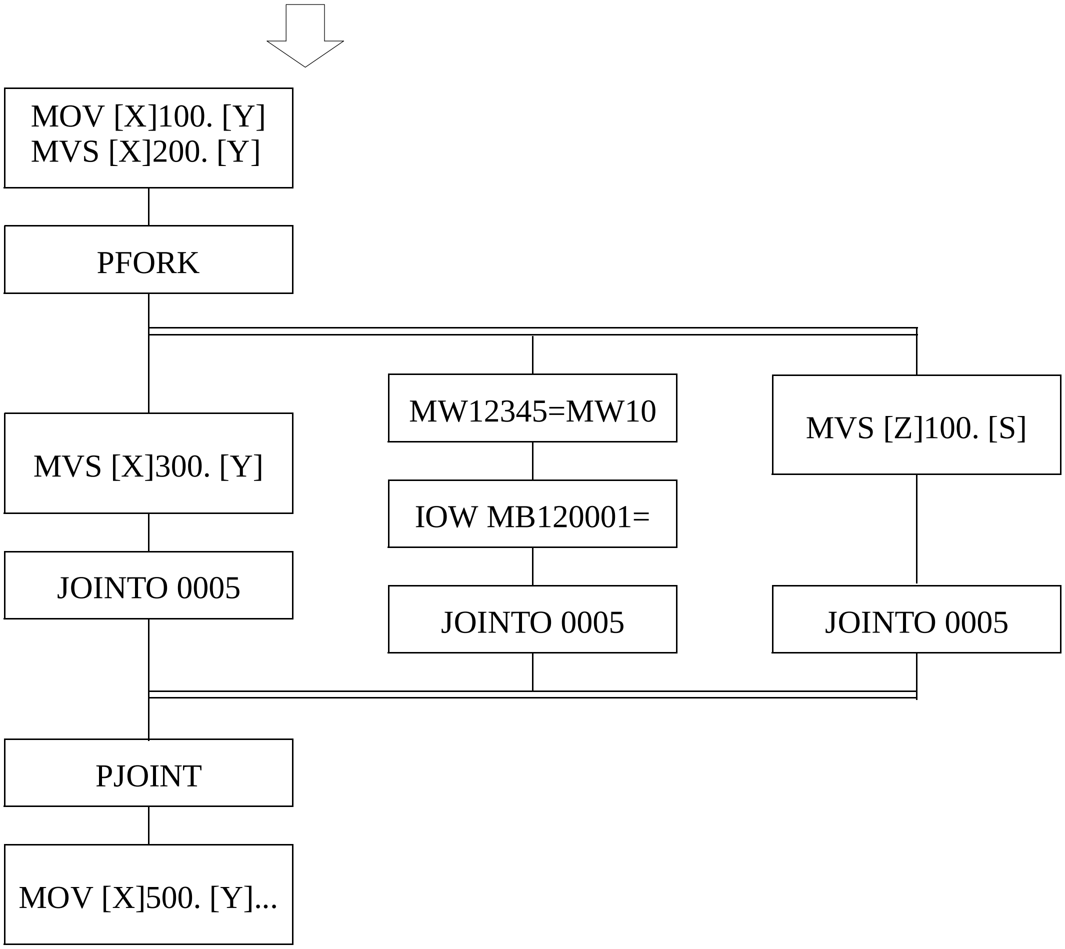
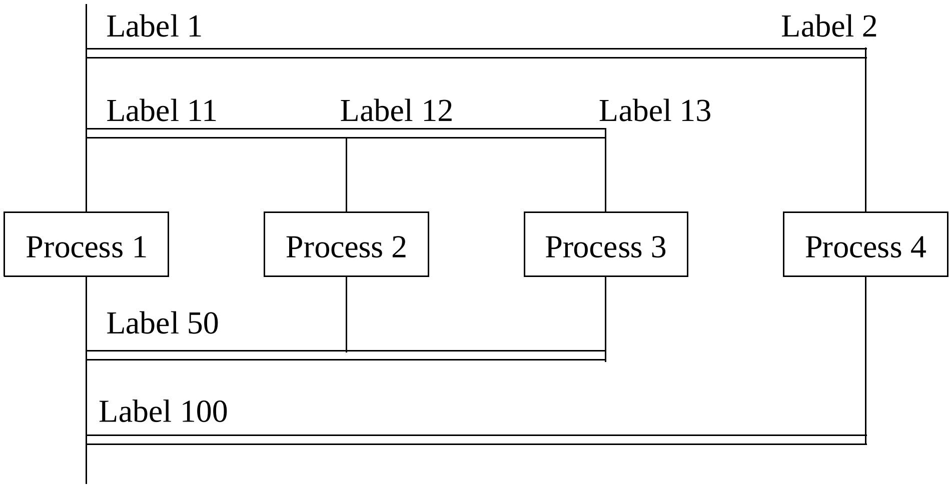


Figure 3.13: Program Example of the Parallel Execution (PFORK, JOINTO, PJOINT) Command

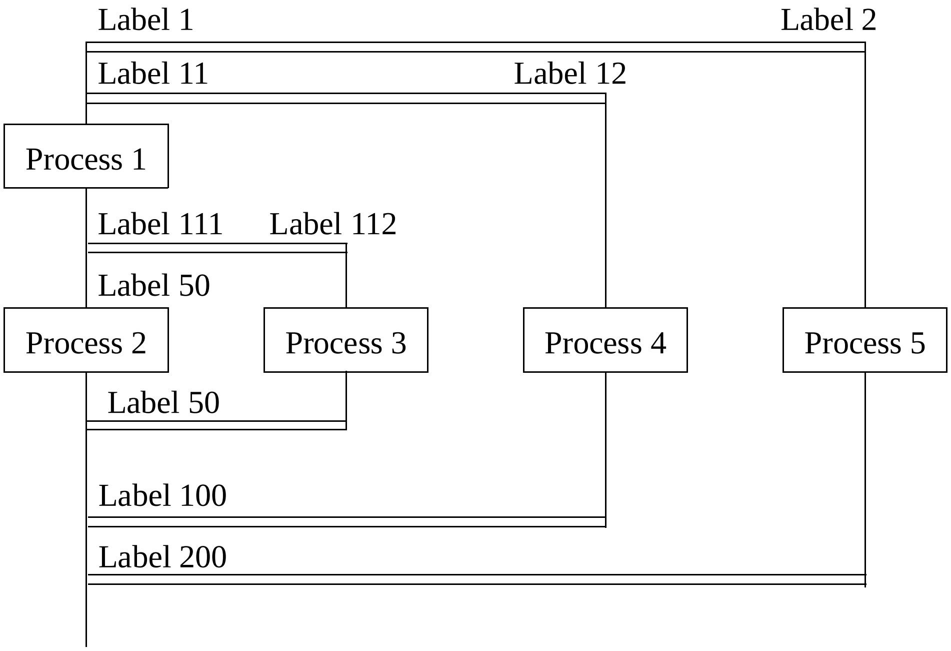
Application Pattern 1

It is possible to execute the following parallel execution.



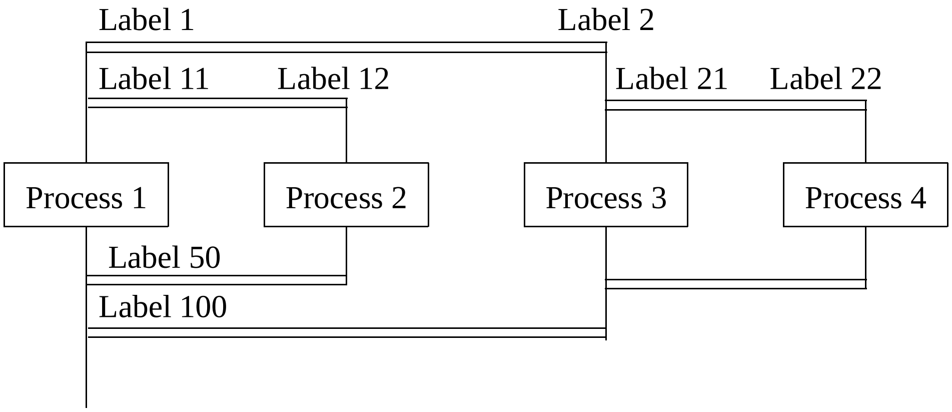
Application Pattern 2

It is possible to execute the following parallel execution.



Important Points

It is not possible to execute the following parallel execution.



3.1.12 Selective Execution (SFORK, JOINTO, SJOINT) Command

■ Outline

When the conditional expression is valid, a label block followed by a "?" mark is executed. After each process is executed, it is merged into the label block designated by the JOINTO command. Up to 4 conditional expressions can be designated.

■ Detailed Explanation

The Selection Execution command (SFORK) method is shown as follows:

```
SFORK      Conditional expression 1? Label 1, Conditional expression 2? Label 2,
           Condition 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:   PJOINT
```

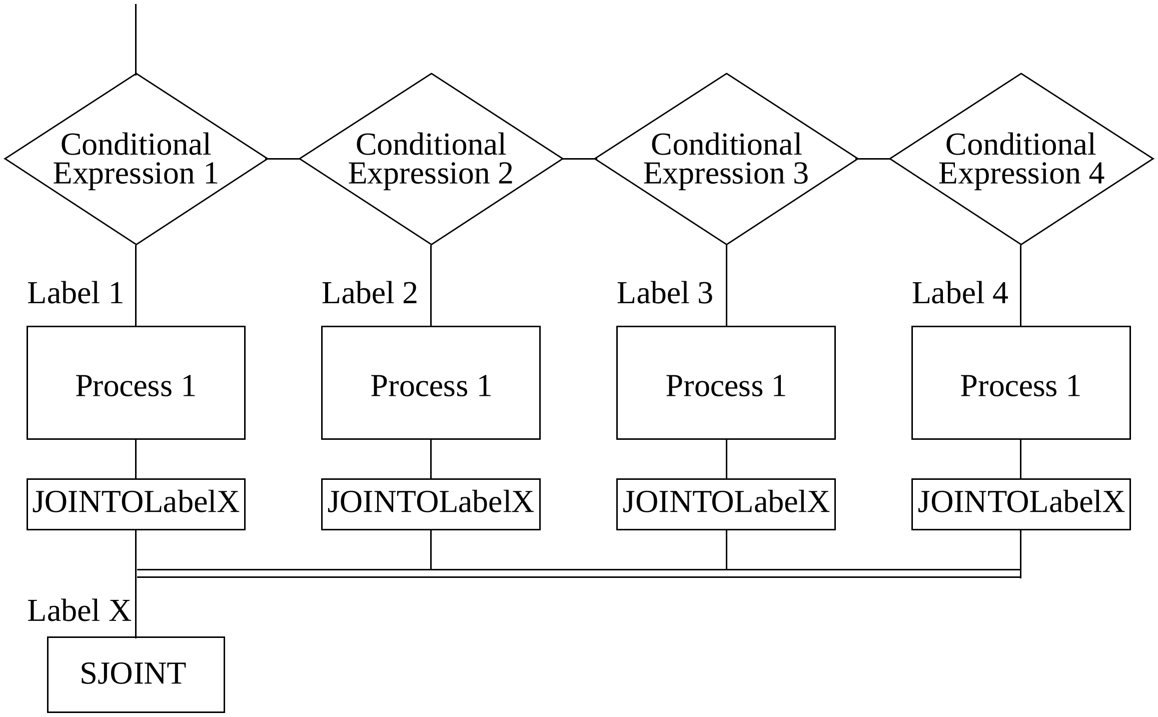


Figure 3.14: Method of Designating Parallel Execution Commands (SFORK, JOINTO, SJOINT)

According to the above command, a label block is executed if SFORK command-designated conditional expressions are valid. After each process is executed, merge them into the labels designated by the JOINTO command. Up to 4 Selective Execution processes can be designated.

- The conditional expressions are verified by starting from the Conditional expression 1. Therefore, even though multiple conditional expressions exist, the label process in which the conditional expression is first validated is executed.
- In the conditional expression, the condition must be described. If the condition is invalid, the execution waits at the SFORK command block until it is validated.

■ Program Example

```
0001: MOV [X]100. [Y]150.:
      MVS [X]200. [Y]250. F1000:
      SFORK MW00100=1? 002, MW00100=2? 003, MW00100=3? 004;
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.;
```

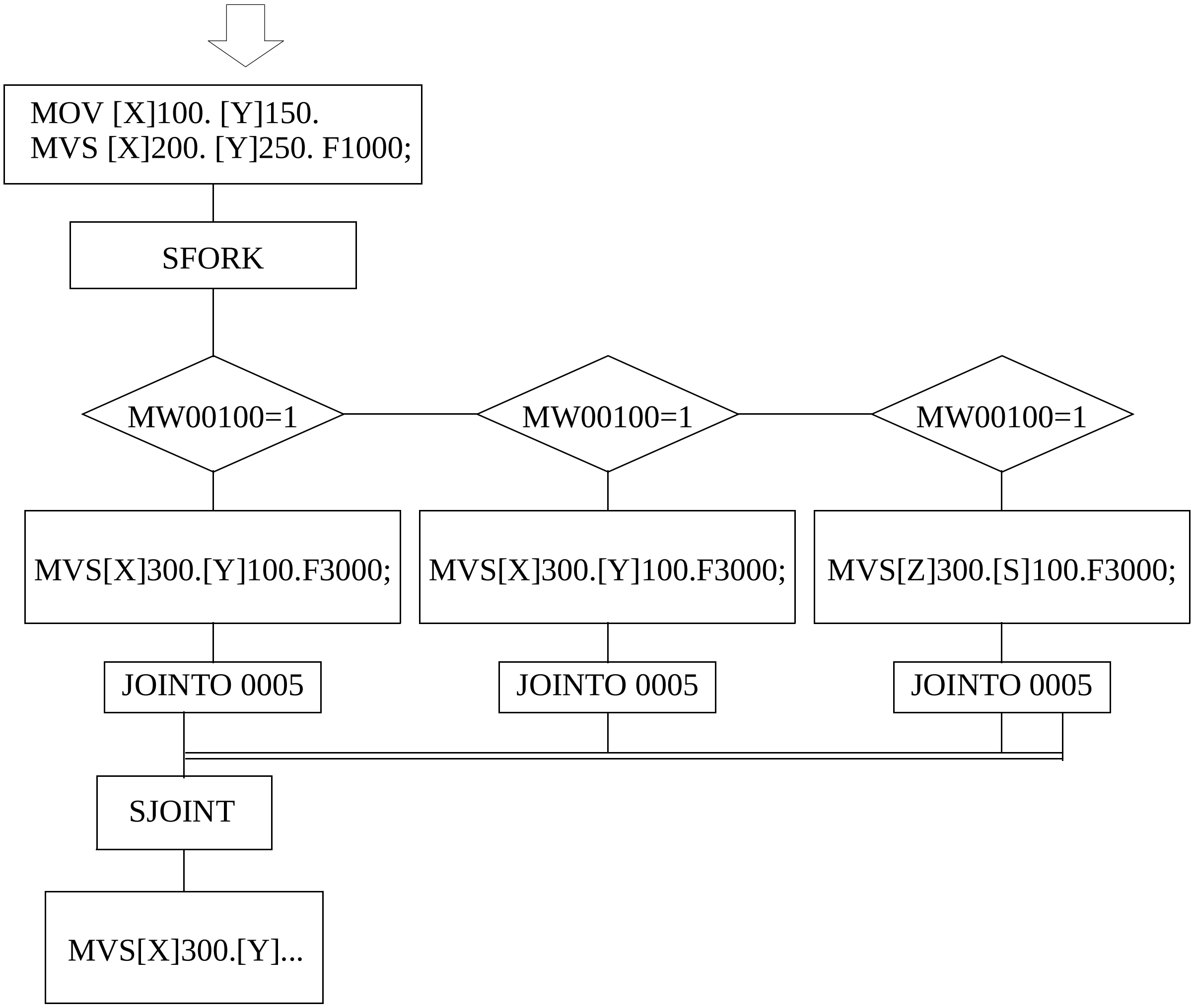


Figure 3.15: Program Example of the Selective Execution Commands (SFORK, JOINTO, SJOINT)

3.2 Speed/Acceleration Commands

This section explains how to program the commands that set the speed and acceleration for the axis movement commands.

3.2.1 Acceleration Time Change (ACC) Command

■ Outline

This command changes the acceleration time for each axis of positioning-related commands (MOV, MVT, EXM). It sets up time for reaching the feed speed set by the VEL command.

■ Detailed Explanation

The ACC command method is shown as follows:

$$\text{ACC}\quad \underbrace{\text{[axis1]—[axis2]—...;}}_{\text{Acceleration Time}}$$

According to the above command, the acceleration time for each axis is changed. But that of the undesignated axis is not changed.
The acceleration time of the MOV (Positioning) command is changed as follows.

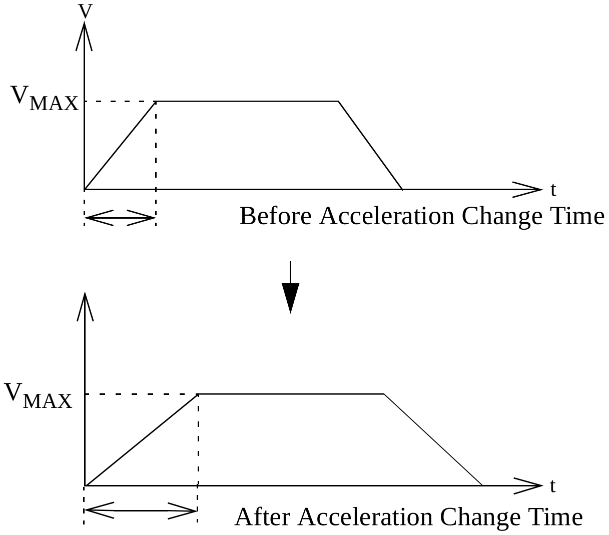


Figure 3.16: Acceleration Time Change

- When the ACC command is not sent, acceleration time is determined by setting parameter OWxx0C (Linear Acceleration Time Setting). Deceleration time is automatically set to be the same as the acceleration time.
- The acceleration time changed by the ACC command is valid until being reset by the next ACC command.
- A designating range of the ACC command is as indicated below.

1 ~ 32767[msec]

Supplement

According to the ACC command, the setting parameter OWxx0C (Linear Acceleration Time Setting) is changed. It is possible to program “OWxx0C=xxxx” instead of the ACC command.

■ Program Examples

```
INC;
VEL [axis1]6000;        Feed Speed Setting
ACC [axis1]1000;        Acceleration Time 1sec
MOV [axis1]100000;      Positioning
ACC [axis1]500;         Acceleration Time 0.5sec
MOV [axis1]50000
```

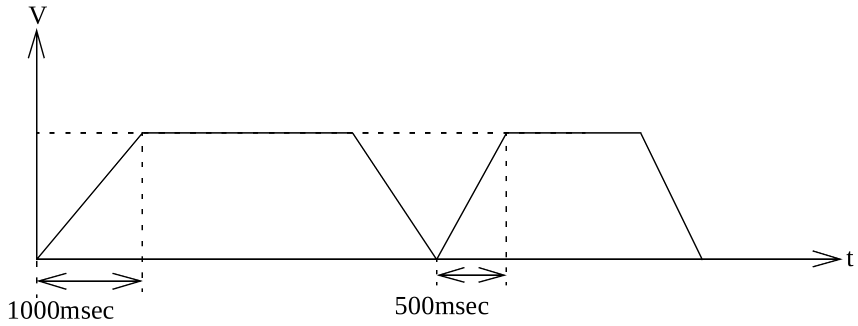


Figure 3.17: Program Example of the Acceleration Time Change

3.2.2 S-Curve Time Constant Change (SCC) Command

■ Outline

The S-curve Time Constant Change (SCC) command sets up the S-curve accel/decel function parameter that suppresses machine vibration during accel/decel.

■ Detailed Explanation

The SCC command method is shown as follows:

```
SCC   [axis1]—[axis2]—...;
     ______________________
     S-curve Time Constant
```

According to the above command, the S-curve time constant of each axis is changed; however, that of the undesignated axis is not changed.

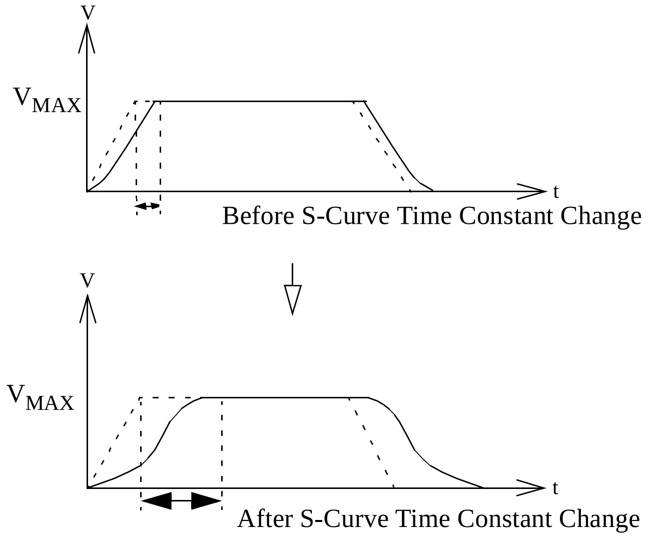

V
VMAX
t
Before S-Curve Time Constant Change
V
VMAX
t
After S-Curve Time Constant Change

Figure 3.18: S-Curve Time Constant (SCC) Change

- When the SCC command is not sent, the S-curve time constant is determined by the S-curve Time Constant setting parameter.
- The S-curve time constant changed by the SCC command is valid until being reset by the next SCC command.
- A designating range of the SCC command is as indicated below.

$$\boxed{0 \sim 510[\text{msec}]}$$

■ Program Examples

```
INC;
SCC [axis1]1000;
MOV [axis1]100000;
SCC [axis1]500;
MOV[axis1]100000
```

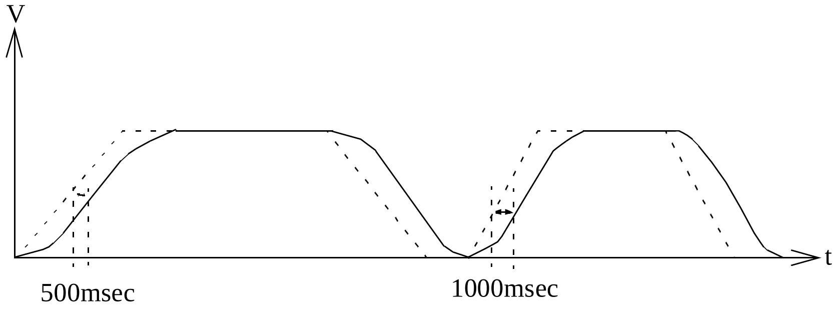


Figure 3.19: Program Example of the S-Curve Time Constant (SCC)

3.2.3 Feed Speed Change (VEL) Command

■ Outline

This command changes the feed speed for each axis of positioning-related commands (MOV, MVT, EXM).

■ Detailed Explanation

The VEL command method is shown as follows:

$$\text{VEL}\quad \underline{\text{[axis1]—[axis2]—...;}}$$
$$\text{Feed Speed}$$

According to the above command, the feed speed of each axis is changed. But that of the undesignated axis is not changed.

- When the VEL command is not sent, the feed speed is determined by the setting parameter [Feed Speed of Each Axis] and the setting data of the previous VEL command.
- The speed changed by the VEL command is valid until being reset by the next VEL command, or when an MVT command is sent.
- The VEL command cannot be used within the same block as the MOV command.
- A designating range of the VEL commands is as indicated below.

0 ~ 2^{31}-1[10^{n}command units/min]
n=number of decimal places

Supplement

According to the VEL command, the setting parameter OLxx22 Fast Feed Speed of each axis is changed. Therefore, it is possible to program “OLxx22=xxxx” instead of the VEL command.

■ Program Examples

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

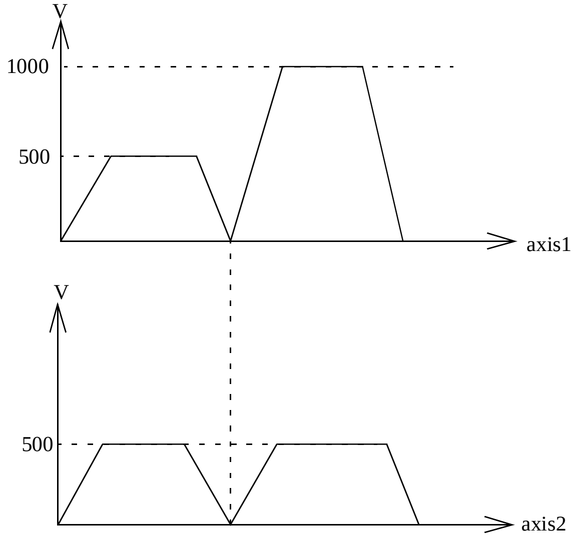


Figure 3.20: Program Example of the Feed Speed Change (VEL)

3.2.4 Interpolation Feed Speed Ratio Setting (IFP) Command

■ Outline

The Interpolation Feed Speed Ratio Setting (IFP) command can set the interpolation feed speed by designating the percentage of the Maximum Interpolation Feed Speed (FMX).

■ Detailed Explanation

The IFP command method is shown as follows:

```
IFP P--;
  Speed Designation Value (Percentage)
```

According to the above command, the value multiplied by the percentage which is designated by the Maximum Interpolation Feed Speed (FMX) is set as the interpolation feed speed.

- The IFP command cannot be used within the same block as the Interpolation command.
- When designating an interpolation feed speed with the IFP command (instead of using the F command), designate the Maximum Interpolation Feed Speed with the FMX command first.
- When the IFP command is sent after the F command designation, the interpolation feed speed set with the F command is canceled.
- The speed set by the IFP command is valid until being reset by the next IFP or F command.
- A designating range of the IFP command is as indicated below:

```
1 ~ 100 (%)
```

Important Point

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 the F and IFP commands are sent without designating the FMX, an error occurs.

■ Program Examples

```
INC;
FMX T1000;
IFP P50
MVS [axis1] 1000  [axis2] 1000;
IFP P100
MVS [axis 1] 1000 [axis 2] 1000;
```

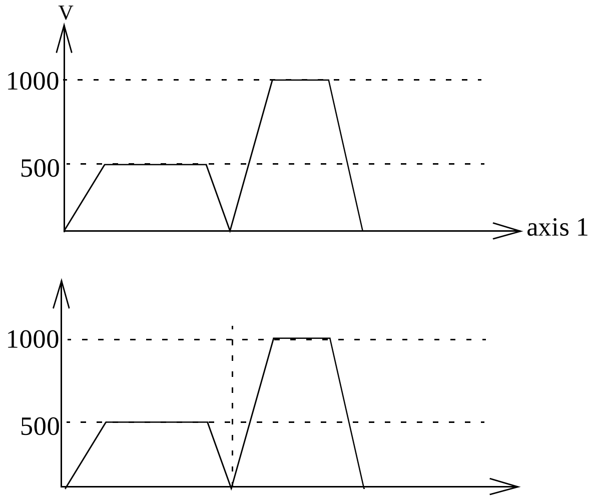


Figure 3.21: Program Example of Interpolation Feed Speed Ratio Setting (IFP)

3.2.5 Maximum Interpolation Feed Speed Setting (FMX) Command

■ Outline

The Maximum Interpolation Feed Speed Setting (FMX) command sets up the maximum speed for the execution of the interpolation feed commands (MVS, MCC/MCW, SKP).

■ Detailed Explanation

The FMX command method is shown as follows:

```
FMX T--;
  Maximum Interpolation Feed Speed
```

According to the above command, the Maximum Interpolation Feed Speed is set. When creating a motion program that uses the interpolation commands, always designate it at the beginning of the program. After being set, the speed is valid until being reset.

- The interpolation acceleration time set according to the IAC and IDC commands is the acceleration time up to the FMX designation, as well as the deceleration time from the FMX designation.

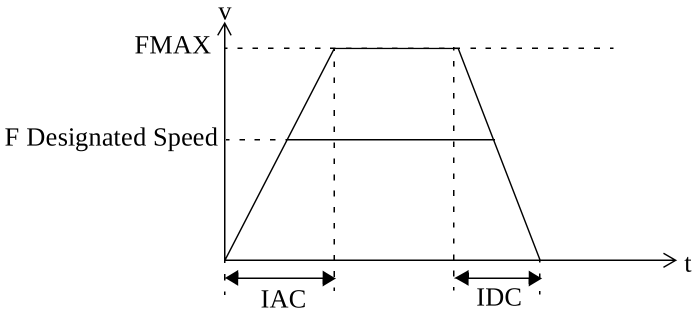


Figure 3.22: Maximum Interpolation Feed Speed

- The Interpolation Feed Speed Ratio Setting (IFP) command sets up the interpolation feed speed value. This value is multiplied by the percentage designated by the Maximum Interpolation Feed Speed (FMX).
- A designating range of the FMX commands is as indicated below:

1 ~ 2^{31}-1 [Command units/min]

■ Program Examples

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

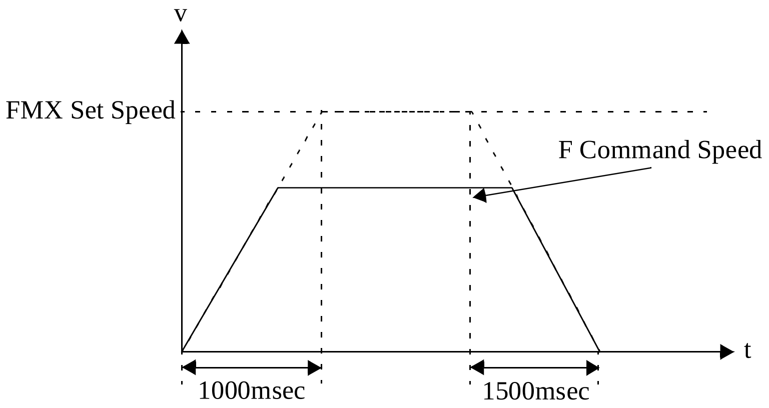


Figure 3.23: Program Example of the Maximum Interpolation Feed Speed Setting

Important Points

- An alarm occurs if the FMX is not designated in a motion program that uses the interpolation command.
- A decimal point cannot be used with the FMX designation. Add as many “0” as the decimal places when setting.

  Example: Decimal places=3
  Command unit=mm
  When setting the maximum interpolation feed speed as 50000.000mm..., program as follows:

  ```
  FMX     T50000000;
  ```

3.2.6 Interpolation Acceleration Time Change (IAC) Command

■ Outline

The Interpolation Acceleration Time Change (IAC) command sets up the acceleration time for the execution of the interpolation feed commands (MVS, MCC/MCW, SKP).

■ Detailed Explanation

The IAC command method is shown as follows:

```
IAC   T ; —
  Acceleration Time
```

The interpolation acceleration time designated by the IDC command equals the amount of time required to reach the maximum interpolation feed speed designated by the FMX command.

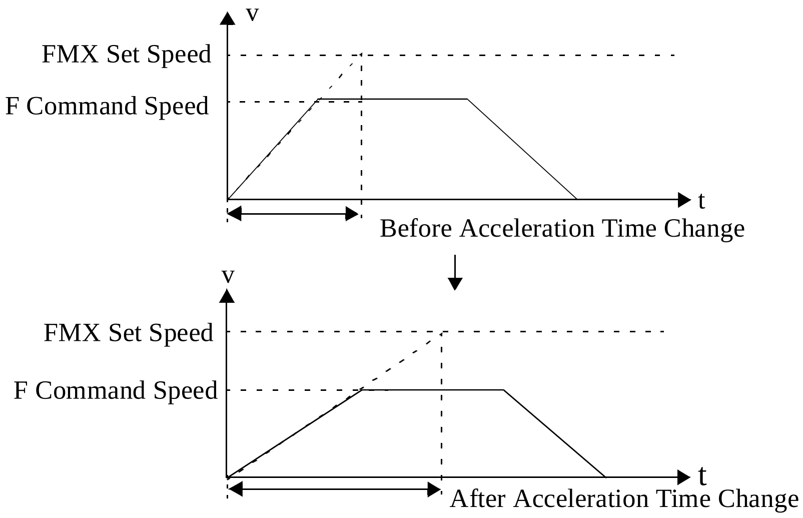


Figure 3.24: Interpolation Acceleration Time Change

- When the IAC command is not executed, the acceleration time = 0 (rectangle).
- The acceleration time changed by the IAC command is valid until being reset by the next IAC command.
- A designating range of the IAC commands is as indicated below

```
0 ~ 32767 [msec]
```

■ Program Examples

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

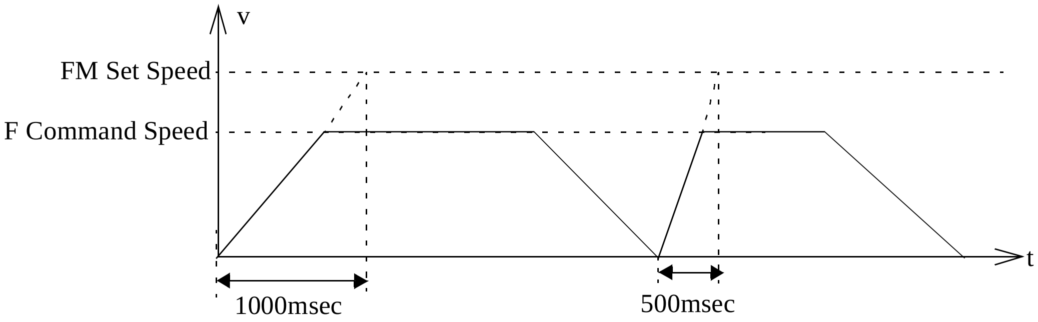


Figure 3.25: Program Example of the Interpolation Acceleration Time Change

Important Point

When designating the IAC command, set up the maximum interpolation feed speed in advance with the FMX command, to avoid an error.

3.2.7 Interpolation Deceleration Time Change (IDC) Command

■ Outline

The Interpolation Deceleration Time Change (IDC) command sets up the deceleration time for when interpolation feed commands (MVS, MCC/MCW, SKP) are executed. It can be set freely with the motion program.

■ Detailed Explanation

The IDC command method is shown as follows:

```
IDC T--;
  Deceleration Time
```

The interpolation deceleration time is the time between the speed set up by the FMX command, until stop.

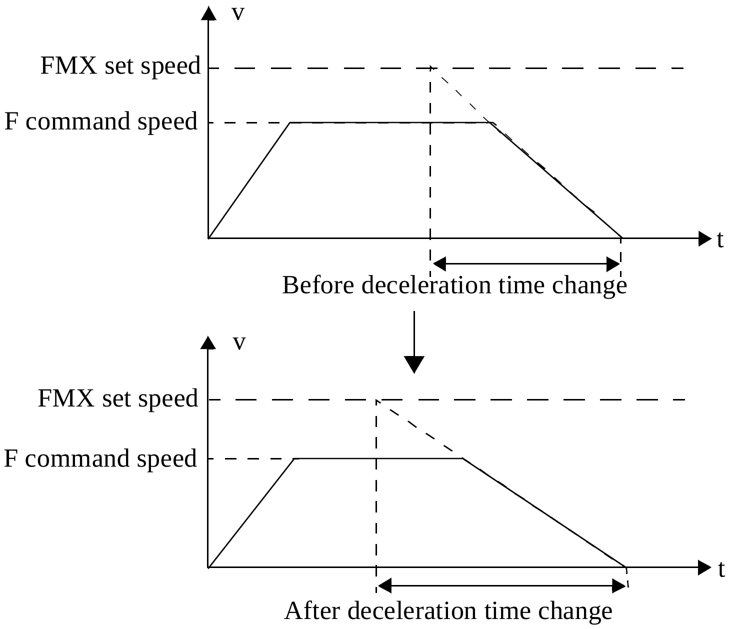

v
FMX set speed
F command speed
t
Before deceleration time change
v
FMX set speed
F command speed
t
After deceleration time change

Figure 3.26 Interpolation Deceleration Time Change (IDC)

- When the IDC command is not executed, the deceleration time constant = 0 (rectangle)
- The deceleration time changed according to the IDC command is valid until being reset by the next IDC command.
- A designating range of the IDC command is as indicated below:

0 ~ 32767 msec

■ Program Examples

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

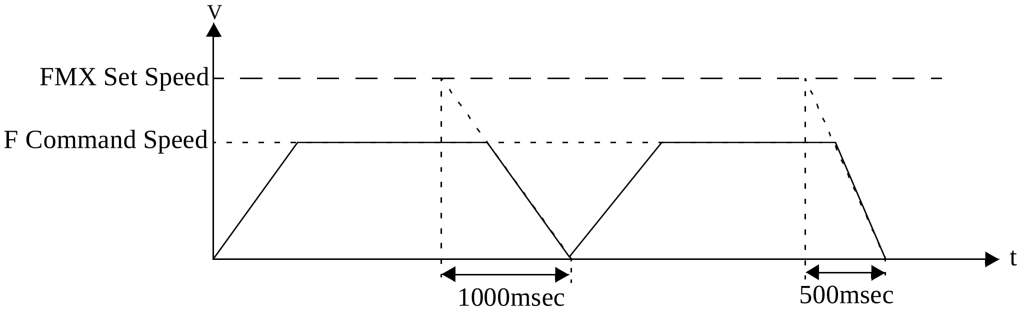


Figure 3.27: Program Example of the Interpolation Deceleration Speed Time Change (IDC)

Important Points

When designating the IDC command, set up the maximum interpolation feed speed in advance with the FMX command. Otherwise, an error occurs.

4 Sequence Commands

This chapter explains how to program the sequence commands including the calculation commands, branch commands, repeat commands, etc.

4.1 Sequence Command Outline

This section includes an outline of the sequence commands as well as a combination of the calculations.

4.1.1 Calculation Commands

The calculation commands execute general calculations which combine global variables, local variables, and constants with operators and functions. Variables can be substituted for the results.

The basic format of the calculation is: Variable=<Calculation format>. It uses the following calculations and functions.

Type	Command	Name	Command Format
Arithmetic Calculations	=	Substitution	MW– =MW–;
	+	Addition	MW– =MW– +MW–;
	–	Subtraction	MW– =MW– –MW
	*	Multiplication	MW– =MW–*MW–;
	/	Division	MW– =MW–/MW–;
	MOD	Remainder	MW– =MOD;
Logical Calculations	|	OR (Logical OR)	MB– =MB– | MB–;
	^	XOR (Exclusive OR)	MB– =MB– ^MB–
	&	AND (Logical AND)	MB– =MB– &MB–;
	!	NOT (invert)	MB– =MB– !MB–;
Value Comparisons	= =	Same	IF MW– = =MW–;
	< >	Not same	IF MW– < >MW–;
	>	Greater	IF MW– >MW–;
	<	Less	IF MW– <MW–;
	>=	Greater or equal	IF MW– > =MW–;
	<=	Less or equal	IF MW– < =MW–;
Data Operations	SFR	Right-shift	SFR MB– N– W–;
	SFL	Left-shift	SFL MB– N– W–;
	BLK	Block transfer	BLK MW– MW– W–;
	CLR	Clear	CLR MB– W–;

Type	Command	Name	Command Format
Basic Functions	SIN	Sine	SIN (MW–);
	COS	Cosine	COS (MW–);
	TAN	Tangent	TAN (MF–);
	ASN	ARC sine	ASN (MF–);
	ACS	ARC cosine	ACS (MF–);
	ATN	ARC tangent	ATN (MW–);
	SQT	Square root	SQT (MW–);
	BIN	BCD→BIN	BIN (MW–);
	BCD	BIN→BCD	BCD (MW–);
	S{}	Designated bit ON	S{MB–}=MB– &MB–;
	R{}	Designated bit OFF	R{MB–}=MB– &MB–;

4.1.2 Arithmetic Calculation Combinations

Arithmetic calculation can be executed by combining the arithmetic calculation commands and the functions.

The order of calculations is as follows: functions, multiplication/division, addition/ subtraction.

An example of the calculation format is shown below:

```
Example:  MW00100=MW00102 + MW00104 * SIN(MW00106);
                         (3)       (2)    (1)
```

The calculation is executed in the above-numbered order.

Supplement

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

```
• MF00100=100. + 200. * 0.5 * SIN(30.);
    (150.)    (4)    (2)   (3)   (1)

• MF00100=(100. + 200.) * 0.5 * SIN(30.);
     (75.)     (2)     (3)   (4) (1)

• MF00100=(100. + 200. * 0.5) * SIN(30.);
    (100.)     (3)    (2)    (4) (1)
```

4.1.3 Logical Calculation Combinations

Logical calculations can be executed by combining the Logical Calculation commands.

There is no priority order for the calculations. It is also possible to combine the arithmetic calculations, but it is impossible to execute the real number calculation.

An example of the calculation format is shown below:

```
Example:   MW00100=MW00102 & MW00102 | MW00106 ^ MW00108;
                          (1)       (2)       (3)
```

The calculation is executed in the above-numbered order.

Supplement

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

```
• MW00100=MW00101 & MW00102 | MW00106 ^ MW00104;
                 (1)       (2)       (3)

• MW00100=MW00101 & MW00102 | (MW00106 ^ MW00104);
                 (2)       (3)        (1)
```

4.2 Arithmetic Calculations

In this section, the Arithmetic Calculation commands are explained.

4.2.1 Substitution (=)

■ Outline

Substitute the calculation result on the right side with the register on the left side. Priorities for the arithmetic calculations and the logical calculations vary. Refer to items 4.1.2 Arithmetic Calculation Combinations and 4.1.3 Logical Calculation Combinations for details.

■ Detailed Explanation

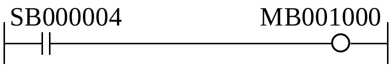
Command Method

```
(Results) = (Calculation Format);
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)
○	○	○	○

■ Program Examples

Type	Motion Program	Ladder Program
B	MB001000=1;	
W	MW00100=12345;	├─12345 ⇒MW00100
L	ML00100=1234567;	├─1234567 ⇒ML00100
F	MF00100=1.2345;	║├─1.2345 ⇒MF00100

4.2.2 Addition (+)

■ Outline

The addition function performs integer and real number addition on the right side, and stores the result in the register on the left side. For the right-side addition, a constant can also be used instead of the register. Even if the integer and real numbers are mixed, the result can still be stored by the data type on the left side.

■ Detailed Explanation

Command Method

```
MW– =MW– +MW–;
```

Data Types

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)
×	○	○	○

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	MW00101=MW00100+12345;	├─MW00100+12345⇒MW00101
L	ML00106=ML00102+ML00104;	├─ML00102+ML00104⇒ML00106
F	MF00202=MF00200+1.23456;	|├─MF00200+1.23456⇒MF00202

Important Point

When executing the variable calculation which has different data types, the results vary depending on the data type on the left side. Refer to item 5.1.1Variable Overview for details.

4.2.3 Subtraction (–)

■ Outline

The subtraction function performs integer and real number subtraction on the right side, and stores the result in the register on the left side. For the right-side subtraction, a constant can also be used instead of the register. Even if the integer and real numbers are mixed, the result can still be stored by the data type on the left side.

■ Detailed Explanation

Command Method

```
MW– =MW– – MW–;
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)
×	○	○	○

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	MW00101=MW00100–12345;	├─MW00100–12345⇒MW00101
L	ML00106=ML00102–ML00104;	├─ML00102–ML00104⇒ML00106
F	MF00202=MF00200–1.23456;	║├─MF00200–1.23456⇒MF00202

4.2.4 Multiplication (*)

■ Outline

The multiplication function performs integer and real number multiplication on the right side, and stores the result in the register on the left side. For the right-side multiplication, a constant can also be used instead of the register. Even if the integer and real numbers are mixed, the result can still be stored by the data type on the left side.

■ Detailed Explanation

Command Method

```
MW– =MW– * MW–;
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)
×	○	○	○

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	MW00102=MW00100*MW00101;	├─MW00100*MW00101⇒MW00101
L	ML00106=ML00102*ML00104;	├─ML00102*ML00104⇒ML00106
F	MF00202=MF00200*1.23456;	│├─MF00200*1.23456⇒MF00202

4.2.5 Division (/)

■ Outline

The division function performs integer and real number division on the right side, and stores the result in the register on the left side. For the right-side division, a constant can also be used instead of the register. Even if the integer and real numbers are mixed, the result can still be stored by the data type on the left side.

■ Detailed Explanation

Command Method

```
MW– =MW– / MW–;
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)
×	○	○	○

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	MW00102=MW00100/MW00101;	├─MW00100/MW00101⇒MW00101
L	ML00106=ML00102/ML00104;	├─ML00102/ML00104⇒ML00106
F	MF00202=MF00200/1.23456;	‖├─MF00200/1.23456⇒MF00202

4.2.6 Remainder (MOD)

■ Outline

When designating a block following the division command as a MOD, the remainder is stored in a designated variable. Even if the integer and real numbers are mixed in the block following the division command, the remainder is still stored by the data type on the left side.

■ Detailed Explanation

Command Method

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

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)
×	○	○	×

■ Program Example

Type	Motion Program	Ladder Program
B	—	—

W	MW00101=MW00100/3;
	MW00102=MOD;
		├──MW00100/3 ⇒MW00101
		MOD ⇒MW00102

L	ML00106=ML00102/ML00104;
	ML00108=MOD;
		├──MW00102/ML00104 ⇒MW00106
		MOD ⇒MW00108

F		

Example: 32-bit Integer Number

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

Important Point

The MOD command must be designated in the block following the division command.

4.3 Logical Calculations

In this section, the commands that execute logical calculations for the bit and integer number are explained.
There is no priority order for the calculations. The calculation is executed from the right, starting with item number 1. However, the items in parentheses are calculated first. It is possible to combine the arithmetic calculations; however, the real number calculation is impossible.

4.3.1 Logical OR (|)

■ Outline

Execute the logical OR of the immediately previous calculation result and a designated register, and leave the calculation result. The real number register cannot be used.

True Value Table of the Logical OR (OR:C=A|B)

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

■ Detailed Explanation

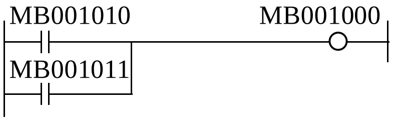
Command Method

```
MB001000=IB000100|SB000200;
MW00100=DW00101|0AAAAH;
ML00104=DL00106|IL00100;
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
○	○	○	×	○

■ Program Example

Type	Motion Program	Ladder Program
B	MB001000=MB001010|MB001011;	
W	MW00100=MW00101|MW00102;	├─MW00101^MW00102⇒MW00100
L	ML00106=ML00102|ML00104;	├─ML00102^ML00104⇒ML00106
F	—	—

4.3.2 Logical AND (&)

■ Outline

Execute logical AND of the immediately previous calculation result and a designated register, and leave the calculation result. The real number register cannot be used.

True Value Table of the Logical Multiplication (AND:C=A&B)

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

■ Detailed Explanation

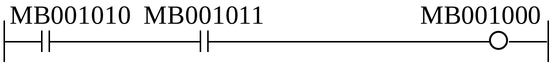
Command Method

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

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
○	○	○	×	○

■ Program Example

Type	Motion Program	Ladder Program
B	MB001000=MB001010&MB001011;	
W	MW00101=MW00100&00FFH;	├─MW00100^H00FF ⇒MW00101
L	ML00106=ML00102&ML00104;	├─ML00102^ML00104 ⇒ML00106
F	—	—

4.3.3 Exclusive OR (^)

■ Outline

Execute exclusive OR of the immediately previous calculation result and a designated register, and leave the calculation result. The real number register cannot be used.

True Value Table of the Exclusive OR (XOR:C=A^B)

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

■ Detailed Explanation

Command Method

```
MB00100=MW00101^MW00102;
ML00100=ML00102^12345678H;
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
○	○	○	×	○

■ Program Example

Type	Motion Program	Ladder Program
B		
W	MW00101=MW00100^00FFH;	├─MW00100⊕H00FF⇒MW00101
L	ML00106=ML00102^ML00104;	├─ML00102⊕ML00104⇒ML00106
F	—	—

4.3.4 NOT (!)

■ Outline

Invert a designated register data, and leave the calculation result. The real number register cannot be used.

■ Detailed Explanation

Command Method

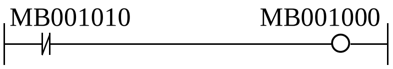
```
MB001000=!MW00101;
MW00100=!MW00101;
ML00100=!ML00102;
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
○*	○	○	×	∇

* It is impossible to designate a bit-type constant.

■ Program Example

Type	Motion Program	Ladder Program
B	MB001000=!MB001010;	
W	MW00100=!MW00101;	├─MW00101 COM ⇒MW00100
L	ML00100=!ML00102;	├─ML00102 COM ⇒ML00100
F	—	—

(Ex.) MW00100=!MW00101;

MW00101

0001	0010	0011	0100

(1234H)

⇩

MW00100

1110	1101	1100	1011

0EDCBH

4.4 Value Comparison

In this section, the Value Comparison commands which are used to distinguish conditional expressions are explained.

4.4.1 Value Comparison Command (= =, < >, >, <, >=, <=)

■ Outline

The Value Comparison commands are used to distinguish conditional expressions including the Branch (IF), Repeat (WHILE), I/O Wait (IOW) commands, etc.
The Value Comparison commands include the following 6 types.

Command	Meaning
= =	Equal
< >	Not equal
>	Greater
<	Less
>=	Greater or equal
<=	Less or equal

■ Detailed Explanation

Command Method

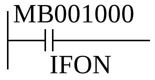
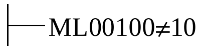
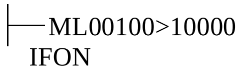
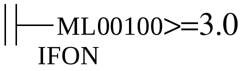
```
IF MB00100= =MW00102+MW00104;
WHILE MB0010001< >1;
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)
○*	○	○	○

In a bit-type conditional expression, only the “= =” command is enabled.

■ Program Example

Type	Motion Program	Ladder Program
B	IF MB00100= =1;	
W	IF MW00100< >10;	
L	IF ML00100>10000;	
F	IF MF00100>=3.0;	

Important Point

In ladder program command operation, store comparison results in the bit register, and then distinguish these results with the IFON command.

4.5 Data Operations

In this section, the data commands that execute data shift, movement, and initialization are explained.

4.5.1 Bit Right-shift (SFR) Command

■ Outline

In a bit string designated by heading bit number and bit width, the Bit Right-shift (SFR) command shifts as many numbers as designated towards the right.

■ Detailed Explanation

Command Method

```
SFR   MB001000   N5   W10;
         A       B     C

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

Data Type

	Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
Heading Bit Number	○	×	×	×	×
Number to be Shifted	×	○	×	×	○
Bit Width	×	○	×	×	○

○ = Yes × = No

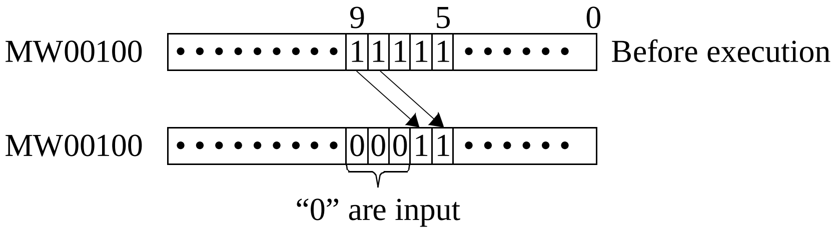
■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	SFR MB001000 N=5 W=10;	SHFTR MB001000 N=5 W=10
L	—	—
F	—	—

Example: Heading: MB001005 (bit 5 of MW00100)/bit width: 5

Shift it 3 bits towards the right.

```
SFR    MB00100A    N3    W5;
```



Supplement

With the SFR command, when a number to be shifted >= bit width, all data with a designated bit width becomes 0.

4.5.2 Bit Left-shift (SFL) Command

■ Outline

In a bit string designated by heading bit number and bit width, the Bit Right-shift (SFR) command shifts as many numbers as designated towards the left.

■ Detailed Explanation

Command Method

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

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

Data Type

	Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
Heading Bit Number	○	×	×	×	×
Number to be Shifted	×	○	×	×	○
Bit Width	×	○	×	×	○

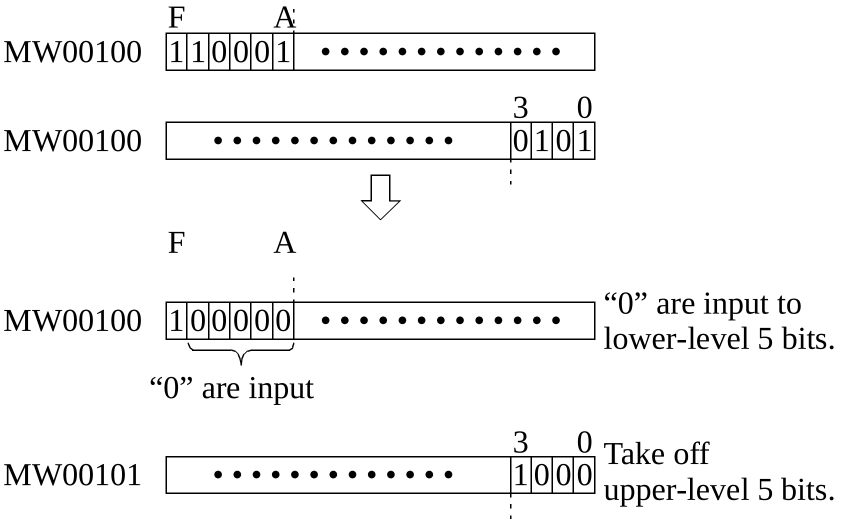
○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	SFR MB001000 N=MW00101 W=MW00102;	SHFTR MB001000 N=MW00101 W=MW00102
L	—	—
F	—	—

Example: Heading: MB00100A (bit A of MW00100)/bit width: 10
Shift 5 bits towards the left.

```
SFL   MB00100A   N5   W10;
```



Supplement

With the SFL command, when a number to be shifted >= bit width, all data with a designated bit width becomes 0.

4.5.3 Block Transfer (BLK)

■ Outline

The Block Transfer (BLK) command transfers as much word data as designated, from the register heading in transfer origin to that in transfer destination.

■ Detailed Explanation

Command Method

```
BLK   MW001000   DW00100   W10;
          A          B       C

 A: Heading register in transfer origin
 B: Heading register in transfer destination
 C: Number of transfer blocks
```

Data Type

	Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
Heading Register in Transfer Origin	×	○	×	×	×
Heading Register in Transfer Destination	×	○	×	×	×
Number of Transfer Blocks	×	○	×	×	○

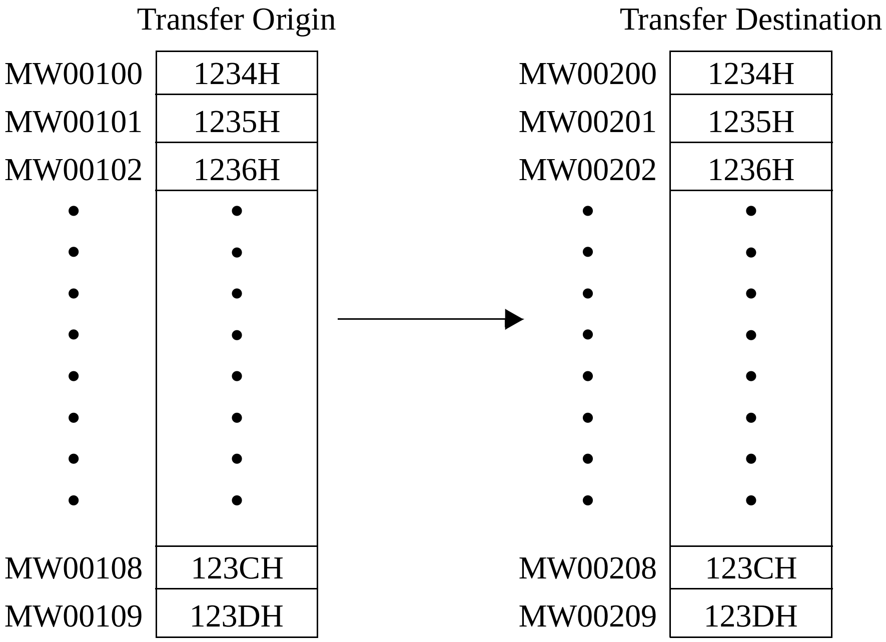
○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	BLK MW001000 DW00100 W=10;	COPYW MW001000⇒DW00100 W=10
L	—	—
F	—	—

Example: Transfer MW00100~MW00109 to MW00200~MW00209.

```
BLK MW00100 MW00200 W10;
```



	Transfer Origin			Transfer Destination
MW00100	1234H		MW00200	1234H
MW00101	1235H		MW00201	1235H
MW00102	1236H		MW00202	1236H
•	•		•	•
•	•		•	•
•	•		•	•
•	•	→	•	•
•	•		•	•
•	•		•	•
•	•		•	•
•	•		•	•
MW00108	123CH		MW00208	123CH
MW00109	123DH		MW00209	123DH

Supplement

Even though data is duplicated in both the transfer origin and destination, the block of transfer data still moves to the transfer destination.

4.5.4 Clear (CLR)

■ Outline

The Clear (CLR) command clears the word data in designated blocks to 0, from the heading register of the data to be cleared.

■ Detailed Explanation

Command Method

```
CLR   MW001000    W10;
          A         C

 A: Heading of the data to be cleared
 C: Number of blocks
```

Data Type

	Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
Heading of the data to be cleared	×	○	×	×	×
Number of Transfer Blocks	×	○	×	×	○

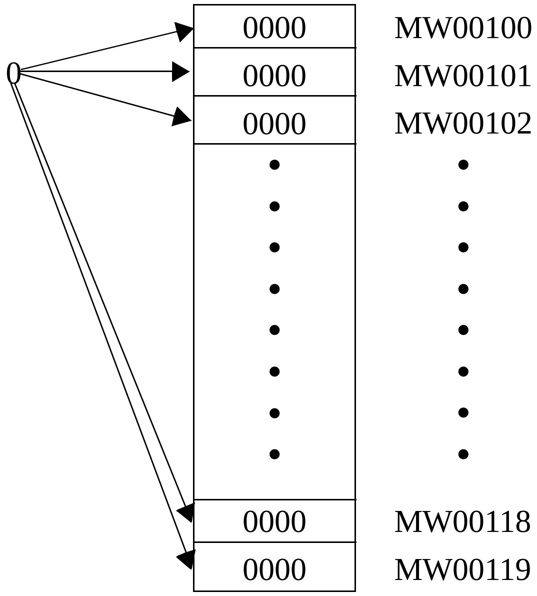
○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	CLR MW00100 W=10;	SETW MW001000 DW0000 W=10
L	—	—
F	—	—

Example: Clear the content of MW00100~MW00119 to 0.

```
CLR MW00100 W20;
```



4.6 Basic Functions

This section explains the basic functions including trigonometric function, square root, BIN, BCD, etc.

4.6.1 Sine (SIN)

■ Outline

The Sine (SIN) command leaves the sine of the integer number and real number data as a calculation result. The 32-bit integer number data cannot be used.

■ Detailed Explanation

Command Method

```
MW00100=SIN(MW00101);
MW00100=SIN(90);
MF00200=SIN(MF00202);
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
×	○	×	×	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	MW00102=SIN(MW00100);	├─MW00100 SIN⇒MW00102
L	—	—
F	DF00202=SIN(DF00200);	║├─DF00200 SIN⇒DF00202

Input unit and output results vary depending on the integer number and real number data.

- Integer Number
  It is used within a range of -327.68°~327.67°. Use an immediate calculation result (integer number data) as an input, and leave the calculation result in the integer number register (1 input unit = 0.01°). The calculation result is output by multiplying by 10000°.
- Real Number
  Use an immediate calculation result (real number data) as an input, and leave the sine in the real number register (unit: degree).

(Ex.)

- Integer Number Data

```
MW00102=SIN(MW00100);
 (05000)         (03000)
```

Equivalence → 0.5=SIN30°

- Real Number Data

```
MF00102=SIN(MF00100);
 (0.5)           (30.0)
```

Supplement

When inputting an integer number data outside of the range of -327.68°~327.67, the correct result cannot be obtained.

4.6.2 Cosine (COS)

■ Outline

The Cosine (COS) command leaves the cosine of the integer number and real number data as a calculation result. The 32-bit integer number data cannot be used.

■ Detailed Explanation

Command Method

```
MW00100=COS(MW00101);
MW00100=COS(90);
MF00200=COS(MF00202);
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
×	○	×	○	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	MW00102=COS(MW00100);	├─MW00100 COS⇒MW00102
L	—	—
F	DF00202=COS(DF00200);	│├─DF00200 COS⇒DF00202

Input unit and output results vary depending on the integer number and real number data.

- Integer Number
  It is used within a range of -327.68°~327.67°. Use an immediate calculation result (integer number data) as an input, and leave the calculation result in the integer number register (1 input unit = 0.01°). The calculation result is output by multiplying by 10000°.
- Real Number
  Use an immediate calculation result (real number data) as an input, and leave the cosine in the real number register (unit: degree).

(Ex.)

- Integer Number Data

```
 MW00102=COS(MW00100);    Equivalence →  0.5=COS60°
 (05000)       (06000)
```

- Real Number Data

```
 MF00102=COS(MF00100);
 (0.5)         (60.0)
```

Supplement

When inputting an integer number data outside of the range of -327.68°~327.67, the correct result cannot be obtained.

4.6.3 Tangent (TAN)

■ Outline

The TAN command uses a designated variable and constant (unit: degree) as an input, and leaves the tangent in the real number register.

■ Detailed Explanation

Command Method

```
MF00100=TAN(MF00102);
MF00200=TAN(1.0);
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
×	×	×	○	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	—	—
L	—	—
F	DF00202=TAN(DF00200);	┤├—DF00200 TAN⇒DF00202

Example: Calculate the tangent of input value (θ=45°): TAN (θ)=1.0.

```
DF00102=TAN(DF00100);
 (1.0)          (45.0)
```

Supplement

The TAN command can only be used for real number data. When a bit, integer, and 32-bit integer are designated, an error occurs during compilation.

4.6.4 Arc Sine (ASN)

■ Outline

The ASN command uses a designated variable and constant (unit: degree) as an input, and leaves the arc sine to the real number register.

■ Detailed Explanation

Command Method

```
MF00100=ASN(MF00102);
MF00200=ASN(1.0);
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
×	×	×	○	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	—	—
L	—	—
F	DF00202=ASN(DF00200);	||─DF00200 ASIN⇒DF00202

Example: Calculate the arc sine of the input value (0.5): 30°=θ=ASN (0.5).

```
MF00202=ASN(MF00200);
 (30.0)          (0.5)
```

Supplement

The ASN command can only be used for real number data. When a bit, integer, and 32-bit integer are designated, an error occurs during compilation.

4.6.5 Arc Cosine (ACS)

■ Outline

The ACS command uses a designated variable and constant (unit: degree) as an input, and leaves the arc cosine to the real number register.

■ Detailed Explanation

Command Method

```
MF00100=ACS(MF00102);
MF00200=ACS(60.0);
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
×	×	×	○	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	—	—
L	—	—
F	DF00202=ACS(DF00200);	‖—DF00200 ACOS⇒DF00202

Example: Calculate the arc cosine of input value (0.5): 60°=θ=ACS (0.5)

```
MF00100=ACS(MF00102);
 (0.5)          (60.0)
```

Supplement

The ACS command can only be used for real number data. When a bit, integer, and 32-bit integer are designated, an error occurs during compilation.

4.6.6 Arc Tangent (ATN)

■ Outline

The ATN command leaves the arc tangent of the integer number and real number data as a calculation result. The 32-bit integer number data cannot be used.

■ Detailed Explanation

Command Method

```
MW00100=ATN(MW00102);
MW00100=ATN(100);
MF00200=ATN(MF00202);
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
×	○	×	○	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	MW00102=ATN(MW00100);	├─MW00100 ATN⇒MW00102
L	—	—
F	DF00202=ATN(DF00200);	│├─DF00200 ATN⇒DF00202

Input unit and output results vary depending on the integer number and real number data.

- Integer Number
  It is used within a range of -327.68°~327.67°. Use an immediate calculation result (integer number data) as an input, and leave the calculation result in the integer number register (1 input unit = 0.01°). The calculation result is output by multiplying by 100°.
- Real Number
  Use an immediate calculation result (real number data) as an input, and leave the inverse tangent in the real number register (unit: degree).

Example:

- Integer Number Data

```
 MW00100=ATN(MW00102);   Equivalence →   45=ATN(1.0)
  (04500)        (00100)
```

- Real Number Data

```
 MF00100=ATN(MF00102);
  (45.0)         (1.0)
```

4.6.7 Square Root (SQT)

■ Outline

The SQT command leaves the square root of the integer and real number as a calculation result. The 32-bit integer number data cannot be used.

■ Detailed Explanation

Command Method

```
MW00100=SQT(MW00101);
MW00100=SQT(100);
MF00200=SQT(MF00202);
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
×	○	×	○	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	MW00102=SQT(MW00100);	├─MW00100 SQRT⇒MW00102
L	—	—
F	DF00202=SQT(DF00200);	‖├─DF00200 SQRT⇒DF00202

Input unit and output results vary depending on the integer number and real number data.

- Integer Number
  The calculation format (shown below) is different from the mathematical square root.

$$32768 \times sign(A) \times \sqrt{|A|/32768}$$

$sign(A)$: Mark of the A register
$|A|$: Absolute value of the A register
That is, the result of the mathematical square root is multiplied by $\sqrt{32768}$. When inputting a negative number, calculate the square root of the absolute value, and leave the negative number as a calculation result. The calculation error is ±2.

- Real Number
  Use an immediate calculation result (real number data) as an input, and leave the square root to the real number data.

Example:

- Integer Number Data

When an integer is input:

```
MW00100=SQT(MW00102);
 (01448)        (00064)
```

$$\underset{(8)}{\sqrt{64}} \times \underset{(181)}{\sqrt{32768}} = 1448$$

When a negative number is input:

```
MW00100=SQT(MW00102);
 (-01448)       (-00064)
```

$$\left\{\underset{(8)}{\sqrt{64}} \times \underset{(181)}{\sqrt{32768}}\right\} = 1448$$

- Real Number Data

When an integer is input:

```
MF00100=SQT(MF00102);
 (8.0)          (64.0)
```

When a negative number is input:

```
MF00100=SQT(MF00102);
 (-8.0)         (-64.0)
```

4.6.8 BCD→BIN (BIN)

■ Outline

The BIN command converts the BCD code data to a binary number (BIN code). It can only be used for integer number data. If a value other than the BCD code is designated, the correct result cannot be obtained.

■ Detailed Explanation

Command Method

```
MW00100=BIN(MW00101);
MW00100=BIN(1234H);
MF00200=BIN(ML00202);
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
×	○	○	×	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	MW00101=BIN(MW00100);	├─MW00100 BIN⇒MW00101
L	ML00102=BIN(MW00100);	├─ML00100 BIN⇒ML00101
F	—	—

(Example 1)

Convert

						⇨					
MW00101	1	2	3	4		⇨	MW00101	0	4	D	2

(1234H) 1234 (Decimal)

(Example 2)

Convert

MW00101	1	2	3	F	⇨	MW00101	0	4	D	D

(1234H) 1245 (Decimal)

Supplement

If data other than the BCD code is designated, the correct result cannot be obtained.

4.6.9 BIN→BCD (BCD)

■ Outline

The BCD command converts a binary number (BIN code) to the BCD code. It can only be used for the integer number data. If the BIN data is greater than 9999, or it is a negative value, the correct result cannot be obtained.

■ Detailed Explanation

Command Method

```
MW00100=BCD(MW00101);
MW00100=BCD(1234);
MF00200=BCD(ML00202);
```

Data Type

Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
×	○	○	×	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	—	—
W	MW00101=BCD(MW00100);	├─MW00100 BCD⇒MW00101
L	ML00102=BCD(MW00100);	├─ML00100 BCD⇒ML00102
F	—	—

(Example 1)

MW00101	0	4	D	2	Convert ⇨	MW00100	1	2	3	4

1234 (Decimal) (1234H)

(Example 2)

MW00101	3	0	3	9	Convert ⇨	MW00100	C	3	4	5

12345 (Decimal) (C345H)

Supplement

If the BIN data is greater than 9999, the correct result cannot be obtained.

4.6.10 Designated Bit ON (S{ })

■ Outline

The Designated Bit ON (S{}) command turns a designated bit ON if the logical calculation result is true. However, it does not turn a designated bit OFF if the result is false.

■ Detailed Explanation

Command Method

```
S{MB001000}=MB001010&MB001011
      A                B

 A: Designated bit
 C: Logical Calculation format
```

Data Type

	Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
Designated Bit	○	×	×	×	×
Logical Calculation Format	○	×	×	×	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	S{MB001000}=MB001010&MB001011	—
W	—	—
L	—	—
F	—	—

4.6.11 Designated Bit OFF (R{ })

■ Outline

The Designated Bit OFF (R{}) command turns a designated bit OFF if the Logical Calculation result is true. However, it does not turn a designated bit OFF if the result is false.

■ Detailed Explanation

Command Method

```
R{MB001000}=MB001010&MB001011
      A                B

 A: Designated bit
 C: Logical Calculation format
```

Data Type

	Bit (B)	Integer (W)	32-bit Integer (L)	Real (F)	Constant
Designated Bit	○	×	×	×	×
Logical Calculation Format	○	○	○	×	○

○ = Yes × = No

■ Program Example

Type	Motion Program	Ladder Program
B	R{MB001000}=MB001010&MB001011	—
W	—	—
L	—	—
F	—	—

5 Variables (Registers)

This chapter contains information on the variables that can be used in the motion programs.

5.1 Outline

In this section, an outline of the variables is described.

5.1.1 Variable Overview

In motion programs the variables can be used to describe values instead of directly giving the values. When using variables, the motion program uses the values stored in an area reserved for variables.

■ Variable (Register) Type

Motion programs can use the following 7 types of registers (described in the following table) as variables. The S, M, I, O, and C registers are called global variables which can be used in both motion programs and ladder programs. The # and D registers are called local variables which are only capable of referencing within an individual program. However, the local variables are secured in program memories; they take space from the programs.

Variable Type

Type	Name	Command Method	Range	Content	Characteristic
S	System Register	SB, SW, SL, SFnnnnn	SW00000 ~SW01023	Registers prepared in the system. The register number nnnnn is displayed with a decimal. When the system starts, the SW00000~SW00049 is cleared to 0.	Used by all kinds of programs
M	Data Register	MB, MW, ML, MFnnnnn	MW00000 ~MW32767	Registers used for the I/F between each DWG. The register number nnnnn is displayed with a decimal.	
I	Input Register	IB, IW, IL, IFhhhh	IW0000 ~IW07FF	Registers used for I/O module input data. The register number hhhh is displayed with a hexadecimal. If a register number is after C000, it is used for interface with the servo monitoring parameters.	
O	Output Register	OB, OW, OL, OFhhhh	OW0000 ~OW07FF	Registers used for I/O module output data. The register number hhhh is displayed with a hexadecimal. If a register number is after C000, it is used for interface with the servo setting parameters.	
C	Constant Register	CB, CW, CL, CFhhhhh	CW00000 ~CW4095	Registers that can be referenced only by a program. The register number nnnnn is displayed with a decimal.	

Variable Type (Continued)

#	# Register	#B, #W, #L, #Fnnnnn	#W00000~ #W16383	Registers that are only referenced within a corresponding DWG. Actual range is designated in MotionWorks™ by the user. The register number nnnn is displayed with a decimal. **However, it cannot be used in the motion programs.**	Used for individual program
D	D Register	DB, DW, DL, DFnnnnn	DW00000 ~DW16383	Registers owned by each DWG. Therefore they can only be used for the DWG. Actual range is designated in MotionWorks™ by the user. The register number nnnnn is displayed with a decimal.	

Important Point

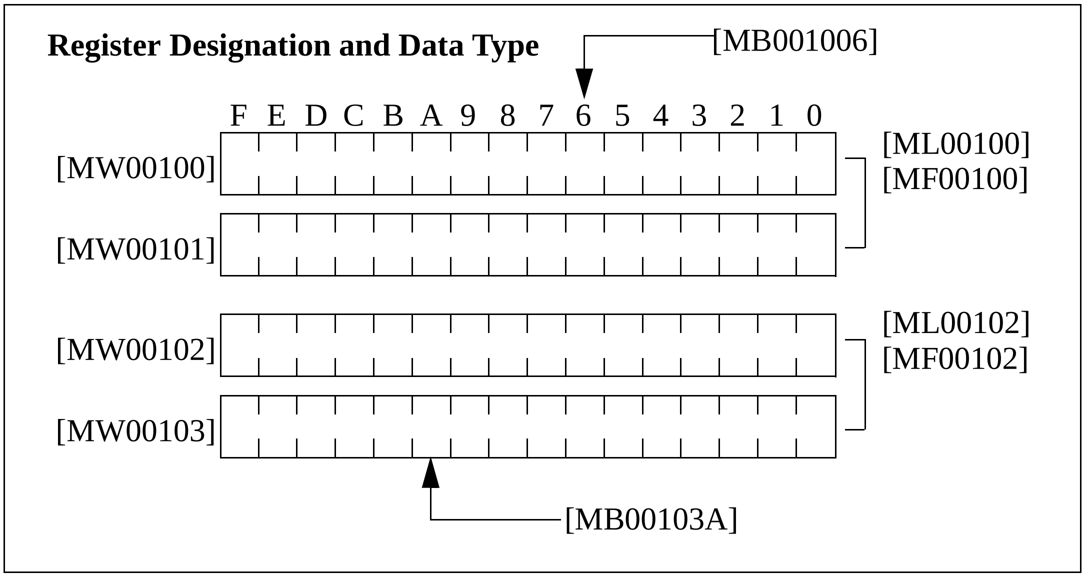
In the motion programs, the # register cannot be used. If a # register is used, a syntax error results during compilation of the motion program.

■ Data Type

As shown below, there are bit, integer number, 32-bit integer number, and real number data. They are used for different purposes.

Data Type

Code	Data Type	Value Range	Notes
B	Bit	ON(1), OFF(0)	Used for relay sequence or ON/OFF condition distinction.
W	Integer	-32768~+32767 (8000H~7FFFH)	Used in value calculations; as well as logical calculations if in the ( ).
L	32-bit	-2147483648~+2147483647 (80000000H~7FFFFFFFH)	Used in value calculations; as well as logical calculations if in the ( ).
F	Real	±(1.175E-38~3.402E+38),0	Used in advanced value calculations.



5.1.2 Global and Local Variables

■ Global Variables

The global variables can be used in ladder programs, user functions, and motion programs. In other words, a result calculated in a ladder program can be used for user function or a motion program. The size of each global variable is secured in the system.

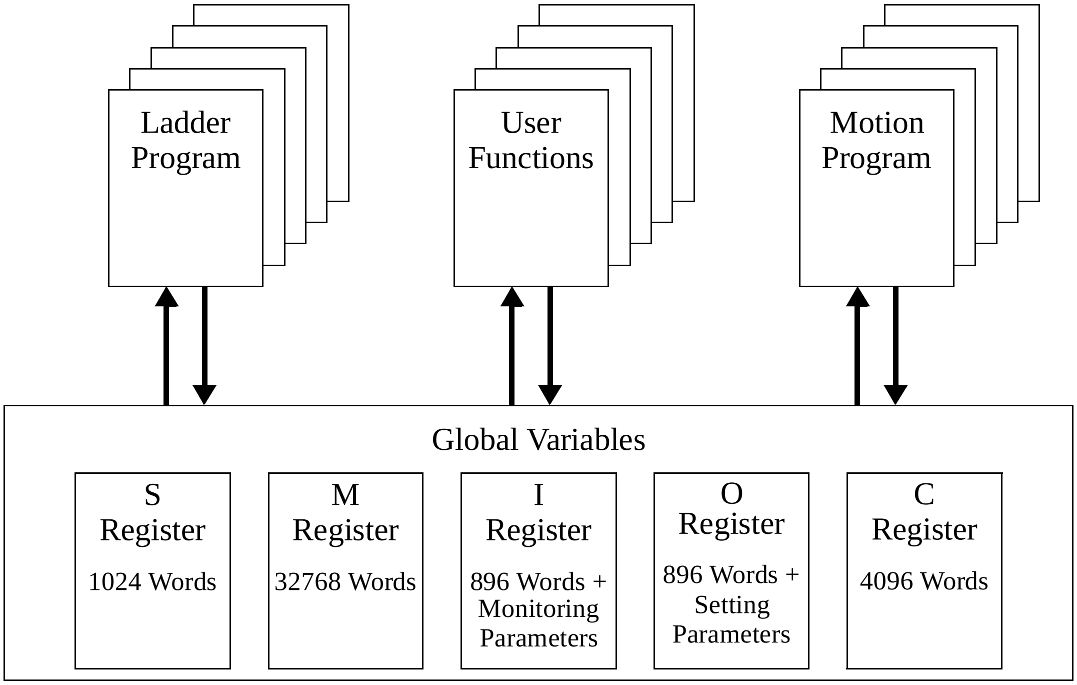


Figure 5.1: Global Variables

■ Local Variables

Local variables are used for each program. They are secured in each program; therefore, the result calculated in each program is stored in that program.

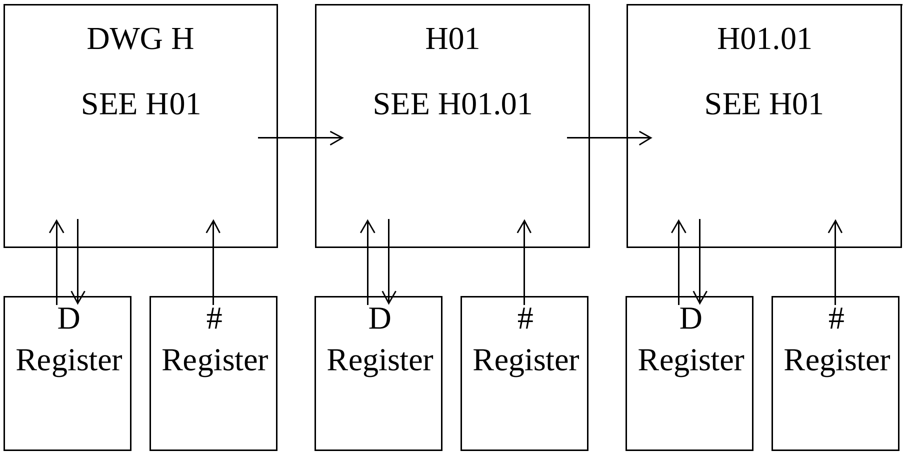


Figure 5.2: Local Variables

The range that variables use in each program is designated in the DWG and Motion Program Configuration screens. Up to 16kW can be secured for each DWG.

Important Point

In a motion program, the # register cannot be used.

■ Important Points During Variable Calculations

1. An error occurs in the following program:

- Integer data is stored in a bit variable

```
MB000100=123;
MB000100=MW00100;
```

2. When a type of data is stored in a different variable, the following results occur:

- When real number data is stored in an integer number variable:

```
MB000100=MF00200;
(00001)      (1.234)
```

The real number value is stored by converting to an integer number.

- When real number data is stored in a 32-bit integer number variable:

```
ML000100=MF00200;
(1234567)    (123456.7)
```

The real number value is stored by converting to an integer number.

- When 32-bit integer number data is stored in an integer number variable:

```
MW000100=ML00200;
(1234567)   (123456.7)
```

The lower-level 16bit of the 32-bit integer number data is stored untouched.

- When integer number data is stored in a 32-bit integer number variable:

```
ML000100=MW00200;
(0001234)    (1234)
```

The integer number data is stored by converting to a 32-bit integer number.

3. When storing real number data in an integer number variable, the value will be rounded-off. Values of x.50 and less are rounded down; values of x.51 and more are

rounded up.

```
MW00100=MF00200 + MF00202;
(0124)      (123.49)        (0.02)
(0123)      (123.49)        (0.01)
```

The operation result varies according to the value of the variable being calculated.

5.2 How to Use the Variables

This section explains how to use each variable.

5.2.1 System Variables

■ Outline

The system variables (S registers) are prepared in the machine controller system; they can read the error or operation status of the system. The S registers can be used among the motion programs.

■ Detailed Explanation

The S register command method is shown as follows:

```
SB000000   ~   SB01023F

SW000000   ~   SW01023

SL000000   ~   SL01023

SF000000   ~   SF01023
```

The variable numbers are displayed with decimals. However, the BIT number designated by the bit is displayed with a hexadecimal.

■ Program Example

```
• Bit designation
  OB000010=SB000402|SB000403;
• Integer number designation
  MW00100=SW00041;
• 32-bit integer number designation
  ML00100=SL00062;
• Real number designation
  MF00200=SF00085;
```

Important Point

The S registers are dedicated for status read only. Therefore if executing the write, the system motion cannot be guaranteed.

5.2.2 Data Variables (M Registers)

■ Outline

The data variables (M registers) can be used in ladder programs, user functions, and each motion program; they also handle the interface between the motion programs, as well as the ladder programs.

■ Detailed Explanation

The M register command method is shown as follows:

MB000000	~	MB32767F
MW000000	~	MW32767
ML000000	~	ML32767
MF000000	~	MF32767

The M registers are used as variables to substitute the calculation results, or designate the positioning coordinate values and speeds with variables. The variable numbers are displayed with decimals.

■ Program Example

When designating the positions and speeds with variables during an axis movement command

```
• Parameter (Command unit: mm, decimal place: 3)
  ML00100=100000;                               → 1000.000mm
  ML00102=200000;                               → 2000,000mm
  ML00104=300000;                               → 3000,000mm
  ML11016=500000;                               → 5000,000mm/min
  MVS[X]ML00100[Y]ML00102[Z]ML00104FML00106;
```

When using the variables in the calculations

```
• Bit designation
  MB001001=IB000100&IB000201;
• Integer number designation
  MW00101=(MW00101|MW00102)&OFF0CH
• 32-bit integer number designation
  ML00200=(MI00202*ML00204/ML00206)*3:
• Real number designation
  MF00200=MF00202*MF00204/MF00206*3.14;
```

Important Point

When designating the moving amount coordinate values or speeds with variables during the following motion commands, the 32-bit integer number 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)

■ Outline

The input variables (I registers) use external input signals and servo monitoring parameters. In this case, it is possible to write the servo parameters, but the values cannot be guaranteed.

■ Detailed Explanation

The I register command method is shown as follows:

```
IW0000   ~   IW07FF

IWC000   ~   IWC37F
```

Register numbers of the external input signals:

• Local input	IW0000 (IB00000~IB0000F 16 points)
• Remote input	According to the address set in the Module Configuration.

Register numbers of the servo monitoring parameters:

Axis Number	IW Address	Axis Number	IW Address
1	IWC000~IWC03F	8	IWC1C0~IWC1FF
2	IWC040~IWC07F	9	IWC200~IWC23F
3	IWC080~IWC0BF	10	IWC240~IWC27F
4	IWC0C0~IWC0FF	11	IWC280~IWC2BF
5	IWC100~IWC13F	12	IWC2C0~IWC2FF
6	IWC140~IWC17F	13	IWC300~IWC33F
7	IWC180~IWC1BF	14	IWC340~IWC37F

■ Program Example

Designate positions and speeds with the variables during an axis movement command.

```
• Bit designation
  MB01000=IB0010&IB00105;
• Integer number designation
  MW00100=IWC016;
• 32-bit integer number designation
  ML00100=ILC022;
```

Supplement

The register number for each axis in the servo monitor parameter is obtained by using the following formula:

Each axis monitoring parameter register address = IWC000+(Axis number -1)×40(HEX)

5.2.4 Output Variables (O Registers)

■ Outline

The output variables (O registers) use external output signals and servo setting parameters.

■ Detailed Explanation

The O register command method is shown as follows:

OW0000	~	OW07FF
OWC000	~	OWC37F

Register numbers of the external output signals

• Local output OW00001 (OB000010~OB0000IF 16 points)
• Remote output According to the address set in the Module Configuration.

Register numbers of the servo setting parameters

Axis Number	OW Address	Axis Number	OW Address
1	OWC000~OWC03F	8	OWC1C0~OWC1FF
2	OWC040~OWC07F	9	OWC200~OWC23F
3	OWC080~OWC0BF	10	OWC240~OWC27F
4	OWC0C0~OWC0FF	11	OWC280~OWC2BF
5	OWC100~OWC13F	12	OWC2C0~OWC2FF
6	OWC140~OWC17F	13	OWC300~OWC33F
7	OWC180~OWC1BF	14	OWC340~OWC37F

■ Program Example

Designate the positions and speeds with variables during an axis movement command.

• Bit designation
OB01000=MB001000&IB00100;
• Integer number designation
OWC020=MW00100;
• 32-bit integer number designation
OLC022=ML00100+ML00200;
• Real number designation
OF00200=MF00100+MF00102;

Supplement

The register number for each axis in the servo monitor parameter is obtained using the following formula:

Axis setting parameter register address = OWC000+(Axis number -1)×40(HEX)

5.2.5 Constant Variables (C Registers)

■ Outline

The constant variables (C registers) are only capable of referring to the data created by a parameter list within a program. The write function is disabled.

■ Detailed Explanation

The C register command method is shown as follows:

```
CW0000    ~    CW4095
```

■ Program Example

When using the variables in the calculations

```
• Bit designation
  MB001000=CB001001;
• Integer number designation
  MWC00100=CW00100;
• 32-bit integer number designation
  MLC00100=CL00100;
• Real number designation
  MF00100=CF00100;
```

5.2.6 D Variables (D Registers)

■ Outline

The D variables (D registers) are the internal variables owned by each motion program. They can only be used within the corresponding programs.

■ Detailed Explanation

The D register command method is shown as follows:

```
DW00000    ~    DW10240 (maximum)
```

The D registers are used as variables to substitute the calculation results, or designate the positioning coordinate values and speeds with variables. The variable numbers are displayed with decimals.
The size is set up by the Program Configuration: Property. (Default: 32 words)

■ Program Example

When designating position and speed during an axis movement command execution

```
• Parameter (Command unit: mm, decimal place: 3)
  DL00100=100000;                              → 1000.000mm
  DL00102=200000;                              → 2000,000mm
  DL00104=300000;                              → 3000,000mm
  DL11016=500000;                              → 5000,000mm/min
  MVS[X]DL00100[Y]DL00102[Z]DL00104FDL00106;
```

When using the variables in the calculations

```
• Bit designation
  DB001000=IB001001&MB000101;
• Integer number designation
  DW00102=CW00103|DW00104&DW00105;
• 32-bit integer number designation
  DLC00106=DL00108*PL0011/ML00200;
• Real number designation
  DF00200=MF00202*DF00202*3.14;
```

Important Point

When designating the moving amount coordinate values or speeds with variables during the following motion commands, the 32-bit integer number must be used.
MOV, MVS, MCW/MCC, ZRN, SKP, MVT, EXM, POS,
ACC, SCC, IAC, IDC, IFP, FMX, INP

YASKAWA
A World of Automation Solutions™

YASKAWA ELECTRIC AMERICA, INC.
Chicago-Corporate Headquarters   2121 Norman Drive South, Waukegan, IL 60085, U.S.A.
Phone: (847) 887-7000   Fax: (847) 887-7310   Internet: http://www.yaskawa.com

MOTOMAN INC.
805 Liberty Lane, West Carrollton, OH 45449, U.S.A.
Phone: (937) 847-6200   Fax: (937) 847-6277   Internet: http://www.motoman.com

YASKAWA ELECTRIC CORPORATION
New Pier Takeshiba South Tower, 1-16-1, Kaigan, Minatoku, Tokyo, 105-0022, Japan
Phone: 81-3-5402-4511   Fax: 81-3-5402-4580   Internet: http://www.yaskawa.co.jp

YASKAWA ELETRICO DO BRASIL COMERCIO LTDA.
Avenida Fagundes Filho, 620 Bairro Saude Sao Paolo-SP, Brasil CEP: 04304-000
Phone: 55-11-5071-2552   Fax: 55-11-5581-8795   Internet: http://www.yaskawa.com.br

YASKAWA ELECTRIC EUROPE GmbH
Am Kronberger Hang 2, 65824 Schwalbach, Germany
Phone: 49-6196-569-300   Fax: 49-6196-888-301   Internet: http://www.yaskawa.de

MOTOMAN ROBOTICS AB
Box 504 S38525, Torsas, Sweden
Phone: 46-486-48800   Fax: 46-486-41410

MOTOMAN ROBOTEC GmbH
Kammerfeldstrabe 1, 85391 Allershausen, Germany
Phone: 49-8166-900   Fax: 49-8166-9039

YASKAWA ELECTRIC UK LTD.
1 Hunt Hill Orchardton Woods Cumbernauld, G68 9LF, Scotland, United Kingdom
Phone: 44-12-3673-5000   Fax: 44-12-3645-8182

YASKAWA ELECTRIC KOREA CORPORATION
Paik Nam Bldg. 901 188-3, 1-Ga Euljiro, Joong-Gu, Seoul, Korea
Phone: 82-2-776-7844   Fax: 82-2-753-2639

YASKAWA ELECTRIC (SINGAPORE) PTE. LTD.
Head Office: 151 Lorong Chuan, #04-01, New Tech Park Singapore 556741, SINGAPORE
Phone: 65-282-3003   Fax: 65-289-3003

TAIPEI OFFICE (AND YATEC ENGINEERING CORPORATION)
10F 146 Sung Chiang Road, Taipei, Taiwan
Phone: 886-2-2563-0010   Fax: 886-2-2567-4677

YASKAWA JASON (HK) COMPANY LIMITED
Rm. 2909-10, Hong Kong Plaza, 186-191 Connaught Road West, Hong Kong
Phone: 852-2803-2385   Fax: 852-2547-5773

BEIJING OFFICE
Room No. 301 Office Building of Beijing International Club,
21 Jianguomanwai Avenue, Beijing 100020, China
Phone: 86-10-6532-1850   Fax: 86-10-6532-1851

SHANGHAI OFFICE
27 Hui He Road Shanghai 200437 China
Phone: 86-21-6553-6600   Fax: 86-21-6531-4242

SHANGHAI YASKAWA-TONJI M & E CO., LTD.
27 Hui He Road Shanghai 200437 China
Phone: 86-21-6533-2828   Fax: 86-21-6553-6677

BEIJING YASKAWA BEIKE AUTOMATION ENGINEERING CO., LTD.
30 Xue Yuan Road, Haidian, Beijing 100083 China
Phone: 86-10-6232-9943   Fax: 86-10-6234-5002

SHOUGANG MOTOMAN ROBOT CO., LTD.
7, Yongchang-North Street, Beijing Economic & Technological Development Area,
Beijing 100076 China
Phone: 86-10-6788-0551   Fax: 86-10-6788-2878

YEA, TAICHUNG OFFICE IN TAIWAN
B1, 6F, No. 51, Section 2, Kung-Yi Road, Taichung City, Taiwan, R.O.C.
Phone: 886-4-2320-2227 Fax: 886-4-2320-2239
Phone: 55-11-5071-2552   Fax: 55-11-5581-8795   Internet:http://www.yaskawa.com.br

Yaskawa Electric America, Inc., September 2001
YEA-SIA-C887-1.3B Printed In U.S.A.