MITSUBISHI

MITSUBISHI PERSONAL MACHINE CONTROLLER

MODEL W

MOTION PROGRAM MANUAL

BNP-B2108*

CONTENTS

1. OVERVIEW

This manual provides the formats and details of the functions which can be specified in motion programs. For the pages corresponding to the functions, refer to the function list given at the end of this manual.
For the errors of motion programs, refer to the instruction manual.

1.1 Motion Program

A variety of instructions, such as axis move commands and signal output command, are available for motion programs. The format of each instruction you can give is:
Instruction (data, data, ...) CRLF
or
Instruction CRLF
(CRLF is a line feed character. CR may be omitted.)
For instance, an axis move command is written:

MOV(100.000)

Alternatively, an incremental command given for an axis to move in incremental value is written as follows:

IST

Data required for each instruction (hereafter called an "argument") is written within the parentheses following the instruction. When two or more arguments are needed, use "," to separate them. To move two axes with interpolation performed, for example:

MOV(120.23,23.98)

In some instructions, arguments may be omitted. In this case, "," may also be omitted. For instance, move commands are represented as follows:

MOV(10.001,20.002,30.003)	When all three axes are to be positioned, they are separated by ",".
MOV(10.001,20.002)	When the first and second axes are to be positioned, "," used to separate the second and third axis positions may be omitted.
MOV(10.001,,30.003)	When the first and third axes are to be positioned, "," used to separate the position data of the first and second axes and those of the second and third axes cannot be omitted.
MOV(,20.002,30.003)	When the second and third axes are to be positioned, "," used to separate the position data of the first and second axes cannot be omitted.

Similarly,

MOV(10.001)	When the first axis is to be positioned
MOV(,20.002)	When the second axis is to be positioned
MOV(,,30.003)	When the third axis is to be positioned

Namely, "," can be omitted when the argument(s) placed later is (are) omitted. "," cannot be omitted when any of the arguments placed later is written.

1.2 Variables and Operation

1.2.1 Variables

In motion programs, you can use variables. Variables are represented by "# variable numbers". Variables available for the motion programs are:

#100 to #149 Variables corresponding to operations
#500 to #599 Variables common to operations

You can also attach names to the variables.
The variables corresponding to operations are variables provided for operation numbers started. For instance, variable #110 of the motion program run under operation number 1 is different from variable #110 of the motion program run under operation number 2.
The variables common to operations are common variables independent of operation numbers. For example, variable #520 of the motion program run under operation number 1 is the same as variable #520 of the motion program run under operation number 2. Therefore, when 100.01 is assigned to #520 in the motion program run under operation number 1, the value of variable #520 also turns to 100.01 in the motion program run under operation number 2.
The variables corresponding to operations and variables common to operations are all battery-backed. When data during power-off is not to be used, the program must be written to assign initial values (e.g. #100=0).

1.2.2 Operation and branch instructions

Operation instructions and various functions used between variables are available for motion programs. As branch and other instructions are also available, you can change the moving order of axes according to conditions.

1.2.3 Use of variables in instructions

In motion programs, you can use a variable as the argument of an instruction, e.g.:

```
#100=100.34
MOV(#100)
```

This command is equal to:

```
MOV(100.34)
```

Since variables can be used as the arguments of instructions as described above, you can use variables with operation, branch and other instructions to easily write a program in which operation is repeated to change a travel little by little, for example.

1.3 Command Functions

You can specify the command functions in motion programs. However, the variables of the motion programs cannot be used as the arguments of the motion programs (except when the variables of the motion programs are specified in $DSET and $WRIT).
With the command functions, you can access various data of the machine controller, copy files, and give simple control commands. By using the command functions in motion programs, you can incorporate a wide variety of functions.
For full information on the command functions, refer to the command function manual.

Example 1: Simple teaching function

Assign the contents of an R register in a PLC to a variable. By using that value as a point number, set the current position in the point table. Starting this motion program from the PLC will achieve a simple teaching function.

GPR(X,Y)	Specifies the axes to which data will be set.
$DSET(PLC,R1000,L_VAL,#100)	Assigns the value of R1000 to #100 (command function).
PRD(#100)	Sets the current position to the point.
STOP	

2 MOVING THE AXES

Operated Axes
Axis Move Commands
Speed Command
Ending the Program

Operated Axes	Operated Axis Command

Format

GRP(first axis name, second axis name, third axis name)

Used to specify the axes operated in a motion program. Also used to change the operated axes at any point in the motion program.
The axis names used should be those set in the parameters. Up to three axes may be specified by a character string of up to six characters long.
Specifying more than four axes or no axes will result in an error.
If any axis specified in the operated axis command has already been operated by the other program or PLC, an error will occur and operation terminated.
If any other command is given without the operated axis command, an error will occur and operation terminated.
Executing this command will result in the following status:

SPD(1)	Speed as set in the first speed parameter
AST	Absolute value command
PCF	Deceleration check not made

Example

GRP(ARM1, HND1, TBL1)	Defines the ARM1, HND1 and TBL1 axes as operated axes.
GRP(ROL)	Defines the ROL axis as an operated axis.

Axis Move Command	Positioning

Format

MOV(first axis position, second axis position, third axis position)

The axes move at the speed specified in the speed command, with interpolation performed. The axes specified in the GRP command move to the positions specified in MOV. When there is an axis which you do not want to move, the position of that axis may not be specified. Specifying no axis positions will result in an error.

Example

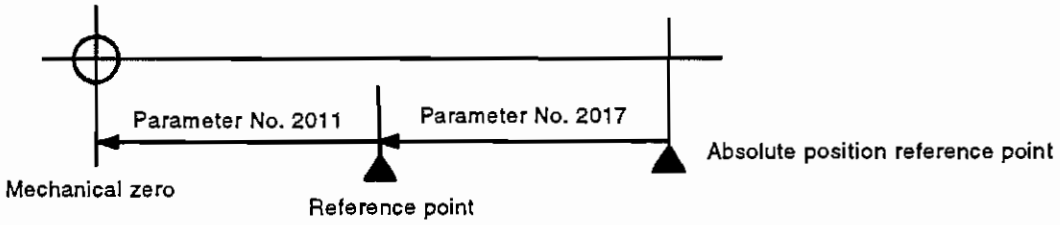
```
AST
MOV(10. 0, 20. 0, 30. 0)    Moves the first axis to 10. 0, the second axis to 20. 0 and the
                            third axis to 30. 0.
MOV(, 40. 0)                Moves the second axis to 40. 0.
```

Axis Move Command	Reference Point Return

Format

ZRN

Returns all the axes specified as operated axes to the reference points.
The reference point is a position offset from the mechanical zero by the mechanical zero offset value set in the parameter.
For the axis provided with an incremental encoder, the first return to the reference point after power-on is made in a dog mechanism. After the return to the reference point, the mechanical co-ordinates are determined.



Example

```
GRP(ARM1, HND1, TBL1)
ZRN                         Moves the ARM1, HND1 and TBL1 axes to the reference
                            points.
```

Speed Command	Parameter Speed Command

Format

SPD(parameter number)

Sets the travel speed of the axis move command, point command feed or circular interpolation to the speed set in the parameter.
There are five speed parameters (parameters No. 1041 to 1045). Specify the parameter number in the parameter speed command. If the parameter setting is 0, a program error will occur.

Example

SPD(1)	Sets the travel speed to the speed set in parameter No. 1041.
SPD(5)	Sets the travel speed to the speed set in parameter No. 1045.

Ending Run	Ending Run

Format

STOP

Ends the run of a motion program.
After the operated axes have completed specified deceleration, the start signals of those axes switch off to switch off the motion program run signal.
At the end of a program file, this command may be omitted.

Example

STOP	Ends run.

3. VARIOUS CONTROL FUNCTIONS

Position Command Units
Axis Move Commands
Timer
Speed Command
External Signal Outputs
Point Commands
Deceleration Check

Position Command	Unit Absolute Command

Format

AST

Defines the position setting of the axis move command or point command as absolute values. Motion program operation starts with the absolute value command.

Example

AST	Sets the position command unit to absolute values.
MOV(10. 0, 20. 0, 30. 0)	Moves to 10. 0, 20. 0, 30. 0.
MOV(–1. 0, –2. 0, –3. 0)	Moves to -1. 0, -2. 0, -3. 0.

Position Command	Unit Incremental Command

Format

IST

Defines the position setting of the axis move command or point command as incremental values. Motion program operation starts with the absolute value command.

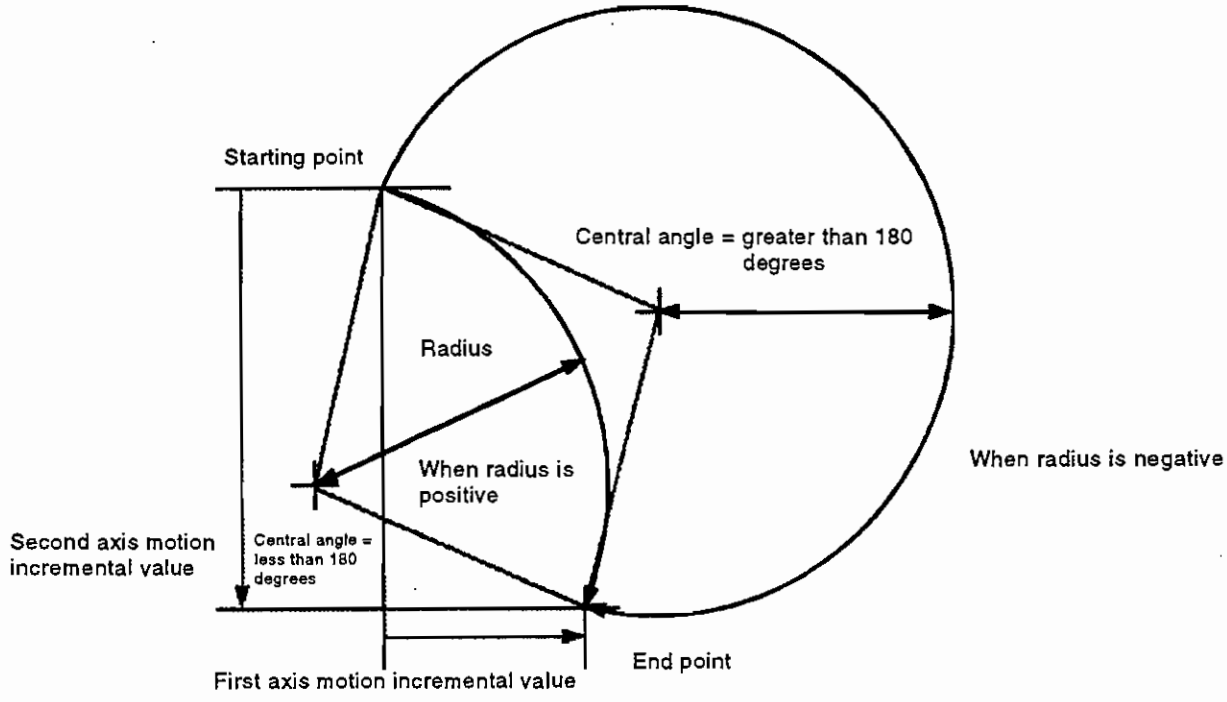
Example

AST	Sets the position command unit to absolute values.
MOV(100. 0, 100. 0, 100. 0)	Moves to 100. 0, 100. 0, 100. 0.
IST	Sets the position command unit to incremental values.
MOV(10. 0, 20. 0, 30. 0)	Moves to 110. 0, 120. 0, 130. 0.
MOV(–10. 0, –20. 0, –30. 0)	Moves to 100. 0, 100. 0, 100. 0.

Format

MCW(first axis motion incremental value, second axis motion incremental value, radius)

The axes move with circular interpolation performed at the speed specified in the speed command.
Circular interpolation is performed clockwise with the first axis denoted as the horizontal line and the second axis as the vertical line.
Circular interpolation is specified with only incremental values even in the absolute command.
If three axes have been specified in the operated axis command, circular interpolation is performed for the first and second axes only. The third axis cannot be specified.
To make the central angle less than 180 degrees, set the radius with a positive value.
To make the central angle greater than 180 degrees, set the radius with a negative value.



Example

MCW(10. 0, -20. 0, 15. 0)

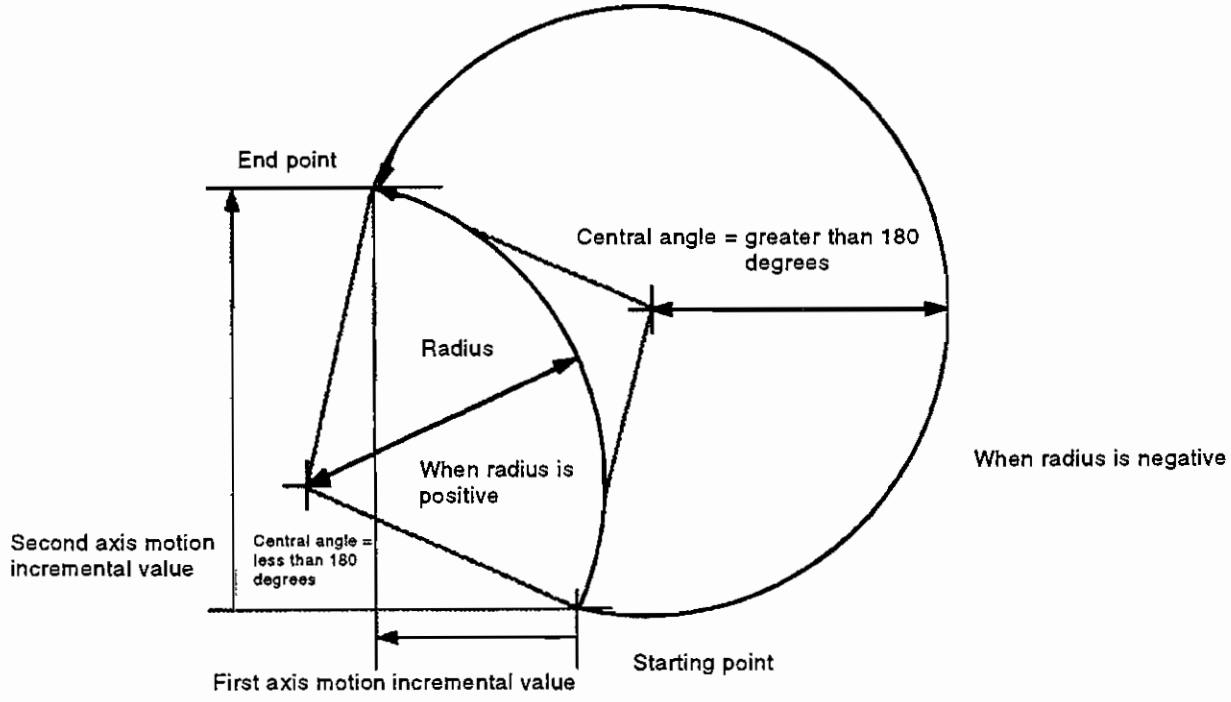
MCW(10. 0, -20. 0, 15. 0)

Axis Move Command	Circular Interpolation (Counterclockwise)

Format

MCC(first axis motion incremental value, second axis motion incremental value, radius)

The axes move with circular interpolation performed at the speed specified in the speed command.
Circular interpolation is performed counterclockwise with the first axis denoted as the horizontal line and the second axis as the vertical line.
Circular interpolation is specified with only incremental values even in the absolute command.
If three axes have been specified in the operated axis command, circular interpolation is performed for the first and second axes only. The third axis cannot be specified.
To make the central angle less than 180 degrees, set the radius with a positive value.
To make the central angle greater than 180 degrees, set the radius with a negative value.



Example

MCC(−10. 0, 20. 0, 15. 0)

MCC(−10. 0, 20. 0, −15. 0)

Timer	Timer

Format

TIM(time)

Times for the specified period. When the timer times out, the next command is executed.
Time can be set between 0 and 9999.99 seconds in increments of 10msec.

Example

TIM(10. 01) Times 10.01 seconds.

Speed Command	Feedrate Command

Format

SPN(feedrate)

Used to specify the travel speed of the axis move command, point command feed or circular interpolation. The feedrate may be set between 0.1mm/min and 99999.9mm/min.
Setting any value outside the setting range will result in an error.

Example

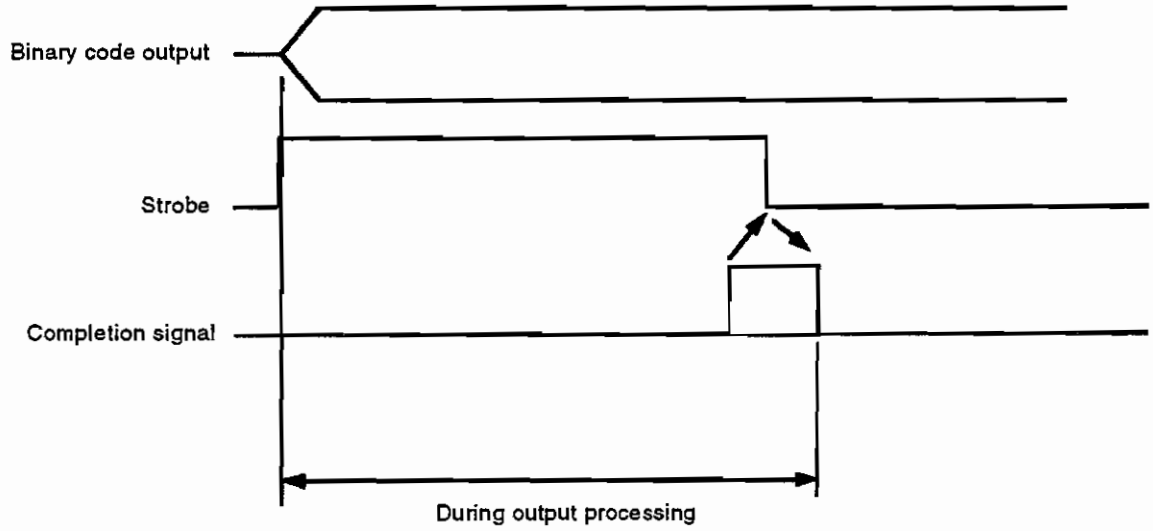
SPN(0. 1) Sets the travel speed to 0.1mm/min.
SPN(99999) Sets the travel speed to 99999.0mm/min.

Signal Output	Code Output

Format

OTD(output register, code)

Outputs the specified code in binary (8 bits).
There are four output registers 1 to 4 and four corresponding interfaces.
The binary code is output together with a strobe signal. The program moves to the next block when it receives the leading edge of a completion signal.
The strobe and completion signals are provided for each output register and can be output to more than one register at the same time. Note that these signals will not be output to the registers already under output processing until the completion of the output processing.



With the code output and independent output commands combined, up to four signal output commands may be given at the same time.
Also, with the code output and independent output commands combined, up to four signal output commands may be given together with the axis move command, point command or timer command.

Example

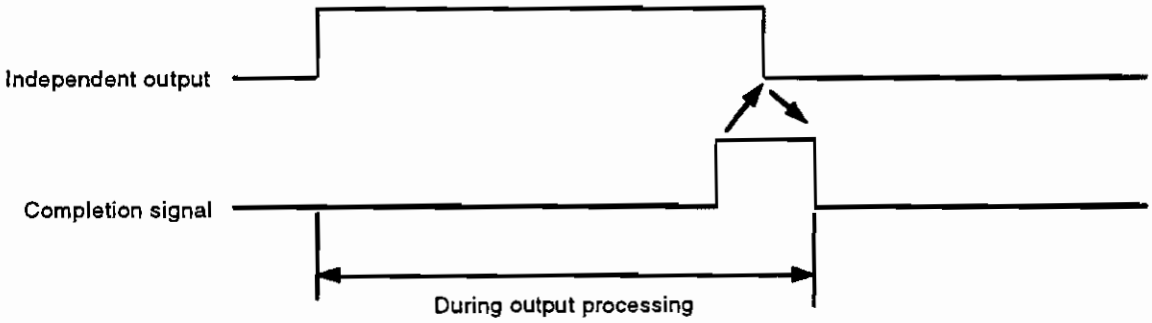
OTD(1, 10)	Outputs 0AH to output register 1.
TIM(10) OTD(4, 255)	Outputs FFH to output register 4 as soon as the timer starts timing.

Signal Output	Independent Output

Format

OTS(signal number)

Outputs an independent signal corresponding to the specified signal number.
There are a total of 32 signal numbers 1 to 32.
The program moves to the next block when it receives the leading edge of a completion signal after signal output.
Each completion signal has a signal number and two or more completion signals can be output at the same time. Note that any signals already under output processing are not output until the output processing is complete.



With the code output and independent output commands combined, up to four signal output commands may be given at the same time.
Also, with the code output and independent output commands combined, up to four signal output commands may be given together with the axis move command, point command or timer command.

Example

OTS(1)	Outputs an independent signal corresponding to signal number 1.
TIM(10) OTS(32)	Outputs an independent signal corresponding to signal number 32 as soon as the timer starts timing.

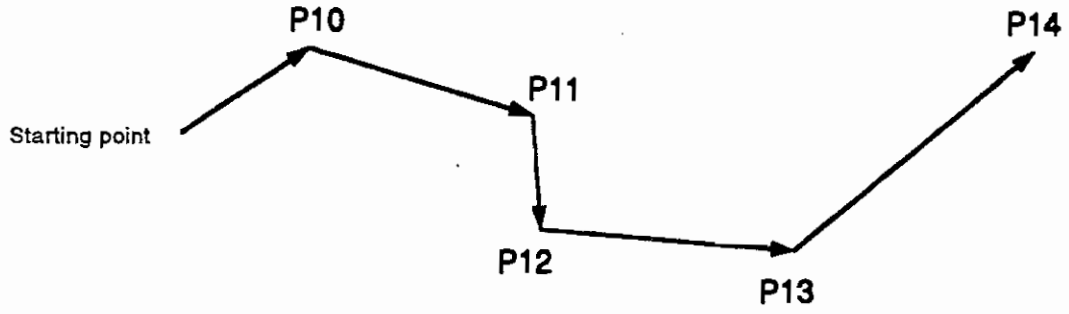
Point Command	Feed

Format

PMV(first point, last point)

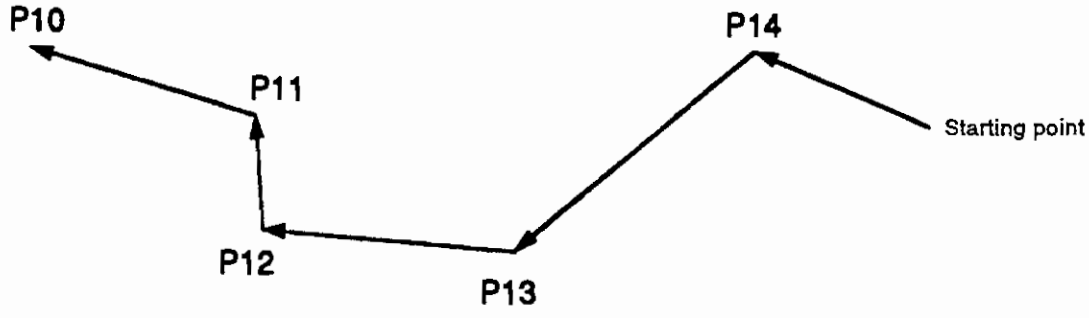
Executes positioning consecutively from the first point to the last point. The moving speed is as specified in the speed command.
When the last point is not specified, positioning is executed up to the first point.
When no points are specified, the axis motion is not executed.

Example
PMV(10, 14)



If the first point number is greater than the last point number, the axis moves backward through the point table.
If the position data of the points are set in incremental values, the signs of the incremental values are inverted (positive sign changed into negative or negative sign into positive) and the axis moves backward through the path it will move when the first point number is greater than the last point number.

Example
PMV(14, 10)



Point Command	Position Setting

Format

PST(point number, first axis position, second axis position, third axis position)

Sets positions to a point table.
This command does not affect the position data of the axes which are not specified in the operated axis command.
Whether the position data defined in position setting is incremental or absolute depends on the command of the position command unit.

Example

```
GRP(ARM1, HND1, TBL1)
AST
PST(1, 0. 1, 0. 2, 0. 3)        Sets 0. 1, 0. 2, 0. 3 to point table 1 in absolute values.
IST
PST(256, 10. 0, 20. 0, 30. 0)   Sets 10. 0, 20. 0, 30. 0 to point table 256 in incremental
                                values.
```

Point Command	Current Position Setting

Format

PRD(point number)

Sets to a point table the mechanical coordinate positions of all axes (up to three axes) currently selected as operated axes.
This command does not affect the position data of the axes which are not specified in the operated axis command.
Position data to be set is absolute.

Example

```
GRP(ARM1, HND1, TBL1)
PRD(1)          Sets the mechanical coordinate positions of ARM1, HND1
GRP(ARM2, ROL)  and TBL1 to point table 1 in absolute values.
PRD(256)        Sets the mechanical coordinate positions of ARM2 and ROL
                to point table 256 in absolute values.
```

Deceleration Check	Deceleration Check Mode

Format

PCM(mode)

You can change the deceleration checking method.
According to the deceleration check mode selected, you can complete deceleration after making an in-position check or specified deceleration check.
When the operated axis command (GRP) is given, the deceleration check mode completes deceleration when the specified deceleration is complete.
Give the PCK or PCF command to specify whether a deceleration completion check is made or not.
This mode is common to the operated axes (axes specified in GRP).

0: Deceleration completes on completion of the command deceleration.
1: Deceleration completes when the position deviation comes within the in-position range set in the parameter.

Any other value than the above will result in an error.

Example

PCM(0)	Sets the deceleration check system in which deceleration is completed on completion of command deceleration.
PCM(1)	Sets the deceleration check system in which deceleration is completed when the position deviation comes within the in-position range set in the parameter.

Note: Set the in-position range in the servo parameter 2224 (MP200) or 2220 (MP205).

Deceleration Check	Deceleration Check Command

Format

PCK
PCF

Specify PCK to execute a deceleration check on the following blocks.
Specify PCF to execute the next block without a deceleration check on the following blocks. When the operated axis command is given, a deceleration check is not executed until the PCK command is given.

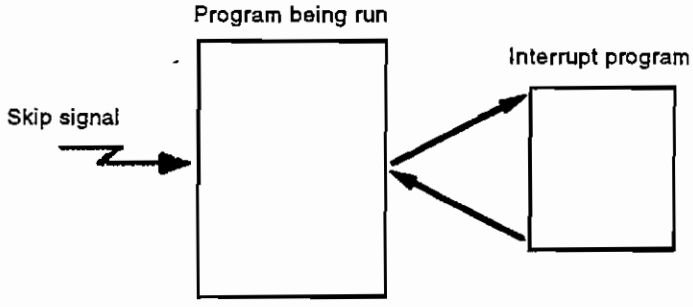
Example

```
PCK              Executes a deceleration check.
MOV(10, 20)              |
MOV(0, 0)                |
PCF              Execute the next block without a deceleration check.
MOV(100, 200)            |
MOV(-1, -2)              |
```

Format

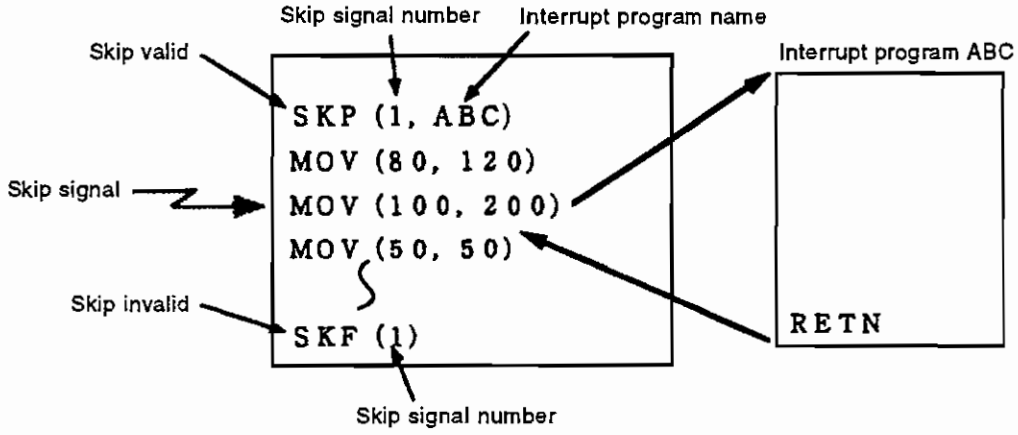
SKP(signal number, interrupt program name, skip mode)
SKF(signal number)

When the external skip signal switches on during program run, the program currently run is suspended (interpolation is suspended if during interpolation) and skip processing is executed.
The skip function is made valid only for the move command or timer command.



When an interrupt program is specified in the skip command, the entry of the skip signal causes the current program to suspend the processing of the current block, shift to the interrupt program, and return to the position of the block next to the one where interrupt had occurred.
When no interrupt program is specified, the entry of the skip signal causes the current program to suspend the processing of the current block and execute the next block.

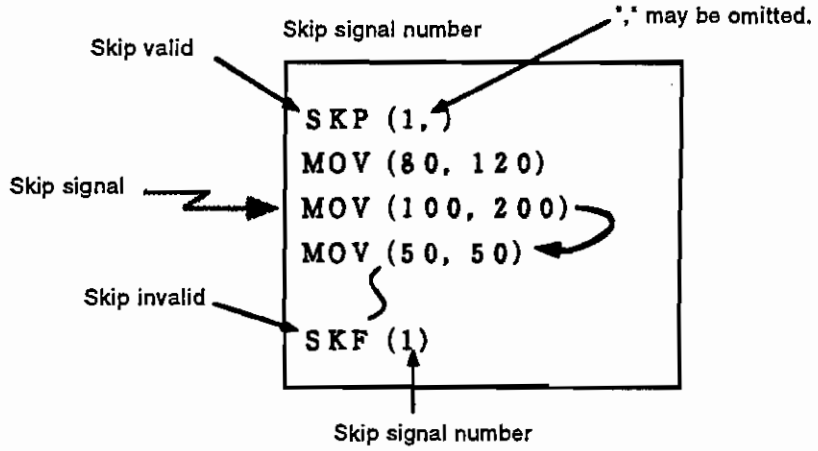
(a) When interrupt program is specified



Though the skip command (SKP, SKF) may be given in an interrupt program, a skip will not occur if the skip signal is switched on during execution of the interrupt program.
Though the skip command may be given in a torque limit skip interrupt program, a skip will not occur if the skip signal is switched on during execution of the interrupt program.

The operated axis, deceleration check mode and other commands used before a shift to the interrupt program are inherited to the interrupt program. When the operated axis, deceleration check mode and other commands are changed in the interrupt program, these commands used in the interrupt program are inherited to the original program.

(b) When interrupt program is not specified



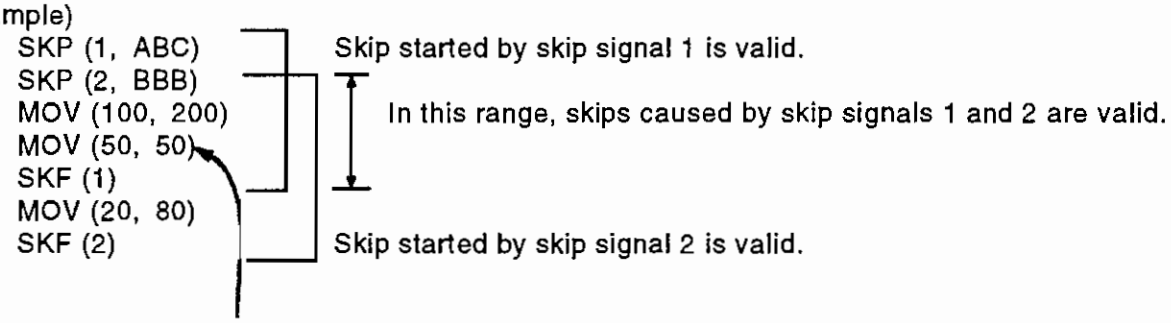
No skip signal number specified in the skip command (SKP) or skip invalid command (SKF) will result in an alarm.

Specify E as the skip mode to perform an edge trigger type skip or omit the skip mode to make a status type skip. Otherwise, an error will occur.
In the edge trigger type skip, skip processing is executed when the skip signal switches on, i.e. on its leading edge. In the status type skip, skip processing is executed while the skip signal is on.

The edge trigger type skip executes a skip on the leading edge of the skip signal. Therefore, if the program proceeds to the next block with the skip signal on, the next block is executed without a skip.
In the status type skip, if the skip signal remains on at the time of a skip to the next block, the program also skips the next block and continues skipping up to the block which has a skip invalid command. Also, if the skip signal remains on, the program calls an interrupt program in every block.

The skip instruction may be given to more than one skip signal.
When the skip instruction is given to more than one skip signal and two or more specified interrupt signals are entered simultaneously, skip processing started by the skip signal having a smaller number is executed (other skip processing is not executed). When the skip command and torque limit skip command are given together and the interrupt signal input and torque limit occur simultaneously, the skip command is made valid.

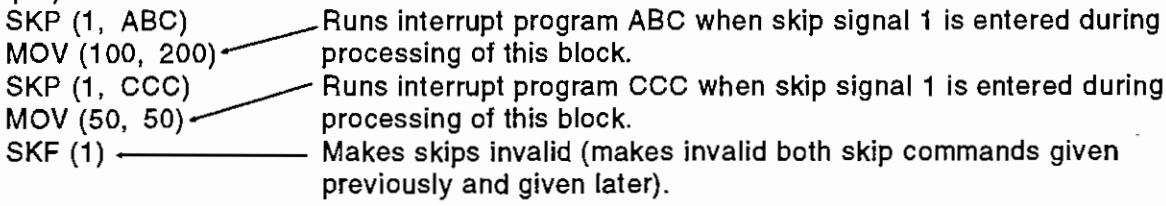
(Example)



When skip signals 1 and 2 are entered simultaneously during processing of this block, interrupt program ABC or skip processing corresponding to skip signal 1 is executed.

When the skip command corresponding to the skip signal which has already executed the skip instruction is given, the previously given skip command is made invalid and the skip command given later is made valid.

(Example)



A return to the reference point (ZRN command) executed in the skip mode will result in an error.
Changing the axes (GRP command) in the skip mode will reset the skip mode.

Format

TLM(first axis mode, second axis mode, third axis mode)

By decreasing the current limit value of a servo motor and increasing an excessive error width, a torque limit can be used for stop-on- contact workpiece positioning.
The torque limit command can be given independently to each axis.
Mode:

Specify 1 to make torque limit valid.
Specify 0 to make torque limit invalid.

Specifying any other value will result in an error.
For the axis without the mode command, the torque limit command remains unchanged.
The torque limit command given from a motion program is automatically reset when the operated axes are changed (GRP command) or operation ends.

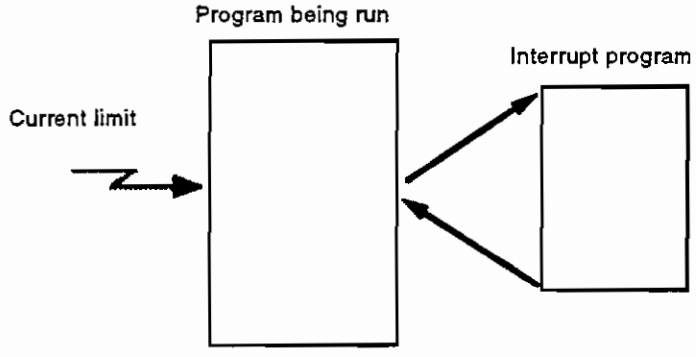
(Example)

```
GRP (X, Y)
TLM (1, 1) ←———— Makes torque limit on X and Y axes valid.
MOV (50, 30)
TLM (0, 0) ←———— Makes torque limit on X and Y axes invalid.
MOV (10, 0)
TLM (1, ) ←———— Makes torque limit only on X axis valid.
```

Format

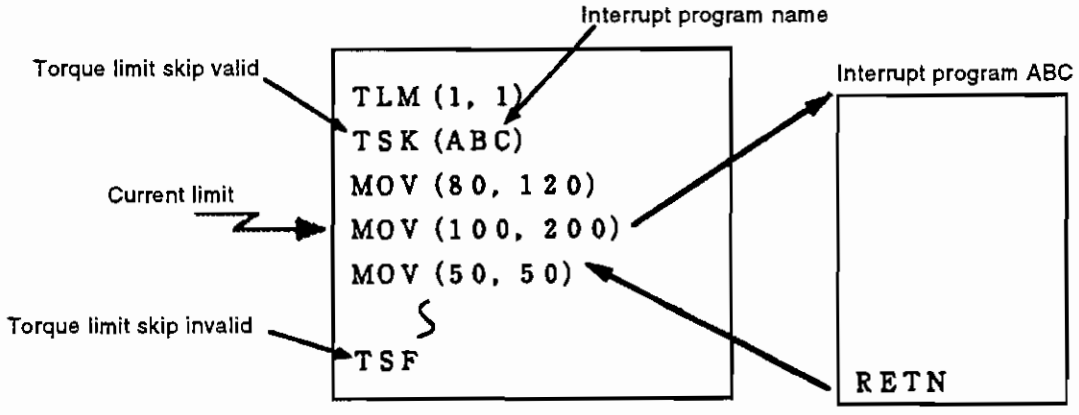
TSK(interrupt program name)
TSF

When a current limit is applied to the operated axes during program run, the program currently run is suspended (interpolation is suspended if during interpolation) and skip processing is executed.
The skip function is made valid only for the move command or timer command.
A torque limit should have been applied using the torque limit command, external torque limit signal or the like.



When an interrupt program is specified in the torque limit skip command, the application of the current limit causes the current program to suspend the processing of the current block, shift to the interrupt program, and return to the position of the block next to the one where interrupt had occurred.
When no interrupt program is specified, the application of the current limit causes the current program to suspend the processing of the current block and execute the next block.

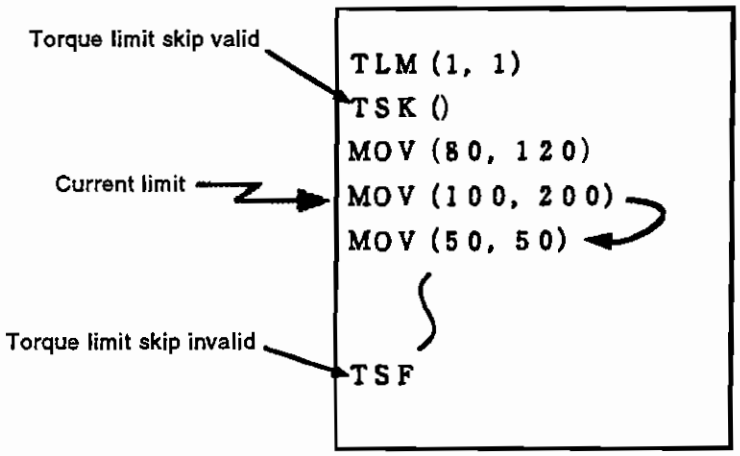
(a) When interrupt program is specified



Though the torque limit is valid as it is in the interrupt program, the current program will not shift to the interrupt program if the torque limit is applied.

Though the torque limit skip command (TSK, TSK) may be given in an interrupt program, a skip will not occur if the current limit is applied during execution of the interrupt program.
Though the torque limit skip command may be given in a skip command interrupt program, a skip will not occur if the current limit is applied during execution of the interrupt program.
The operated axis, deceleration check mode and other commands used before a shift to the interrupt program are inherited to the interrupt program. When the operated axis, deceleration check mode and other commands are changed in the interrupt program, these commands used in the interrupt program are inherited to the original program.

(b) When interrupt program is not specified



When the torque limit skip instruction already given is still valid and another torque limit skip instruction is given, the preceding torque limit skip instruction is made invalid and the latter torque limit skip instruction is made valid.
When the skip command and torque limit skip command are given together and the interrupt signal input and torque limit occur simultaneously, the skip command is made valid.

(Example)

```
TLM (1, 1)
TSK (ABC)
MOV (100, 200) ←——— Runs interrupt program ABC when current limit occurs during
TSK (CCC)           processing of this block.
MOV (50, 50)———————— Runs interrupt program CCC when current limit occurs during
TSF ←————————————— processing of this block.
                    Makes skips invalid (makes invalid both skip commands given
                    previously and given later).
```

A return to the reference points (ZRN command) executed in the torque skip mode will result in an error.
Changing the axes (GRP command) in the torque skip mode will reset the torque skip mode.

Format

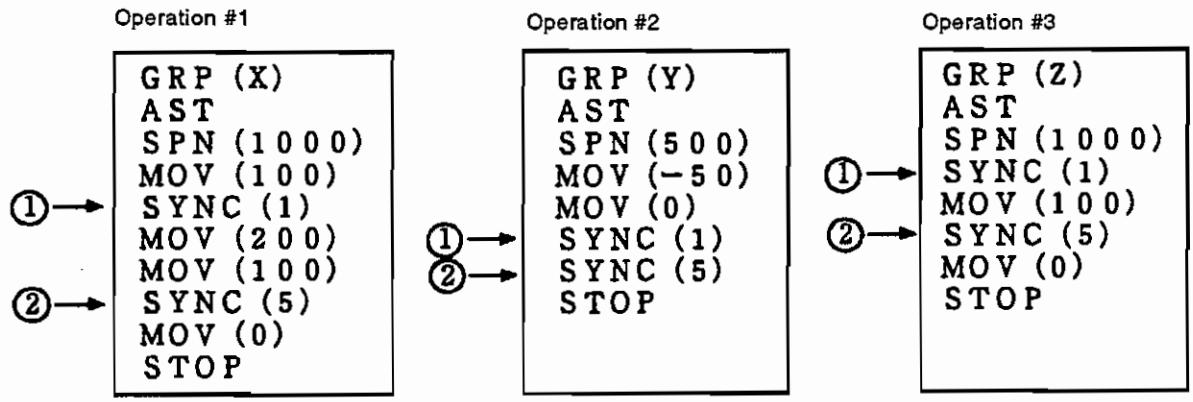
SYNC(synchronization number)

Causes operations started simultaneously to wait for each other.
When the SYNC commands of the same synchronization number are given in all operations started simultaneously, the program shifts to the next block.

Two or more synchronization positions may be specified. When there is a SYNC instruction in the program after a normal start, not after a simultaneous start, the program ignores the SYNC instruction and processes the next block.

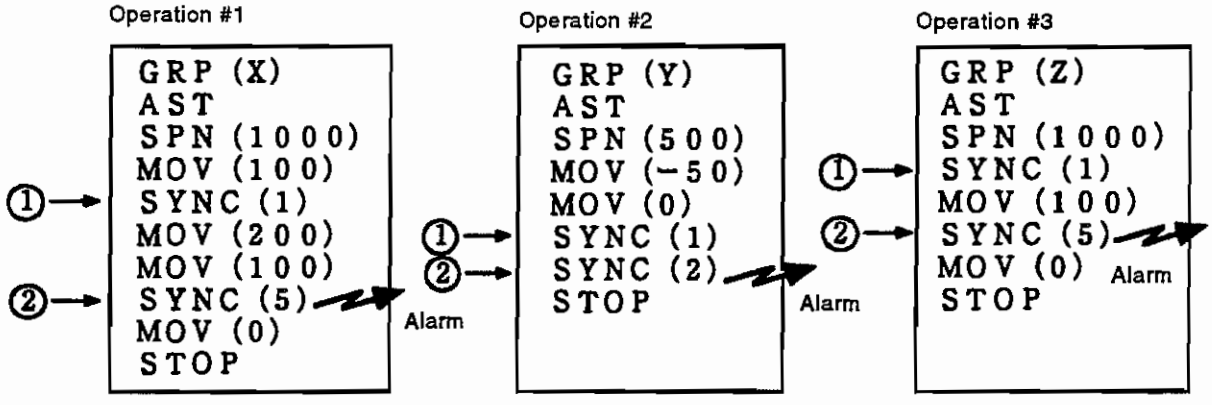
(Example)

The following is an example of synchronization positions specified for operations #1, #2 and #3 started simultaneously:
Operations are synchronized at 1 first. They are then synchronized at 2.



Specify a synchronization number when giving the synchronization instruction. If the synchronization number is omitted, an error will occur. When the synchronization number specified in one operation is different from the other operations started simultaneously, all operations started at the same time will result in an alarm (the alarm will occur when the program has been executed up to the synchronization command).

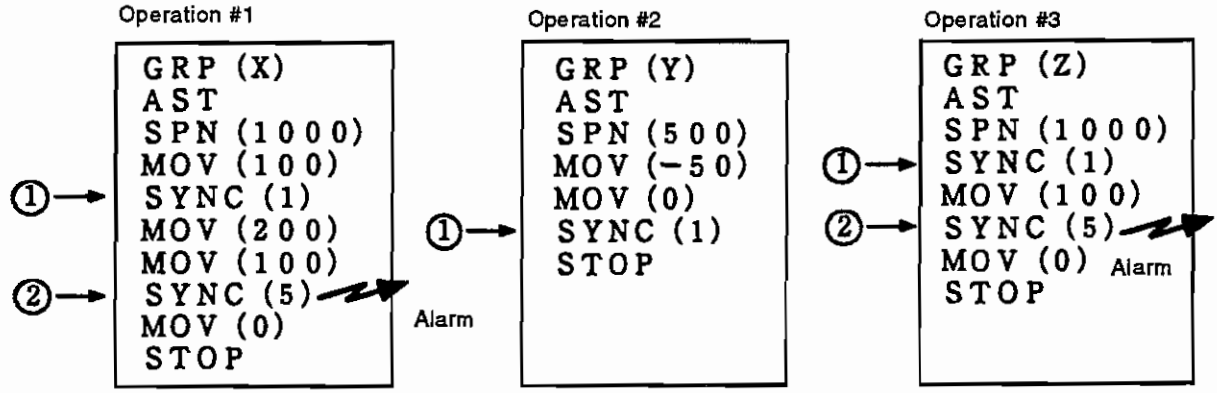
The following is an example of operations #1, #2 and #3 started simultaneously:
Operations are synchronized at 1. However, since the synchronization number of operation #2 is different from those of the others, all operations result in an alarm at 2.



When one operation does not have a synchronization instruction, the other operations started simultaneously result in an alarm at the end of that operation.

(Example)

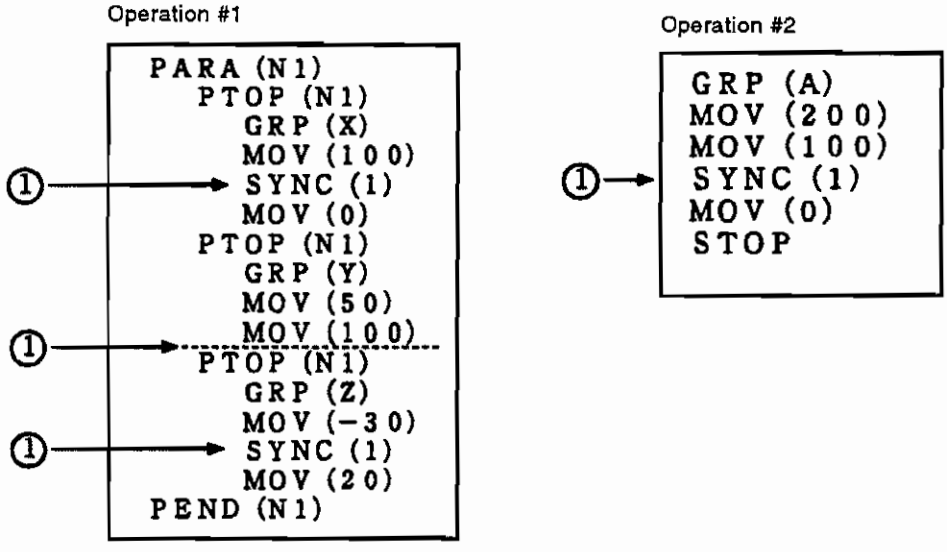
The following is an example of operations #1, #2 and #23 started simultaneously:
Operations are synchronized at 1. However, since operation #2 does not have the synchronization instruction of synchronization number 5, operations #1 and #3 result in an alarm at 2.



The synchronization instructions may also be given to parallel operations.
At this time, specify the synchronization instruction in each PTOP.
PTOP without the synchronization instruction is processed as if the synchronization instruction has been executed at the end of PTOP processing.

(Example)

The following is an example of operations #1 and #2 started simultaneously:
The operations are synchronized at 1.



Format

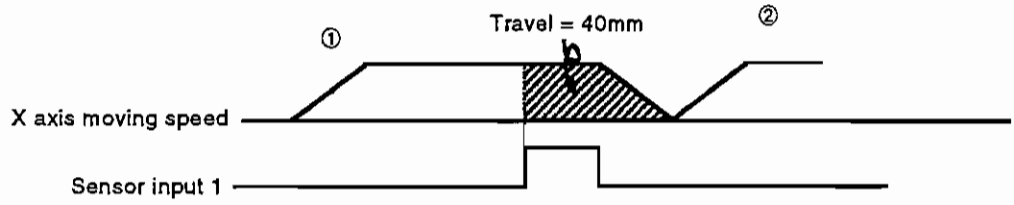
IMV(sensor input signal number, travel starting from sensor input)

Positions the axis by the specified travel, starting at the point where the sensor input signal having the specified number has switched on. When this command is executed, the axis starts moving at the feedrate specified in SPN or SPD. As soon as the sensor input signal of the number specified in argument 1 switches from off to on (on its leading edge), the axis is positioned to the point where it has moved by the travel specified in argument 2. At this time, the axis starts moving in the direction as indicated by the sign of the travel specified in argument 2.

Specify the sensor signal input number with any of 1 to 8 and the travel starting from the sensor input between -300000.000mm and 300000.000mm in an incremental value.

(Example)

	GRP (X)	Defines the X axis as an operated axis.
	PCM (1)	Sets the deceleration check system to in-position.
	PCK	Executes deceleration check.
	SPN (50)	Sets feedrate to 50mm/min.
①	IMV (1, 40)	Interrupt-positions the axis with the sensor signal number set to 1 and the travel starting from sensor input set to 40mm. As the travel starting from sensor input is a positive value, the axis moves in the positive direction.
②	MOV (500)	Moves the axis to the position of 500mm.



Note 1: When the interrupt positioning command is given, the program will start moving the next block before completion of positioning by the specified travel if the specified deceleration check or in-position deceleration check is not executed.

Note 2: The the interrupt positioning command cannot be executed during simultaneous operation (GRP():GRP()). An alarm (5E:09) will occur if the interrupt positioning command is executed during simultaneous operation.

Note 3: An alarm (5E:25) will occur if the instruction of the other motion program is given to the interrupt positioning command block. The signal output instructions such as code output (OTD) and independent output (OTS) cannot be given, either.

Note 4: The interrupt positioning command given while a skip or torque limit skip is valid or given in a skip interrupt program will result in an alarm (5E:2A).

Note 5: Interrupt positioning is executed only when the sensor input signal switches from off to on. If the sensor input signal specified in the motion program is already on at the start of the axis motion by the interrupt positioning command, interrupt positioning is executed when the signal switches on after it has switched off once.

Note 6: The interrupt positioning command is only valid for one operated axis (when only one axis has been specified in the GRP instruction). When there are two or more operated axes, an alarm (5E:2A) will occur.

Note 7: The position command unit (incremental/absolute) written before the interrupt positioning command block is held after the execution of the interrupt positioning command block.

Operation	Operation Instruction Pre-Read Processing

Format

QMCR(mode)

Switches on-off operation instruction pre-read processing which can rapidly process a successive variables read/write instruction or control instruction as a block identical to the next block.

Mode

0: Operation instruction pre-read processing is off.

1: Rapidly processes a successive variables read/write instruction as a block identical to the next block.

2: Rapidly processes a successive variables read/write instruction or control instruction as a block identical to the next block.

The control instruction described here indicates the IF statement, WHILE-WEND statement, GOTO statement, GOSUB statement, CALL statement or RETURN statement.

Note 1: When the program specified for operation instruction pre-read processing is operated on a single block basis, a single block stop is not effected in the variable read/write instruction or control instruction block.

Note 2: Operation instruction pre-read processing switches off when operation ends.

Note 3: Operation instruction pre-read processing is held when a subprogram is called.

Note 4: When the GRP command is given, operation instruction pre-read processing switches off.

Note 5: When reading a system variable, corresponding operation variable or common variable during operation instruction pre-read processing, data is read during execution of the move command in the block located several blocks before the read instruction block.

Example: When the counter value of the axis moved after axis movement is read, the end point of the movement block may not be read correctly. Before reading the counter value, switch off operation instruction pre- read processing and make a deceleration check.

Speed Reference Axis Designation	Designation and Reset of Speed Reference Axis

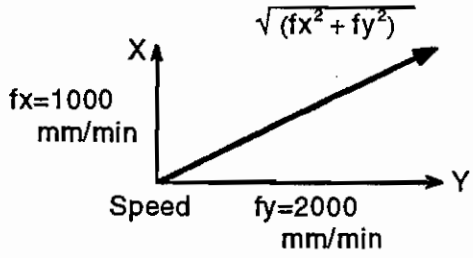
Format

SPA(axis name, axis name, axis name)
SPC

By using the SPA command to designate any axis as a speed reference axis, that axis can be moved at the speed specified in the speed command (SPN or SPD). For three-axis control, the composite speed of two axes may be specified.

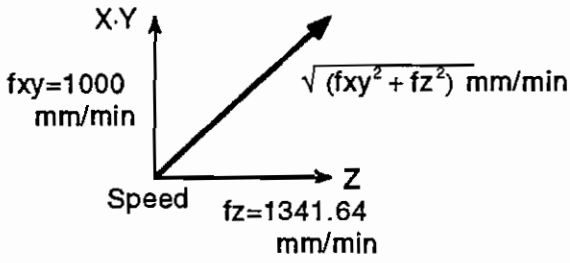
(Example 1)

```
GRP (X, Y)
SPA (X)..... Designates the speed
             reference axis.
SPN (1000)
MOV (10, 20)
```



(Example 2)

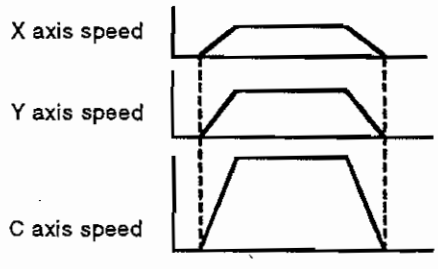
```
GRP (X, Y, Z)
SPA (X, Y)..... Designates the speed
                reference axes.
SPN (1000)
MOV (10, 20, 30)
```



Use the SPA command to designate the speed reference axis. Use the SPC command to reset the speed reference axis designation. Alternatively, the GRP command may be used to reset the speed reference axis designation.
The axis in the other group operated simultaneously may be designated as a speed reference axis. In this case, the speed for synchronization with the axis movement of the other group can be found to move the axes.

(Example 3)

```
GRP (X, Y)      :GRP (C)
SPN (1000)      :SPA (X)
MOV (10, 20)    :MOV (30)
```



Note 1: An alarm (5E:04, specified parameter invalid) will occur if the axis designated is not specified in GRP or the axis name is not designated.

Note 2: The speed reference axis is valid for the MOV or PMV command. When the axis move command is other than these commands, the axis moves at normal speed.

Note 3: Depending on the travel ratio, the speed of one axis may be higher than that of the other. In this case, the speed of the speed reference axis is changed automatically so that the speed of one axis may be a clamp speed.

Note 4: When using the SPA command to designate the axis of the other group in a simultaneous operation program, an alarm (5E:04, specified parameter invalid) will occur if the axis specified is in the group located on the right of the group having the SPA command in the corresponding block.

Note 5: If the travel of the axis specified in SPA is 0, an alarm (5E:2b, operation invalid in speed reference axis designation mode). Note that if the travel of the axis specified in SPA is 0, the alarm will not occur when the travel of the group having the SPA command is 0 (no motion).

Note 6: An alarm (5E:2b, operation invalid in speed reference axis designation mode) will occur if, in simultaneous operation, one group has a linear motion command and the other group has other than a linear motion command or circular command.

Note 7: If the result of speed calculation is not more than 0.1mm/min, the speed is clamped at 0.1mm/min.

4. EDITING PROGRAMS

Subprogram Control

Format

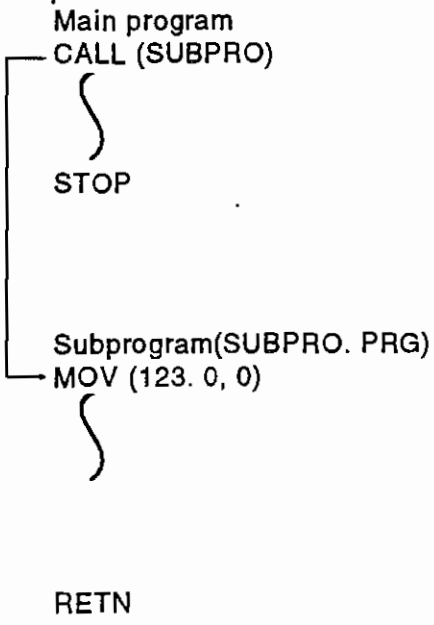
CALL(program name)

Calls a subprogram.
When execution returns from a subprogram to a main program, such data as:

Operated axes
Position command unit

specified in the subprogram are inherited unchanged to the main program.
Up to four subprograms may be called, with subprograms and subroutines combined. Calling more than four will result in an error. (Interrupt programs are counted out.)
A subprogram may also be called from an interrupt program.

Example



Calls a subprogram (SUBPRO.PRG).

Subprogram Control	Label

Format

***label name**

Specifies a label in a program.
A block must be headed by a label. In addition, a command cannot be written after a label. A label name is up to five alphanumeric characters long.

Example
```
*ABC
*SUB1
*12345
```

Subprogram Control	Subroutine Call

Format

GSUB(label name)

Searches a main program for a label and executes a subroutine from that position.
When execution returns from a subprogram to a main program, such data as:
- Operated axes
- Position command unit

specified in the subroutine are inherited unchanged to the main program.
Up to four subroutines may be called, with subprograms and subroutines combined. Calling more than four will result in an error. (Interrupt programs are counted out.)
A subroutine may also be called from an interrupt program.

Example

```
GSUB(ABC)          Executes a subroutine (ABC).
   (
   )
STOP
*ABC
MOV (-10. 0)
   (
   )
RETN
```

Subprogram Control	Return from Subroutine or Subprogram

Format

RETN

Used to return from a subprogram or subroutine.
When execution returns from a subprogram or subroutine, such command data as:

- Operated axes
- Position command unit

are inherited to the main program.

Example

```
Main program
CALL (SUBPRO)
MOV (0, 0, 0)
  (
  )
STOP


Subprogram(SUBPRO. PRG)
GRP (ARM2, HND2, TBL2)
AST
RETN                          Returns to the main program.
```

5 MOVING THE AXES SIMULTANEOUSLY OR IN PARALLEL

Simultaneous Operation
Parallel Operation

Simultaneous Operation	Start of Simultaneous Operation

Format

GRP(first axis name, second axis name, third axis name):GRP(first axis name, second axis name, third axis name)

By separating the operated axis commands by ":", simultaneous operation is executed until a new operated axis command or independent operation command is given.
Up to eight operated axis commands (one operated axis for one command) may be specified.
The axis move, point and other commands corresponding to the operated axes are to be separated by ":". These commands may not be specified for some operated axes. Note that some commands cannot be separated by ":". For the instructions that cannot be separated by ":", refer to the list at the end of this manual.
The "Simultaneous Command" section in the list indicates commands which can be separated by ":". The "Command in Simultaneous Command" section indicates commands which may follow the GRP():GRP() command.
The operation instructions and functions may be specified in a simultaneous command and separated by ":". The control instructions (GOTO, etc.) may be specified in a simultaneous command but cannot be separated by ":".

Example

(1)	(2)	(3)	
GRP(ARM1, HND1, TBL1)	:GRP(ARM2, HND2)	:GRP(ROL)	
MOV(10. 0, 20. 0, 0)	:MOV(1. 0, 2. 0)	:MOV(0. 1)	Operates specified axes (1), (2) and (3) simultaneously.
MOV(11. 0, 22. 0, 0)		:MOV(0. 2)	Operates specified axes (1) and (3) simultaneously.
TIM(10)	The timer may be specified in a simultaneous command but cannot be separated by ":".		

Parallel Operation	Start, Beginning and End of Parallel Operation

Format

```
PARA(mark)
  PTOP(mark)
PEND(mark)
```

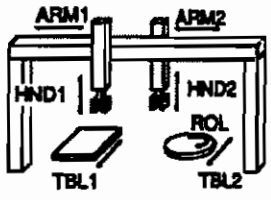
By giving the PARA command, two or more PTOP commands having the same mark as the one specified in PARA are executed in parallel independently of each other.
PTOPs executed in parallel must be located before PEND.
The parallel operation range is from one PTOP to the next PTOP or from PTOP to PEND.
PEND shifts to the next block when the programs run in parallel have all ended.
The marks are up to three alphanumeric characters long.

Example

```
PARA(ABC)              Executes areas (1), (2) and (3) in parallel.
  PTOP(ABC) ┐
            ├ (1)
  PTOP(ABC) ┤
            ├ (2)
  PTOP(ABC) ┤
            ├ (3)
            ┘
PEND(ABC)
```

The PTOP and PEND commands must be followed by the operated axis command.

Examples of Simultaneous Operation and Parallel Operation



Time of day

Axis Motion Timing Chart

ARM1 ARM2 HND1 HND2 TBL1 TBL2 ROL

```
GRP(ARM1,HND1)
SPD(5)
MOV(100,200)
```

Exercises interpolation control of ARM1 and HND1.

Speed 5-specified positioning

```
GRP(ARM2,ROL)
SPD(3)
PMV(4,6)
```

Exercises interpolation control of ARM2 and ROL.

```
GRP(ARM1,ARM2)
SPD(2)
MOV(110,200)
```

Exercises interpolation control of ARM1 and ARM2.

```
GRP(HND1) :GRP(HND2)
SPD(1)    :SPD(4)
MOV(150)  :MOV(200)
```

Executes HND1 and HND2 simultaneously line by line.

Three PTOP(A) commands are executed in parallel independently of each other.

```
PARA(A)
  PTOP(A)
    GRP(ARM1,ARM2)
    SPD(5)
    MOV(100,100)
    MOV(200,200)

  PTOP(A)
    GRP(HND1,HND2,TBL1)
    SPD(4)
    MOV(100,100,180)
    MOV(200,200,45)

  PTOP(A)
    GRP(TBL2,ROL)
    SPD(5)
    MOV(10,90)
    MOV(200,45)

PEND(A)
```

End of parallel processing

```
GRP(ARM1,HND1)
SPD(5)
MOV(100,200)
```

Exercises interpolation control of ARM1 and HND1.

6 PERFORMING OPERATION, ETC.

Operation Instructions
Functions
Control Instructions
Conditional Expressions

Operation Instruction	Variable Value Reference

Format

#variable number

In a motion program, you can use variables to perform various operations. You can also use the contents of variables in each command of the motion program.
There are the following types of variables:

#100 to #149: Local variables corresponding to operations
#500 to #599: Variables common to operations

Designating any variable not found in the specifications will result in an alarm.
In the variable command, # may be followed by:

Numerical value
Expression
Variable name

Example

```
#100
#(100+1)
#ABCDE
MOV (#500)
```

Operation Instruction	Variable Value Setting

Format

#variable number=expression

Set a value as a variable.
Any of the followings may be specified as an expression:

Numerical value
Variable
Function

The block must be headed by the variable value setting.
In addition, a command cannot be written after the variable value setting.
Note that the variable value setting may be specified in a conditional branch instruction.

Example

```
#100=0          Sets 0 to #100.
#101=#101+1     Sets #101+1 to #101.
#102=SIN(60)    Sets SIN(60) to #102.
```

Format

#variable number="#variable name"

Defines a variable name for a variable.
When a variable name is defined, you can use the variable name in the subsequent programs to set or refer to the variable.
The block must be headed by the variable name setting.
In addition, a command cannot be written after the variable name setting.
The variable name is up to seven alphanumeric characters long.
Any of the followings may be specified as a variable number:

- Numerical value
- Expression
- Variable name

When a numeral, alphanumeric characters beginning with a numeral, or a function name has been set as a variable name, that variable name cannot be used to set or refer to the variable.
When different variable names are defined for the same variable, the variable name defined last is made valid.
When #variable name is omitted, the variable name is made invalid, and the variable name cannot be used to set or refer to the variable until the variable name is defined next.

Example

#100="#ABCDE"	Sets "ABCDE" to the variable name of #100.
#101="#A123546"	Sets "A123456" to the variable name of #101.
#102=" "	Makes the variable name of #102 invalid.

Operation Instruction	Arithmetic Operation

Format

Expression Arithmetic operator Expression

Performs arithmetic operation of two numerical values.
Any of the followings may be specified as an expression:

- Numerical value
- Variable
- Function

The arithmetic operator types and operation types are as indicated on the right:

Operation Type	Arithmetic Operator	Example
Addition	+	#100+10
Subtraction	–	#110–#100
Multiplication	*	#120*10
Division	/	#120/5
Remainder	MOD	#100MOD3

Operation Instruction	Logical Operation

Format

Expression Logical operator Expression

Performs logical operation of two numerical values.
Any of the followings may be specified as an expression:

- Numerical value
- Variable
- Function

The logical operator types and operation types are as indicated on the right:

Operation Type	Logical Operator	Example
Logical add	OR	#100 OR #110
Exclusive logical add	XOR	#110 XOR #100
Logical product	AND	#120 AND 10

Operation Instruction	Absolute Value

Format

ABS(expression)

Provides the absolute value of a value specified as an expression.
Any of a numerical value, variable and function may be specified as an expression.

Example

#100=ABS(–10. 0) → #100=10. 000
(ABS(1–102))=1. 0 → #101=1. 000

Format

SIN(expression)

Provides a sine value corresponding to an expression value. Specify the expression unit in degrees. Any of a numerical value, variable and function may be specified as an expression.

Example

#100=SIN(45)	→	#100=0. 707
#101=SIN(–270)	→	#101=1. 000

Format

COS(expression)

Provides a cosine value corresponding to an expression value. Specify the expression unit in degrees. Any of a numerical value, variable and function may be specified as an expression.

Example

#100=COS(60)	→	#100=0. 500
#101=COS(180)	→	#101=–1. 000

Format

TAN(expression)

Provides a tangent value corresponding to an expression value. Specify the expression unit in degrees. Any of a numerical value, variable and function may be specified as an expression.

Example

#100=TAN(45)	→	#100=1. 000
#101=TAN(–125)	→	#101=1. 428

Format

ATAN(expression)

Provides an arctangent value corresponding to an expression value. The unit of the value provided is in degrees. Any of a numerical value, variable and function may be specified as an expression.

Example

```
#100=ATAN(1)          →   #100=45. 000
#101=ATAN(1.732)      →   #101=59. 999
```

Format

ACOS(expression)

Provides an arccosine value corresponding to an expression value. The unit of the value provided is in degrees. Any of a numerical value, variable and function may be specified as an expression.

Example

```
#100=ACOS(0. 5)       →   #100=60. 000
#101=ACOS(0)          →   #101=90. 000
```

Format

SQR(expression)

Provides the square root of a value specified as an expression. Any of a numerical value, variable and function may be specified as an expression.

Example

```
#100=SQR(3*3+4*4)     →   #100=5. 000
#101=SQR(5.76)        →   #101=2. 400
```

Format

BIN(expression)

Converts an expression value (BCD code) into a binary value (binary conversion). After its fraction is discarded, the expression value is converted into binary. A negative expression value will result in an error.
Any of a numerical value, variable and function may be specified as an expression.

Example

#100=BIN(16)	→	#100=10. 000
#101=BIN(512)	→	#101=200. 000
#100=BIN(64.5)	→	#102=40. 000

Format

BCD(expression)

Converts an expression value into a BCD code (BCD conversion). After its fraction is discarded, the expression value is converted into BCD. If the expression value is negative, its absolute value is converted into BCD.
Any of a numerical value, variable and function may be specified as an expression.

Example

#100=BCD(10)	→	#100=16. 000
#101=BCD(100)	→	#101=256. 000
#100=BCD(10.5)	→	#102=16. 000

Format

RND(expression), ROUND(expression)

Provides a value by rounding off the fraction of an expression value. Any of a numerical value, variable and function may be specified as an expression.

Example

#100=RND(1.4)	→	#100=1. 000
#101=RND(20. 5)	→	#101=21. 000
#100=ROUND(–300. 499)	→	#102=–300. 000

Format

FIX(expression)

Provides a value by eliminating the fraction of an expression value. Any of a numerical value, variable and function may be specified as an expression.

Example

```
#100=FIX(10. 1)      →  #100=10. 000
#101=FIX(20. 8)      →  #101=20. 000
```

Format

FUP(expression)

Provides a value by rounding up the fraction of an expression value. Any of a numerical value, variable and function may be specified as an expression.

Example

```
#100=FUP(10. 8)      →  #100=11. 000
#101=FUP(-10. 1)     →  #101=-11. 000
```

Format

LOG(expression)

Provides the natural logarithm (base "e" logarithm) of an expression as a value. Any of a numerical value, variable and function may be specified as an expression.

Example

```
#100=LOG(5)          →  #100=1. 609
#101=LOG(100)        →  #101=4. 605
```

Format

EXP(expression)

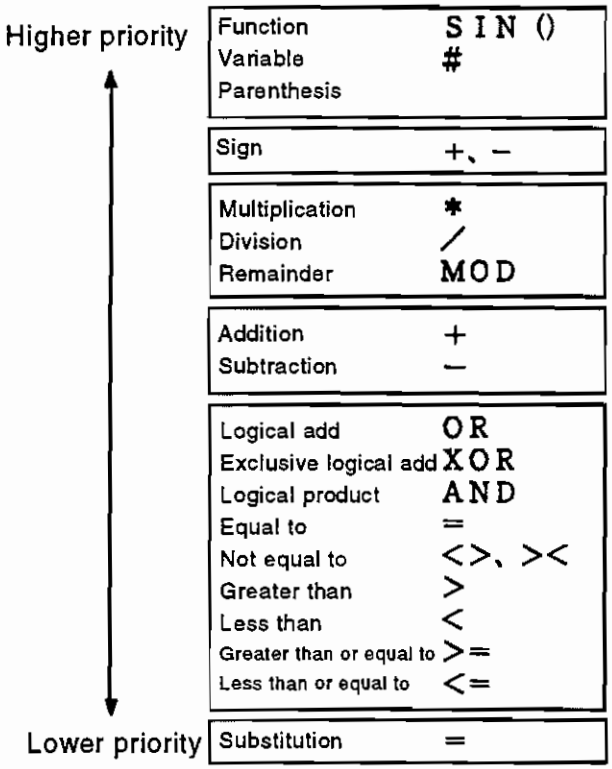
Provides the result of exponential operation on the "e" (base of a natural logarithm) of an expression as a value. Any of a numerical value, variable and function may be specified as an expression.

Example

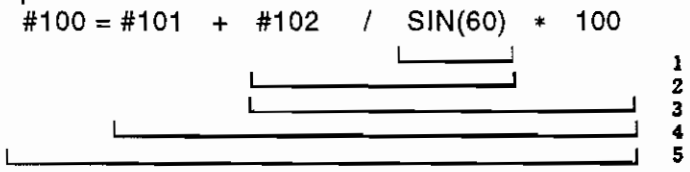
```
#100=EXP(2)          →  #100=7. 389
#101=EXP(-1)         →  #101=0. 368
```

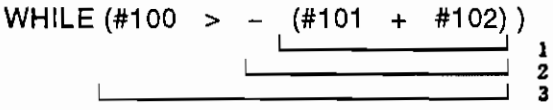
Operation priority

The operators and functions available for motion programs have the following priority.
Operation is executed in order of higher priority. For operators and functions on the same level, operation is performed in order of description.



Example





Format

GOTO(label name)

Searches for a label in a program and branches to the line of the label unconditionally.
When a branch is taken, data specified before the branch, e.g.:

Operated axes
Position command unit

are inherited unchanged.
GOTO may only be used to branch to a label in the same program.
An error will occur if the specified label name does not exist.

Example

```
GOTO(ABC)          Branches to label (ABC).
    ⌇
*ABC
MOV (1. 0)
    ⌇
```

Format

IF(conditional expression) THEN statement ELSE statement

Controls the execution of a program according to the evaluation result of a conditional expression. If the evaluation result of the conditional expression is true (other than 0), the statement following THEN is executed. If the evaluation result of the conditional expression is false (0), the statement following ELSE is executed. ELSE and its subsequent statement may be omitted. When they are omitted, the next line will be executed.

Any of the followings may be used as a statement:

- Variable substitution
- IF statement
- GOTO statement
- GSUB statement

Example

```
#100=0
IF(#100=0)THEN #110=1 ELSE #110=2          Executes #110=1 since #100=0.
IF(#100=0)THEN IF(#110=0)THEN GOTO(ABC)    Executes the next line since
                                           #110=1 though #100=0.
IF(#110=1)THEN GOTO(ABC)                   Branches to ABC since #110=1.
*ABC
MOV (1. 0)
```

Control Instruction	Repeat

Format

```
WHILE(conditional expression)
{
WEND
```

While the conditional expression is true (other than 0), a set of statements between WHILE and WEND is repeated. When the conditional expression becomes false (0), execution shifts to the line after WEND.
If the conditional expression is false (0) from the beginning, the set of statements between WHILE and WEND is never executed.

You can place another WHILE-WEND sequence within the WHILE and WEND sequence (called nesting). WHILE can be nested up to two levels. WHILE- WEND must be on a one-for-one basis. A program which causes a jump from the outside into the WHILE-WEND sequence or, reversely, from the WHILE- WEND sequence to the outside cannot be guaranteed to perform proper operation.

Example

```
#100=0
WHILE(#100<10)        Repeats statements up to WEND 10 times while #100 is less than 10.
{
  #100=#100+1
WEND
```

Format

Expression Relational operator Expression

Compares two numerical values.
The result of comparison is indicated by true (1) or false (0).
A conditional branch, repeat or other instruction is used as a conditional expression.

Any of the followings may be specified as an expression:

- Numerical value
- Variable
- Function

Relational operator types and operation types are as follows:

Operation	Relational Operator	Example
Equal to	=	#100=10
Not equal to	<>	#110<>#100
	><	#110><#100
Greater than	>	#120>10
Less than	<	#120<5
Greater than or equal to	>=	#500>=10
	=>	#500=>10
Less than or equal to	<=	#500<=5
	=<	#500=<5

Note that the relational operator = (equal to) is also used to set a variable value.

7 SYSTEM VARIABLES

Read/Write of Various Data

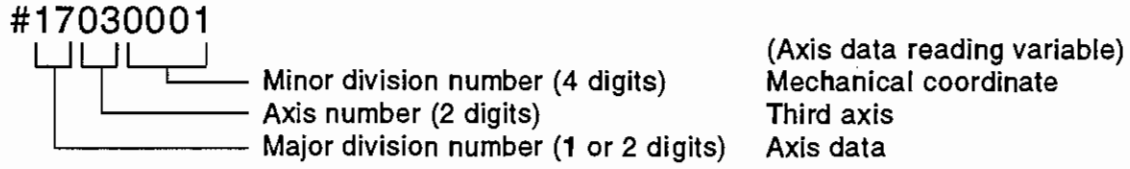
7.1 What are System Variables?

Using system variables, you can access various registers in the built-in PLC, parameters, etc. as direct variables from a motion program.

For example, the acceleration/deceleration times of the first to third axes are set to 200msec as indicated below:

```
#34012006=200
#34022006=200
#34032006=200
```

The variable number of a system variable is 7 or 8 digits in length and consists of a major division number, axis number and minor division number:



System variables are divided into accessible variables and read-only variables. If you attempt to assign data to a read-only system variable, a variable number invalid error will occur.
Also, if the data written is outside the write range, a command value excess error will occur.

Write to read-only variable
#11030001 = 1223.456 ——→ 5E21 error occurs.
When write range is invalid
#34012006 = 50000 ——→ 5E04 error occurs.
(0 to 30000 for first acceleration/deceleration time)

Like the variables corresponding to operations and variables common to operations, system variables may be used as the arguments of the instructions and functions in motion programs.
Further, variables corresponding to operations and variables common to operations can be assigned to system variables.
(Example)

```
#34012006=#500+1
#100=#34012006*2
```

Also, variable (system variable, corresponding-to-operation variable, common-to-operation variable) numbers may be specified by variables.
(Example)

```
#100=#34012006
#(#100)=#500+1
```

For the details of the system variables, refer to the system variable list in Section 7.2.

7.2 System Variable List

(1) Axis data

Major Division Number	Axis Number	Minor Division Number	Minor Division Item	Description	Remarks
17	01 to 08	0001	Mechanical coordinate	Reads the axis data of the corresponding axis in mm. Example: To read the mechanical coordinate of the third axis. #100=#17030001 When the mechanical coordinate of the third axis is 12.345mm, #100 will be 12.345.	Read-only
		0002	Feedback position		
		0003	Acceleration/ deceleration droop		
		0004	Position droop		
18	01 to 08	0001	Mechanical coordinate	Reads the axis data of the corresponding axis in μm. Example: To read the mechanical coordinate of the third axis. #100=#17030001 When the mechanical coordinate of the third axis is 12.345mm, #100 will be 12345.	Read-only
		0002	Feedback position		
		0003	Acceleration/ deceleration droop		
		0004	Position droop		

Major Division Number	Axis Number	Minor Division Number	Minor Division Item	Description	Remarks
17	01 to 08	0005	Block starting point	For operation instruction pre-reading, reads in mm the starting point/ end point position at a time when this system variable is read. Alternatively, when operation instruction pre-reading is off, reads in mm the starting point/end point position of a move command located immediately before. If the axis is not operating, the read value will be 0.	Read-only
		0006	Block end point	Example: To read in mm the move command starting point position of the third axis. QMCR(0) MOV(100,20,30) #100=#17030005	
18	01 to 08	0005	Block starting point	As in major division 17, reads in m the starting point/end point position of a move command. If the axis is not operating, the read value will be 0.	
		0006	Block end point	Example: To read in m the move command end point position of the third axis. QMCR(0) MOV(100,20,30) #100=#17030006	

(2) Parameters

Major Division Number	Axis Number	Minor Division Number	Minor Division Item	Description	Remarks
33	00	1000 ~	Common parameter	Allows parameters to be accessed. Specify the parameter number as the minor division number. For parameter data, refer to the instruction manual. Example: To change the third manual feedrate. #33001033 = 1500	Accessible
34	01 to 08	2000 ~	Axis parameter		
35	01 to 08	2200 ~	Servo parameter		
36	00	9000 ~	Sampling parameter		

Note 1: If the data written is outside the parameter setting range, a 5E04 error will occur.

Note 2: If you have written data in the parameter which requires power to be switched on-off after parameter changing, the data written is not made valid until power is switched on again.

Note 3: There are parameters of character string, decimal data and hexadecimal data. (Refer to the parameter explanation in the instruction manual.)
Decimal or hexadecimal data is accessed as decimal data.
Character string data is accessed as decimal data which has been converted from the four least significant ASCII code characters of a character string.
Example: When the file name of DIO program-specified code 1 is ABC
#100=33001101
If #100 = #33001101, 4276803 (414243HEX) is assigned to #100.

(3) Point data

Major Division Number	Axis Number	Minor Division Number	Minor Division Item	Description	Remarks
49	01 to 08	0001 to 0256	Point data	Accesses the position data of the corresponding point in mm. Example: To read the data of point number 143 of the third axis. #100=#49030143 When the point data is 12.345mm, #100 will be 12.345.	Accessible
50	01 to 08	0001 to 0256	Mode of point data	Accesses the mode of the position data of the corresponding point. The data to be accessed will be: 65 when the point data is an absolute value; 73 when the point data is an incremental value; or 78 when the point data is invalid.	
51	01 to 08	0001 to 0256	Point data	Accesses the position data of the corresponding point in m. Example: To read the data of point number 143 of the third axis. #100=#49030143 When the point data is 12.345mm, #100 will be 12345.	

Note: Write the point data as absolute value data whether the position command unit is the AST or IST instruction. Use the PST instruction to determine the mode of the point data according to the AST/IST instruction.

(4) Motion program variables
You can use the system variables to access the variables of the other operation numbers.

Major Division Number	Axis Number	Minor Division Number	Minor Division Item	Description	Remarks
57	01 to 08 (Indicate operation numbers)	0100 to 0149 (Indicate variable numbers)	Corresponding-to-operation variable	Allows the corresponding-to-operation number variables (#100 to #149) to be accessed. Example: To assign the current operation number to variable #120 of operation number 5. #57050120=#120	Accessible

(4) PLC data

Major Division Number	Axis Number	Minor Division Number	Minor Division Item	Description	Remarks
65	00	0000 to 1279	X device	Accesses a bit device bit-by-bit. Specify the X/Y device number in decimal. Example: To switch on M123. #67000123 = 1 Assign X20 to #100. #100 = #65000032	Accessible
66	00	0000 to 1407	Y device		
67	00	0000 to 5119	M device		
81	00	0000 to 1272	X device	Accesses bit devices byte-by-byte. Specify the X/Y device number in decimal. The most significant bit of a byte is handled as a sign. Example: To read M120-M127 byte-by-byte. M120 1 M122 1 M124 1 M126 0 M121 1 M123 1 M125 1 M127 0 In the following case: if #501 = #83000120, #501 will be 55 (37HEX).	Read-only
82	00	0000 to 1400	Y device		
83	00	0000 to 5112	M device		
74	00	0000 to 1023	D register	Accesses a D or R register word-by-word. (The most significant bit of a word is handled as a sign.) Example: To write 3455 (D7F HEX) to D456. #74000456 = 3455	Accessible
75	00	0000 to 8191	R register		
90	00	0000 to 1022	D register	Accesses a D or R register on a double word basis. (Refer to Note 3.) (The most significant bit of a double word is handled as a sign.) Example: To write 1180721 (120431 HEX) to R1000. If #91001000 = 1180721, R1000 431 HEX R1001 12 HEX.	
91	00	0000 to 8190	R register		

Note 1: Access is executed asynchronously with a ladder.

Note 2: When writing data to a remote I/O unit (Y0 to YFF), the data is output to the remote I/O at the end of one scan in a ladder.
When reading data from a remote I/O unit (X0 to XFF), the data input from the remote I/O is read at the beginning of a scan in a ladder.

Note 3: When accessing a D or R register on a double word basis, the lower and upper words may not be matched depending on the timing of executing the ladder and motion program. When accessing data in double word, take proper measures, e.g. provide a flag to make sure that data is not being written and read data when that confirmation flag is off.

7.3 Examples of Using the System Variables

(1) Example of reading the D register of a PLC
When the X axis positioning point has been set to D register 1000 of a PLC in 4-digit BCD code (10μm increments), the motion program used to read the contents of D1000 and position the axis is as follows:

```
GRP (X)
#500=BIN (#74001000)/100 ←—— Performs binary conversion to provide the data in mm.
SPN (1500)
MOV (#500)               ←—— Positions the axis at 1500mm/min feedrate.
```

(2) Example of monitoring the signal status
The motion program which waits for M100 of a PLC to switch on is as follows:

```
*LOP
IF (#67000100=0) THEN GOTO (LOP)
```

Motion Program Function List (1)

Major Division	Division	Function	Language	Argument 1	Argument 2	Argument 3	Argument 4	Simultaneous Command	Command in Simultaneous Command	Refer to Page
Moving the axes	Operated axes	Operated axis command	GRP	First axis name	Second axis name	Third axis name	–	Possible	Possible	2-2
	Axis move command	Positioning	MOV	First axis position	Second axis position	Third axis position	–	Possible	Possible	2-3
		Reference point return	ZRN	–	–	–	–	Impossible	Impossible	2-3
	Speed command	Parameter-specified speed	SPD	Parameter number	–	–	–	Possible	Possible	2-4
	Ending the program	Ending the program	STOP	–	–	–	–	Impossible	Possible	2-4
Various control functions	Position command unit	Absolute value command	AST	–	–	–	–	Possible	Possible	3-2
		Incremental value command	IST	–	–	–	–	Possible	Possible	3-2
	Axis move command	Circular interpolation (clockwise)	MCW	First axis position	Second axis position	Radius	–	Possible	Possible	3-3
		Circular interpolation (counterclockwise)	MCC	First axis position	Second axis position	Redius	–	Possible	Possible	3-4
	Timer	Timer	TIM	Time	–	–	–	Impossible	Possible	3-5
	Speed commend	Feedrate command	SPN	Feedrate	–	–	–	Possible	Possible	3-5
	External signal output	Data output	OTD	Code number	Code	–	–	Possible	Possible	3-6
		Signal output	OTS	Signal number	–	–	–	Possible	Possible	3-7
	Point command	Positioning	PMV	First point number	Last point number	–	–	Possible	Possible	3-8
		Position setting	PST	Point number	First axis position	Second axis position	Third axis position	Possible	Possible	3-9
		Current position read	PRD	Point number	–	–	–	Possible	Possible	3-9
	Deceleration check	Deceleration check mode	PCM	Mode	–	–	–	Possible	Possible	3-10
		Deceleration check command	PCK	–	–	–	–	Possible	Possible	3-10
		Decaleration check command off	PCF	–	–	–	–	Possible	Possible	3-10
Editing programs	Subprogram control	Subprogram control	CALL	Program	–	–	–	Impossible	Possible	4-2
		Subroutine call	GSUB	Label	–	–	–	Impossible	Possible	4-3
		Return from subprogram	RETN	–	–	–	–	Impossible	Possible	4-4
Moving the axes simultaneously or in parallel	Simultaneous operation	Simultaneous operation	GRP :GRP	First axis position	Second axis position	Third axis position	–	Possible	Possible	5-2
	Parallel operation	Start of parallel operation	PARA	Mark	–	–	–	Impossible	Possible	5-3
		Beginning of parallel operation	PTOP	Mark	–	–	–	Impossible	Possible	5-3
		End of parallel operation	PEND	Mark	–	–	–	Impossible	Possible	5-3

Motion Program Function List (2)

Major Division	Division	Function	Language	Refer to Page
Performing operation, etc.	Operation instruction	Variable command	#	6-2
		Definition, replacement	=	6-2
		Addition	+	6-4
		Subtraction	-	6-4
		Multiplication	*	6-4
		Division	/	6-4
		Remainder	MOD	6-4
		Logical add	OR	6-4
		Exclusive logical add	XOR	6-4
		Logical product	AND	6-4
		Absolute value	ABS()	6-4
	Function	Sine	SIN()	6-5
		Cosine	COS()	6-5
		Tangent	TAN()	6-5
		Arctangent	ATAN()	6-6
		Arccosine	ACOS()	6-6
		Square rot	SQR()	6-6
		Decimal to haxadecimal conversion	BIN()	6-7
		Hexadecimal to decimal conversion	BCD()	6-7
		Rounding off	RND(), ROUND()	6-7
		Truncation	FIX()	6-8
		Rounding up	FUP()	6-8
		Natural logarithm	LOG()	6-8
		Exponent	EXP()	6-8
		Priority of functions and expressions		6-9
	Control instruction	Unconditional branch	GOTO label	6-10
		Branch	IF conditional expression THEN ELSE	6-11
		Repeat	WHILE conditional expression - WEND	6-12
	Conditional expression	Equal to	=	6-13
		Not equal to	<>, ><	6-13
		Greater than	>	6-13
		Less than	<	6-13
		Greater than or equal to	>=, =>	6-13
		Less than or equal to	<=, =<	6-13

REVISION HISTORY

Sub Number	Revision Date	Revisions
*	Nov, 1996	First edition