HITACHI

SOFTWARE MANUAL

PROGRAMMING

HI-FLOW For Windows®

S10V

Programmable Controller

SVE-3-122（E）

SOFTWARE MANUAL

PROGRAMMING

HI-FLOW For Windows®

First Edition, July 2004, SVE-3-122(B) (out of print)
Second Edition, April 2005, SVE-3-122(C) (out of print)
Third Edition, October 2006, SVE-3-122(D) (out of print)
Fourth Edition, February 2009, SVE-3-122(E)

Printed in Japan.

BI-NS-MM<IC-IC> (FL-MW20)

SAFETY PRECAUTIONS

- Read this manual thoroughly and follow all the safety precautions and instructions given in this manual before operations such as system configuration and program creation.
- Keep this manual handy so that you can refer to it any time you want.
- If you have any question concerning any part of this manual, contact your nearest Hitachi branch office or service engineer.
- Hitachi will not be responsible for any accident or failure resulting from your operation in any manner not described in this manual.
- Hitachi will not be responsible for any accident or failure resulting from modification of software provided by Hitachi.
- Hitachi will not be responsible for reliability of software not provided by Hitachi.
- Make it a rule to back up every file. Any trouble on the file unit, power failure during file access or incorrect operation may destroy some of the files you have stored. To prevent data destruction and loss, make file backup a routine task.
- Furnish protective circuits externally and make a system design in a way that ensures safety in system operations and provides adequate safeguards to prevent personal injury and death and serious property damage even if the product should become faulty or malfunction or if an employed program is defective.
- If an emergency stop circuit, interlock circuit, or similar circuit is to be formulated, it must be positioned external to the programmable controller. If you do not observe this precaution, equipment damage or accident may occur when this programmable controller becomes defective.
- Before changing the program, generating a forced output, or performing the RUN, STOP, or like procedure during an operation, thoroughly verify the safety because the use of an incorrect procedure may cause equipment damage or other accident.

This manual provides information for the following program product:

<Program product>
S-7895-03, S10V HI-FLOW System, 02-05

The following change has been made to the S10V HI-FLOW System (02-02).

Description of added changes	Corresponding chapter, section, subsection, or supplement
The OPTET module is newly added for Ethernet communication.	Subsections 5.7.1 thru 5.7.3.
Motion control instructions are newly supported.	Sections 2.2, 2.4, 3.2, 4.16, and chapter 6.
A description of system configuration update procedures for the MP2300H controller is newly added.	Section 6.3 and supplement 6.

(SVE-3-122(D))

The following change has been made to the S10V HI-FLOW System (02-04).

Description of added changes	Corresponding chapter, section, subsection, or supplement
Non-synchronous process end is newly supported.	Sections 2.2, 3.1, 3.2, 4.1, 4.17, and supplement 4.

(SVE-3-122(D))

The following change has been made to the S10V HI-FLOW System (02-05).

Description of added changes	Corresponding chapter, section, subsection, or supplement
A feature called motion communication-task automatic transmission is newly supported.	Section 6.3.

(SVE-3-122(E))

In addition to the above changes, all the unclear descriptions and typographical errors found are also corrected without prior notice.

Revision record

Revision No.	Revision record (revision details and reason for revision)	Month, Year	Remarks
D	The OPTET module is newly added for Ethernet communication.	October 2006	S10V HI-FLOW System (01-07) or later
D	Motion control instructions are newly supported.	October 2006	S10V HI-FLOW System (02-01) or later
D	A description of system configuration update procedures for the MP2300H controller is newly added.	October 2006	S10V HI-FLOW System (02-02) or later
E	Non-synchronous process end is newly supported.	February 2009	S10V HI-FLOW System (02-04) or later
	A feature called motion communication-task automatic transmission is newly supported.		S10V HI-FLOW System (02-05) or later

PREFACE

HI-FLOW, a new programming language for the Hitachi Programmable Controller, is a language in the form of a flowchart and can be easily programmed by anyone without technical knowledge.
This manual describes instruction words designed to develop programs with HI-FLOW.
For ladder programs, see the following manuals:

<Related manuals>

- SOFTWARE MANUAL PROGRAMMING S10V LADDER CHART For Windows® (Manual number SVE-3-121)
- USER'S MANUAL BASIC MODULES (Manual number SVE-1-100)

<Trademarks>

- Microsoft® Windows® operating system, Microsoft® Windows® 2000 operating system and Microsoft® Windows® XP operating system are registered trademarks of Microsoft Corporation in the United States and/or other countries.
- Ethernet® is a registered trademark of Xerox Corp.

<Note for storage capacity calculations>

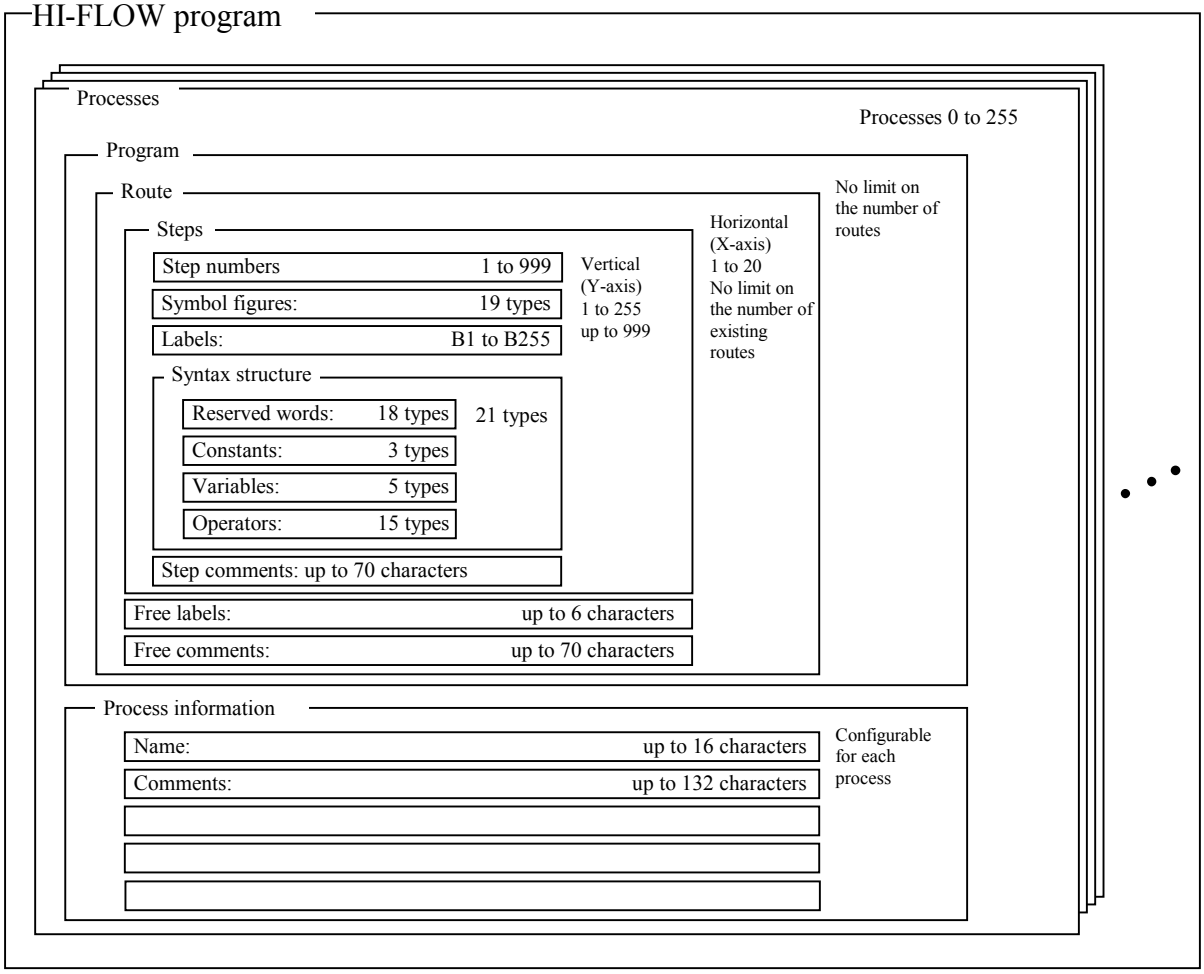
- Memory capacities and requirements, file sizes and storage requirements, etc. must be calculated according to the formula 2^n. The following examples show the results of such calculations by 2^n (to the right of the equals signs).
  1 KB (kilobyte) = 1,024 bytes
  1 MB (megabyte) = 1,048,576 bytes
  1 GB (gigabyte) = 1,073,741,824 bytes
- As for disk capacities, they must be calculated using the formula 10^n. Listed below are the results of calculating the above example capacities using 10^n in place of 2^n.
  1 KB (kilobyte) = 1,000 bytes
  1 MB (megabyte) = $1{,}000^2$ bytes
  1 GB (gigabyte) = $1{,}000^3$ bytes

CONTENTS

1 COMPOSITION OF THE HI-FLOW PROGRAM

1 COMPOSITION OF THE HI-FLOW PROGRAM

This manual describes the standards for, and the contents of, the new HI-FLOW language. Please refer to the manual as necessary when considering a program. A user-created HI-FLOW program consists of the following elements:



2 HOW TO USE THIS MANUAL

2.1 Overview

This manual is organized to cover the elements specified in Chapter 1. Below is a table of the chapters, sections, and pages corresponding to the respective items.

Item	Corresponding chapter or section	Page
Process	Chapter 3	3-2
Program	Section 3.2	3-7
• Routes		3-7
• Steps		3-12
• Step number		3-13
• Symbol figures		3-13
• Labels		3-16
• Syntax structures		3-16
• Reserved words		3-17
• Constants		3-17
• Variables		3-17
• Operators		3-19
• Step comments		3-19
• Free labels		3-20
• Free comments		3-20
Process information	Section 3.3	3-21
• Names		3-21
• Comments		3-21

2.2 Description of Syntax

This manual describes the syntax of the HI-FLOW programming language with regard to each available function, following "outline information." Below is a table of the chapters, sections, and pages corresponding to the respective functions.

Item	Figure	Corresponding chapter or section	Page
Description of syntax		Chapter 4	4-2
Process start and process end		Section 4.1	4-2
• STP			4-3
• RST			4-4
• CLR			4-4
• ACT			4-4
Route start and route end		Section 4.2	4-6
Wait		Section 4.3	4-7
• Conditional expression			4-7
• Timers			4-7
• Output bits			4-7
• Wait timer			4-7
Boxes		Section 4.4	4-9
• Assignment expression			4-9
• ON statements			4-10
• OFF statements			4-11
• Parallel timers			4-11
• TUP			4-12
• TRS			4-13
Control box		Section 4.5	4-14
• ACT			4-14
• RST			4-15
• STP			4-15
• CLR			4-16
Repeat start and repeat end		Section 4.6	4-18
If		Section 4.7	4-19
Jump		Section 4.8	4-21
Escape		Section 4.9	4-22
Para start and para end		Section 4.10	4-23
Select, cell wait, and select end		Section 4.11	4-24
Multi-entry		Section 4.12	4-25
Call		Section 4.13	4-26
Function		Section 4.14	4-27
Wait with precondition		Section 4.15	4-27
Motion		Section 4.16	4-27
Non-synchronous process end		Section 4.17	4-28

2.3 Description of Applied Instructions

HI-FLOW supports applied instructions that are functionally similar to the instructions used in ladder diagrams. Below is a table of items corresponding to the functions of the applied instructions.

Category	Type	Symbol	Function overview	Page
Arithmetic operation instructions	Addition	ADD	S+D → R	5-7
	Subtraction	SUB	S-D → R	5-8
	+1	INC	S+1 → S	5-9
	-1	DEC	S-1 → S	5-10
	Multiplication	MUL	S*D → R	5-11
	Division	DIV	S/D → R	5-12
	Remainder	MOD	Remainder of S/D → R	5-13
	Scale conversion	SCL	S*D1/D2 → R	5-14
Logical operation instructions	Logical product	AND	S AND D → R	5-15
	Logical sum	OR	S OR D → R	5-16
	Exclusive OR	EOR	S EOR D → R	5-17
	Negation	NOT	NOT S → R	5-18
Relational operation instructions	=	EQU	Truth/falsehood of S = D → R	5-19
	<>	NEQ	Truth/falsehood of S<>D → R	5-20
	>	GT	Truth/falsehood of S>D → R	5-21
	>=	GE	Truth/falsehood of S>=D → R	5-22
	<	LT	Truth/falsehood of S<D → R	5-23
	<=	LE	Truth/falsehood of S<=D → R	5-24
	Test	TST	Code S → R	5-25
Data transfer instructions	Transfer	MOV	S → D	5-26
	Collective transfer	MOM	S~Sn → D~Dn	5-27
	Replacement	EXC	S ←→ D	5-28
	FIFO write	PSH	S → D (FIFO table)	5-29
	FIFO read	POP	S (FIFO table) → D	5-30
	Address set	AST	Address S → D	5-31
	Search	SCH	S =D (n) → Set n to R	5-32
Data conversion instructions	BIN-BCD	BTD	BIN → BCD S ----------> R	5-33
	BCD-BIN	DTB	BCD → BIN S ----------> R	5-34
	BIN-7SEG	SEG	BIN → 7-segment S ----------> R	5-35

Category	Type	Symbol	Function overview	Page
Data conversion instructions	BIN-ASC	ASP	BIN → ASCII (pack and unpack) S ----> (R, R+1), (R, R+1, R+2, R+3)	5-36
		ASU		5-37
	ASC-BIN	APB	ASCII (pack and unpack) → BIN (S, S+1), (S, S+1, S+2, S+3) ----> R	5-38
		AUB		5-39
	Absolute value	ABS	| S | → R	5-40
	+/-	NEG	-S → R	5-41
	Decode	DCD	S 2^11~2^15 → Turn ON the 2^n bit of R	5-42
	Encode	ECD	First ON bit number of S → 2^11~2^15 of R	5-43
Shift instructions	Logic right-shift	LSR	S Logic right-shift D → R	5-44
	Logic left-shift	LSL	S Logic left-shift D → R	5-45
	Arithmetic right-shift	ASR	S Arithmetic right-shift D → R	5-46
	Arithmetic left-shift	ASL	S Arithmetic left-shift D → R	5-47
Rotation instructions	CW rotation	ROR	S CW rotation R	5-48
	CCW rotation	ROL	S CCW rotation R	5-49
Function processing instructions	Limiter	LIM		5-50
	Dead band	BND		5-51
	Dead zone	ZON		5-52
	Square root	ROT		5-53
	Maximum value	MAX		5-54
	Minimum value	MIN		5-55
Special instructions	Clear	XCLR YCLR GCLR RCLR KCLR TCLR UCLR CCLR VCLR ECLR FCLR JCLR QCLR HHCLR		5-56

Category	Type	Symbol	Function overview	Page
Ethernet communication instructions	TCP communication	TOP	Open a TCP connection	5-71
		TPOP	Open a TCP connection	5-72
		TCLO	Close a TCP connection	5-73
		TRCV	Receive via TCP	5-74
		TSND	Send via TCP	5-76
	UDP communication	UOP	Open a UDP connection	5-77
		UCLO	Close a UDP connection	5-78
		URCV	Receive via UDP	5-79
		USND	Send via UDP	5-81

2.4 Description of Motion Control Instructions

Motion control instructions can be used to facilitate the programming of motion control under HI-FLOW. Below is a table of items corresponding to the functions of the motion control instructions.

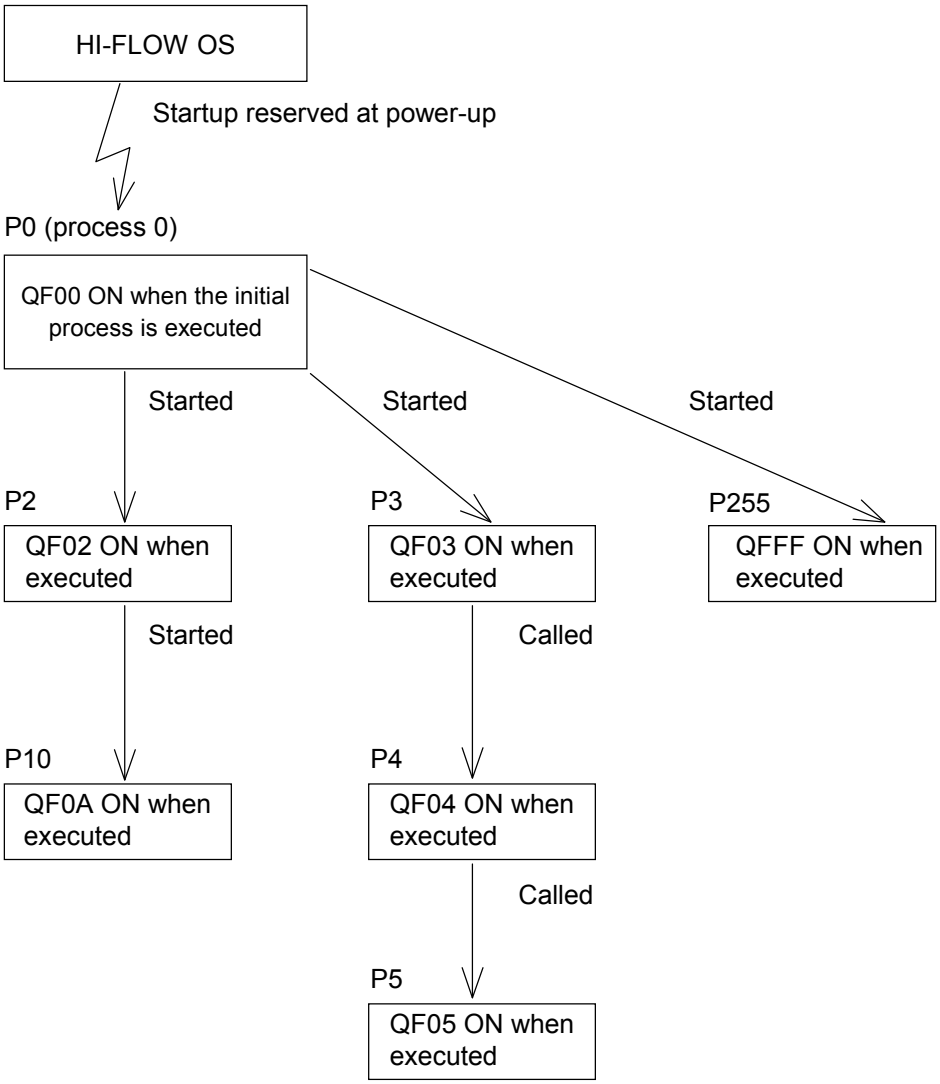
Category	Type	Symbol	Function overview	Page
Motion control instructions	Servo ON	SVON	Turns on a desired servo(s).	6-26
	Servo OFF	SVOFF	Turns off a desired servo(s).	6-28
	Positioning	POS	Positions a specified axis (or axes) at a desired target position(s) at a desired speed(s).	6-30
	External positioning	EXPOS	Moves a specified axis (or axes) a desired distance(s) if the external positioning signal is turned on during a move(s), and then positions them there.	6-34
	Home position return	ZRET	Causes the system to return to the home position.	6-38
	Constant-speed feed	FEED	Moves a specified axis (or axes) in a desired direction(s) at a desired speed(s).	6-43
	Command stop	ABORT	Aborts the command currently under execution	6-47
	Command holding	HOLDS	Temporarily holds the command currently under execution	6-47
	Command reset hold	HOLDE	Releases the command currently under execution from its temporary hold state.	6-47
	Speed control	CHGV	Changes the current speed(s) of movement in progress for a specified axis (or axes).	6-49
	Speed-position control	CHGVP	Changes the current target position(s) and positioning speed(s) for a specified axis (or axes).	6-52
	Torque control	TRQ	Sets a torque value(s) or changes the set torque value(s).	6-55
	Speed override	CHGO	Changes the current percentage value(s) for the set speed(s).	6-58
	Change torque limit	CHGTL	Changes the set torque limits.	6-61
	Change speed loop gain	KVS	Changes the set speed loop gain(s).	6-63
	Change position loop gain	KPS	Changes the set position loop gain(s).	6-65

Category	Type	Symbol	Function overview	Page
Motion control instructions	Set unit	CHGU	Changes a desired unit(s) among those currently set for parameter values, such as speed and positioning units and filter type.	6-67
	Alarm clear	ALMCLR	Clears the alarm(s) for a specified axis (or axes).	6-71
	NOP	NOP	Clears all existing motion control instruction information.	6-73

3 PROCESS

3.1 What is a Process?

Something enclosed by a process start (●) and a process end (◉) or non-synchronous process end (▼) is called a process. It is the largest component of a HI-FLOW program. A process consists of a program composed of at least one route, along with process information about information attached to the process. Create one or more processes for each objective and function and control the target equipment.
Processes are identified by P + process number (a decimal) (P0 through P255).
P0 is called the initial process. Its startup is reserved from the execution controller of HI-FLOW (HI-FLOW OS) when the PCs is turned on. With the startup from the initial process as the turning point, you can control processes P1 through P255.
If a process is being executed, a specified PI/O register will be turned on. Its status can be monitored. (See the command for assigning system bits of "4.7 Utility Functions of the HI-FLOW Process Sheet" of "SOFTWARE MANUAL OPERATION HI-FLOW for Windows® (Manual number SVE-3-132)" of standards QF00 through QFFF.)



Process statuses

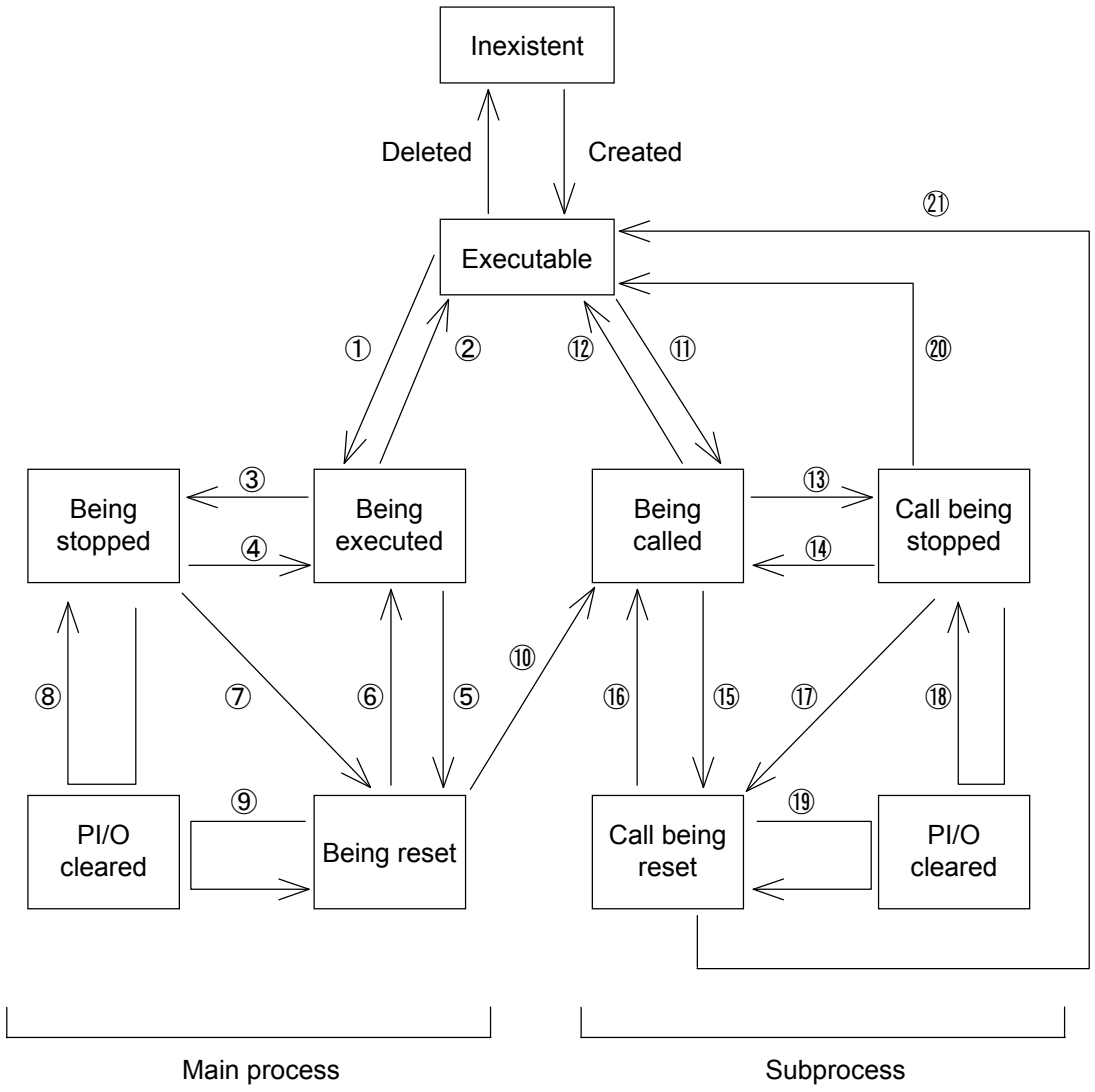
The PCs can be in nine different statuses.

State	Description
Inexistent	The HI-FLOW process does not exist.
Executable	The HI-FLOW process exists and can operate if started.
Under execution	The HI-FLOW process was ACT-started from another process and is being executed.
Standstill	The HI-FLOW process has been stopped at a point in the process because some conditions hold. The information and PI/O value of the process are held. For the time elapsed on the timer, specify a holding operation and continuation of measurement.
Being reset	The HI-FLOW process is stopped at process start because some conditions hold. Process information is initialized. The PI/O value is held. For the time elapsed on the timer, specify upload and reset.
Clear	This clears the bit-type PI/O (ON statement and parallel timer) used in the process to 0 because some conditions hold while the HI-FLOW process is stopped, reset, call stopped, or call reset.
Being called	The HI-FLOW process is executed as subroutine-called from another process.
Call stopped	The HI-FLOW process is stopped at a point in the process because some conditions hold while called. The information and PI/O value of the process are held. For the time elapsed on the timer, specify holding and continuation of measurement.
Call being reset	The HI-FLOW process is stopped at process start because some conditions hold while called. Process information is initialized. The PI/O value is held. For the time elapsed on the timer, specify upload and reset.

While being stopped or reset, the status transits in response to one event where conditions hold. When the conditions no longer hold, the status will remain unchanged. But, clearing is conducted every time the conditions hold.

Process status transition

A process can come in nine different statuses. The chart below shows what (the numbers in the chart) makes the status transit and how (the arrows in the chart).



Relational chart of status transition

① Control box ACT (✚)
② Escape (✕)
③ Process start STP (●), control box STP (✚)
④ Process start ACT (●), control box ACT (✚)
⑤ Process start RST (●), control box RST (✚)
⑥ Process start ACT (●), control box ACT (✚)
⑦ Process start RST (●), control box RST (✚)
⑧ Process start CLR (●), control box CLR (✚)
⑨ Process start CLR (●), control box CLR (✚)
⑩ Process call (▭)
⑪ Process call (▭)
⑫ Process end (●), escape (✕)
Control box RST (✚) to the source process
Process start RST (●) to the source process
⑬ Process start STP (●)
Control box STP (✚) to the source process
Process start STP (●) to the source process
⑭ Process start ACT (●)
Control box ACT (✚) to the source process
Process start ACT (●) to the source process
⑮ Process start RST (●)
⑯ Process start ACT (●)
⑰ Process start RST (●)
⑱ Process start CLR (●)
Control box CLR (✚) to the source process
Process start CLR (●) to the source process
⑲ Process start CLR (●)
⑳ Control box RST (✚) to the source process
Process start RST (●) to the source process
㉑ Control box RST (✚) to the source process
Process start RST (●) to the source process

When a process transits to being executed or being called, there are two startup specifications: master reset and zone. If nothing is specified, zone startup will occur.
When a process shifts to process end (●), escape (✕), or an executable state, the selection of a PI/O value (hold or clear to 0) and the selection of a time elapsed on the timer (upload/reset/continuation of measurement) will be conducted in the same way it is started up.

Status of PCs key switches and processes

Here is how the status of a process on the PCs changes in response to the PCs key switches and a blackout and power restoration of the PCs. HI-FLOW does not distinguish between the RUN and SIM RUN statuses of the PCs and follows the operation of the PCs.

[a] Blackout and power restoration of the PCs (resetting the PCs key switches)
When the PCs undergoes a blackout or power restoration, all processes on the PCs become initialized.
Contents of the initialization
- The process status is made executable.
- The timer is stopped.
- The PI/O is turned off. (The DW, FW, K, and KW are held.)

Process 0 (initial process) is start-reserved. Start reservation means that the process enters a status of being executed when the PCs key switch next enters a RUN status.

[b] PCs key switches being stopped
When the PCs key switches are being stopped, the process status will remain unchanged even if the status of the PCs PI/O and timer change.

[c] PCs key switches in the RUN (SIM RUN) status
The process status will correspond if the PCs PI/O and timer status changes while the PCs key switches are in the RUN (SIM RUN) status.

[d] PCs key switches STOP → RUN (SIM RUN)
When the PCs key switches change from STOP to RUN (SIM RUN), they change from [b] to [c]. At that time, process 0 enters the status of being executed immediately after the PCs undergoes a blackout or power restoration.
Even if not immediately after the PCs undergoes a blackout or power restoration, the system can turn into [c] after an effect (for HI-FLOW-related events only) identical with the blackout and power restoration of the PCs if so specified. (See the system edition commands of “4.7 Utility Functions of the HI-FLOW Process Sheet” of “SOFTWARE MANUAL OPERATION HI-FLOW for Windows® (Manual number SVE-3-132).”)

[e] PCs key switches RUN (SIM RUN) → STOP
The status changes from [c] to [b] when the PCs key switches change from RUN (SIM RUN) to STOP. At that time, the timers (WT and PT) will stop measurements.

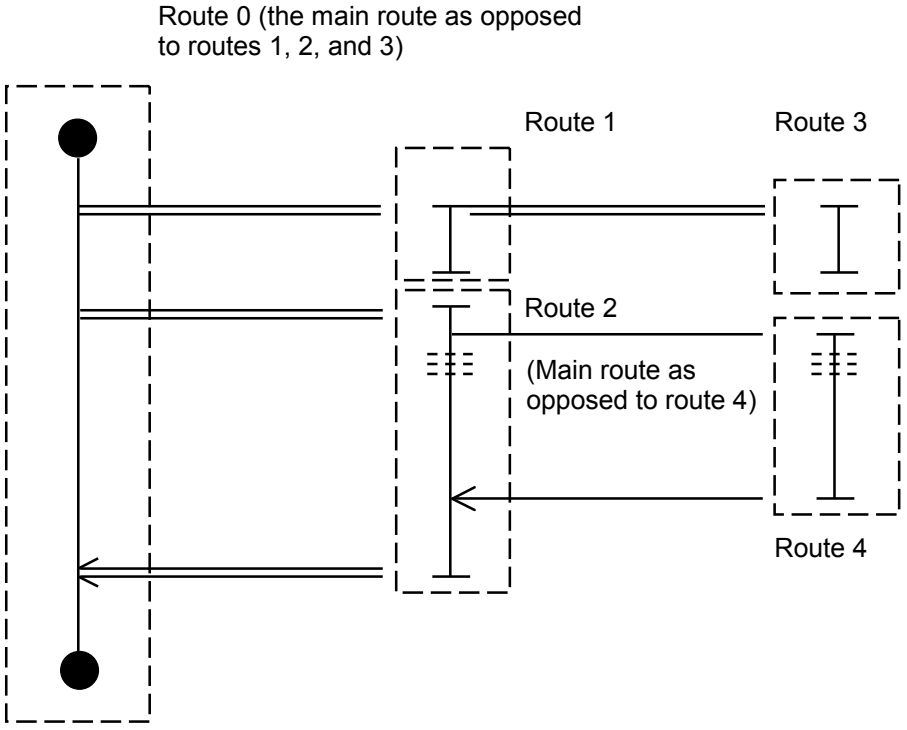
3.2 Program

A process consists of a program and process information. A program is the portion that actually controls the equipment and consists of one or more routes.

Routes

A vertical flow enclosed between process start (●) and process end (●) or between process start (●) and non-synchronous process end (▼) or between route start (┬) and route end (┴) is called the route. It constitutes a component of a process program. A process can be synchronized and/or selectively processed by more than one route. The route where branching occurs is called the main route, while a branching route is called the subroute. A subroute branches by para start (╞) or by select (├), and merges by para end (╡) or select end (┤).

Routes do not need to be identified by number. The route numbers are therefore controlled by the system alone.

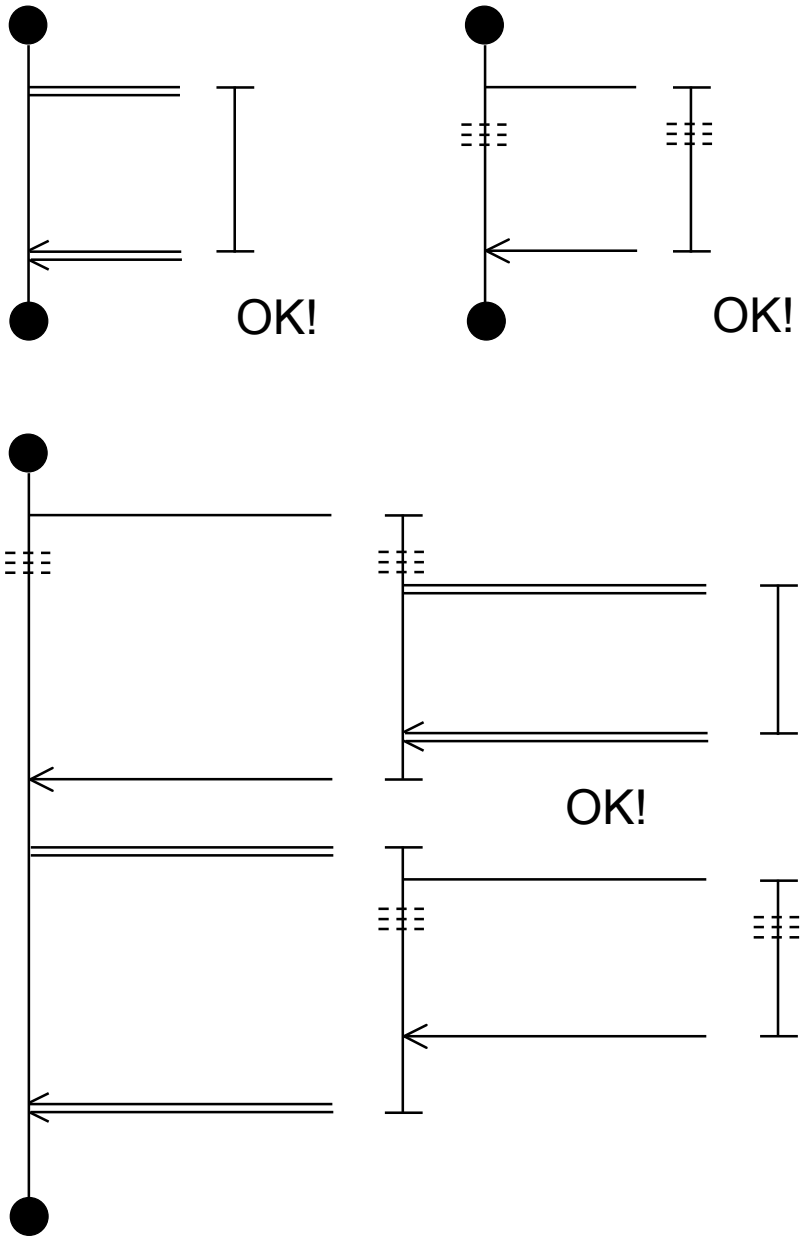


Synchronization routes do not necessarily need to merge. In that case, the source route will only start the route.

The selected routes need to merge in some other route even if unconditionally branched.

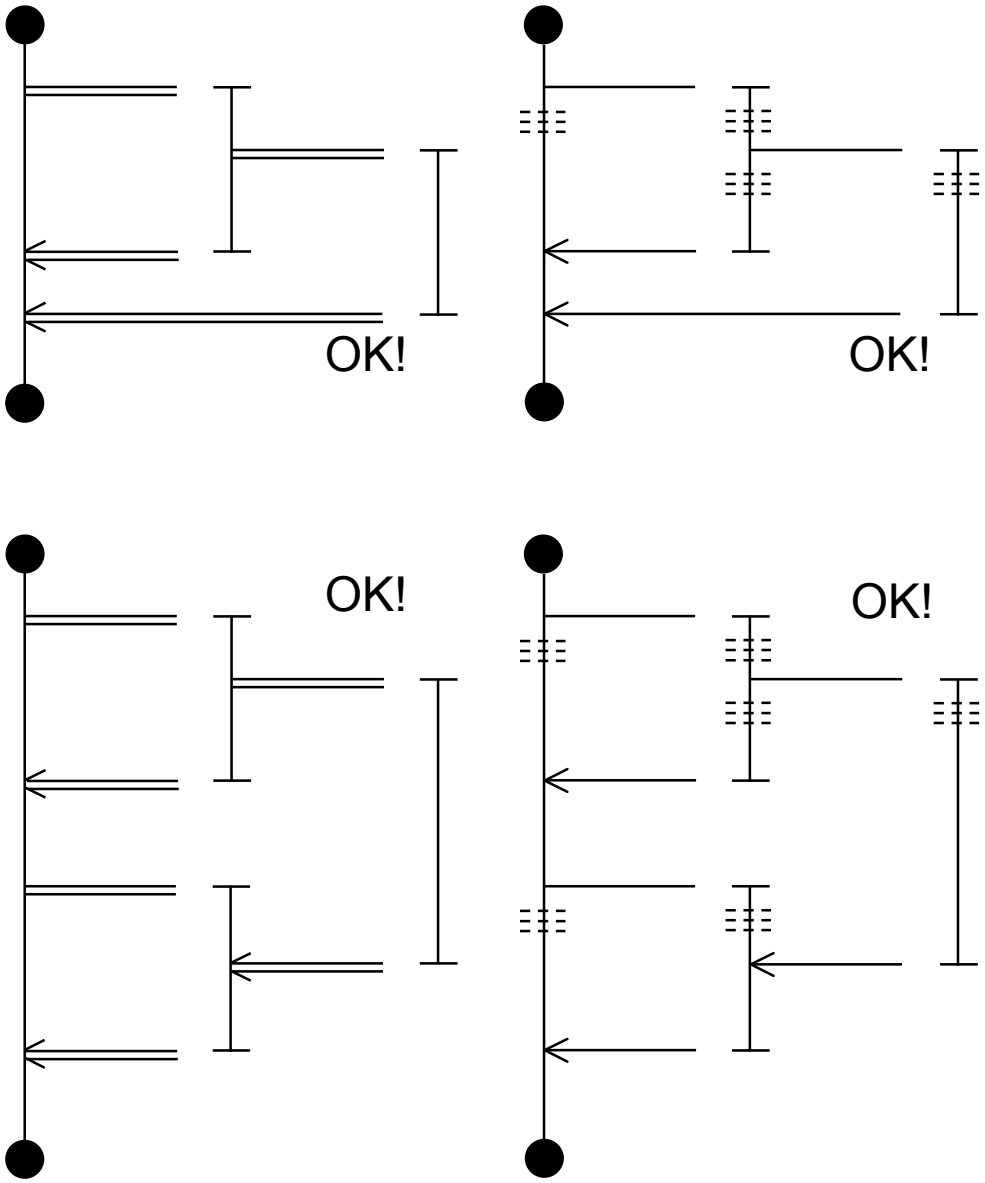
(1) Mixture of synchronization syntax structures and selection syntax structures.
There is no problem with the programming of synchronization syntax structures and selection syntax structures in a closed manner. If they are created in a mixed manner, care should be taken.

(a) The route for starting branching is the same as the route for merging branches.
All patterns are possible for both synchronization and selection.

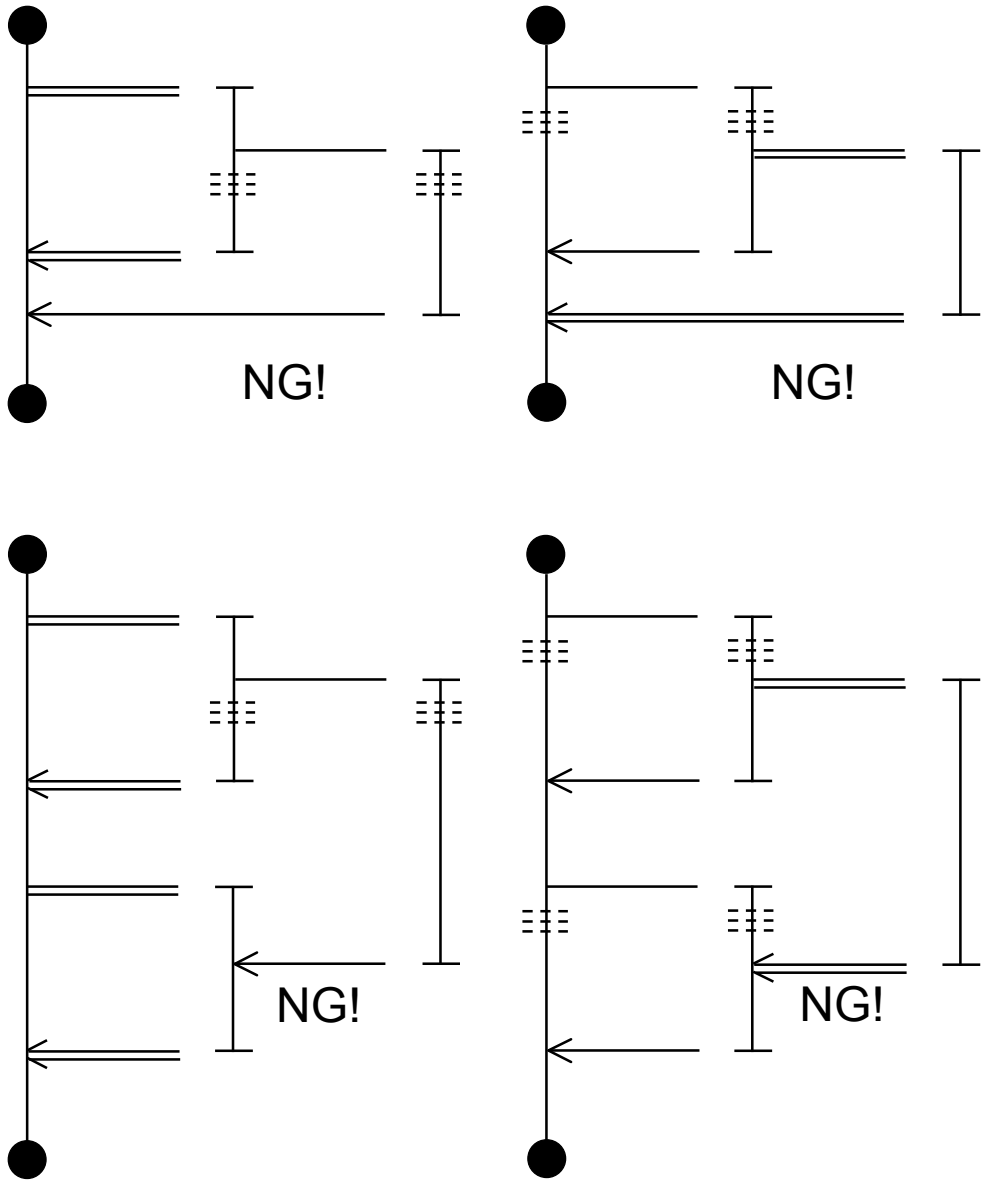


(b) The route for starting branching is different from the route for merging branches.
The system will function as long as a synchronization or selection syntax structure is closed in itself, but it will not function correctly in any other case. This means that these can be created as programs but will not function in actual practice.

[Normal operation]



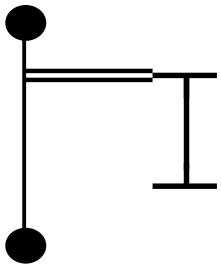
[If it does not function normally]



(2) General syntax structure of non-synchronous routes

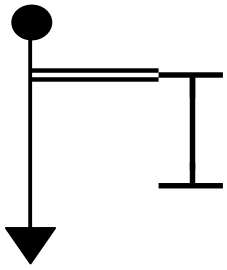
The general syntax structure of non-synchronous branching routes or, simply, non-synchronous routes is a branching route that does not merge to the route from which it has branched and that is used in conjunction with a non-synchronous process end. The major differences between the synchronous and non-synchronous routes are described below.

① Syntax structure of synchronous route (used with a process end)



- The running process involving a synchronous route is terminated by HI-FLOW system only when all of the non-merging branching routes (only one shown left) in that process have reached their ends. If any one or ones of those non-merging branching routes are still on the way to their ends at the time that process has reached its process end, the process will be placed in a wait state until all of them reach their ends.

② Syntax structure of non-synchronous route (used with a non-synchronous process end)



- The running process involving the main route shown left, which is ended by a non-synchronous process end, is terminated by HI-FLOW system immediately when it reaches its process end. This is the case even when not all of the non-merging branching routes (only one shown left) in that process have reached their ends at that time.
- The main route will be initiated by HI-FLOW system again in a next scan even when the execution of any of the non-merging branching routes is being continued.

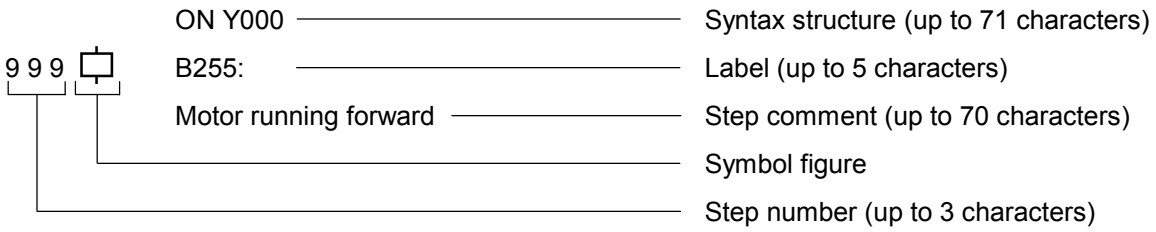
For more information, see “4.17 Non-synchronous Process End.”

CAUTION

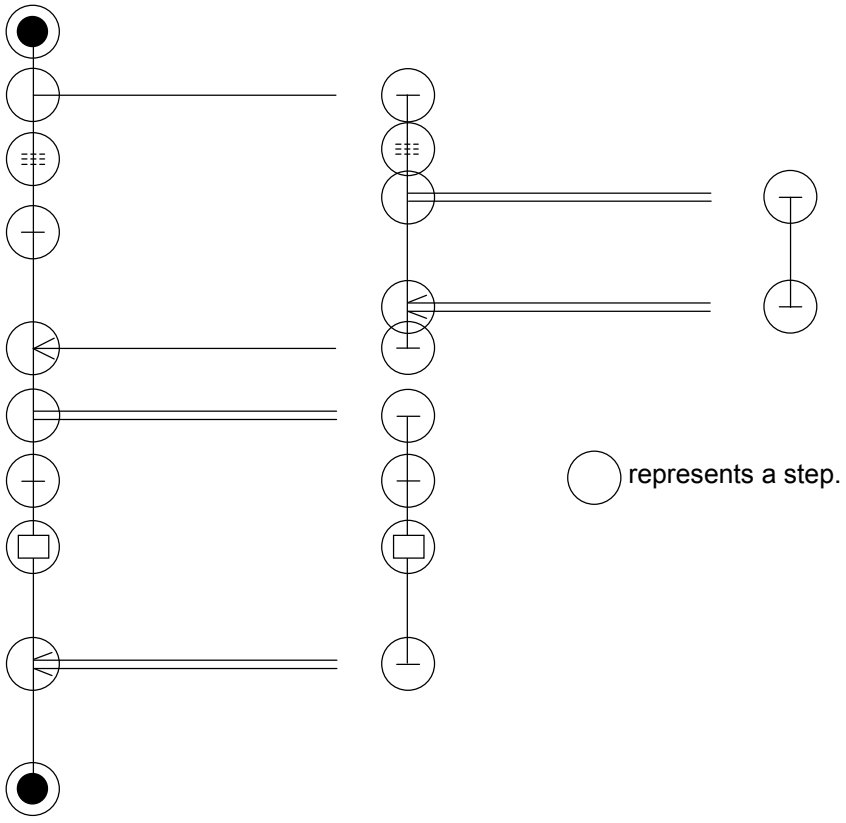
If all of the given branching routes merge to the route from which they have branched, the effect of the non-synchronous process end is the same as that of a process end.

Steps

Steps, together with free labels and free comments, constitute components of a route. A step consists of a step number, symbol figure, label, syntax, and step comment.

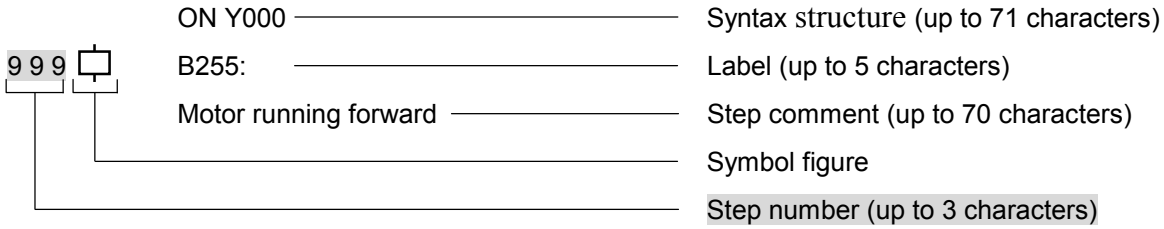


* In combining syntax structure, label, and step comment, you can enter a total of up to 70 characters. Any logical operator in syntax structure consists of one character for editing purposes, but is calculated as 2 characters for counting purposes.



Step number

Is the serial number of a step in the process. During programming, the system automatically assigns such numbers. (The numbers will be from 1 to 999. That is, one process can consist of up to 999 steps.)



Symbol figures

A symbol figure means an overview of conditions, branches, controls, and other factors structure. When creating a step you will need a symbol figure.
Some steps are completed with a symbol figure alone, while others need syntax structure.

A symbol figure comes in 19 types, and the shape of each figure has a meaning.
Here is a list of figures.

* In combining syntax structure, label, and step comment, you can enter a total of up to 70 characters. Any logical operator in syntax structure consists of one character for editing purposes, but is calculated as 2 characters for counting purposes.

List of figures usable on HI-FLOW

(1/2)

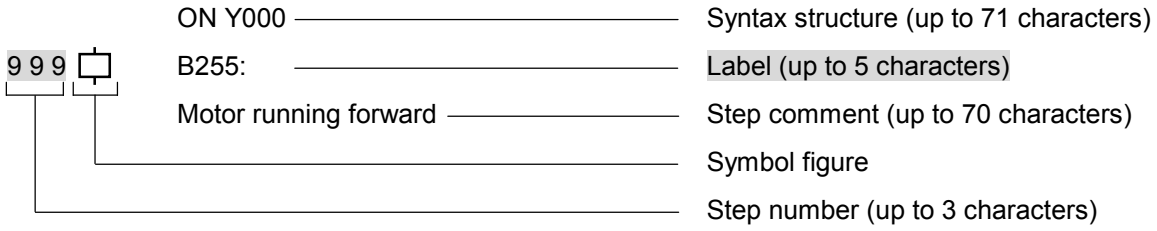
No.	Figure	Name	Function	Syntax	Remark
1		Process start	Starts a process.	Yes	
2		Process end	Ends a process.	No	
3		Non-synchronous process end	Ends a process.	No	This symbol figure, used with a non-synchronous branching route(s), does not force the process to wait for any of those branching routes to reach their ends.
4		Route start	Starts a subroute.	No	
5		Route end	Ends a subroute.	No	
6		Repeat start	Starts a repetition operation.	Yes	An end is judged by >=.
7		Repeat end	Ends a repetition operation.	No	
8		If	Branches an operation by conditions.	Yes	Can be branched to another route.
9		Jump	Unconditional branching	No	Can be branched to another route.
10		Escape	Shuts down its own process.	No	In the case of a subprocess, an identical scan will get you back to the main.
11		Para start	Branches to the synchronization subroutine.	No	
12		Para end	Waits for the synchronization of the synchronization subroutine.	No	When a wait holds for synchronization, go to the next step with an identical scan.
13		Select	Branches to the selection subroutine.	No	
14		Cell wait	Conditions for selecting a route when selectively branched	Yes	Use a pair of route start and select.

(2/2)

No.	Figure	Name	Function	Syntax	Remark
15		Select end	Merges selection subroutes	No	You do not have to merge to the source route. To the next step with no delay in the scan.
		Multi-entry	Re-executes the process, starting with this step, when configured conditions hold	Yes	
16		Wait	Wait for the shift conditions to hold	Yes	
			Wait for a specified time to elapse	Yes	Monitoring is possible for the continuous holding of PI/O.
17		Box	PI/O output	Yes	Equipped with an interlocked Y output
			Assignment expression	Yes	
			PI/O waveform output	Yes	
			Timer reset	Yes	Not limited to 7 pieces
			Timer up	Yes	
18		Control box	Status control for other processes	Yes	With master resetting
			Task control	Yes	
19		Call	Subcall for other processes	Yes	With master resetting
20		Function	Applied instruction	Yes	
21		Condition with clearing of the last status	PI/O clear when shifting between conditions	Yes	Combined with a wait
		Wait timer with clearing of the last status	PI/O clear when the timer is up	Yes	
22		Motion	Motion control instruction	Yes	

Labels

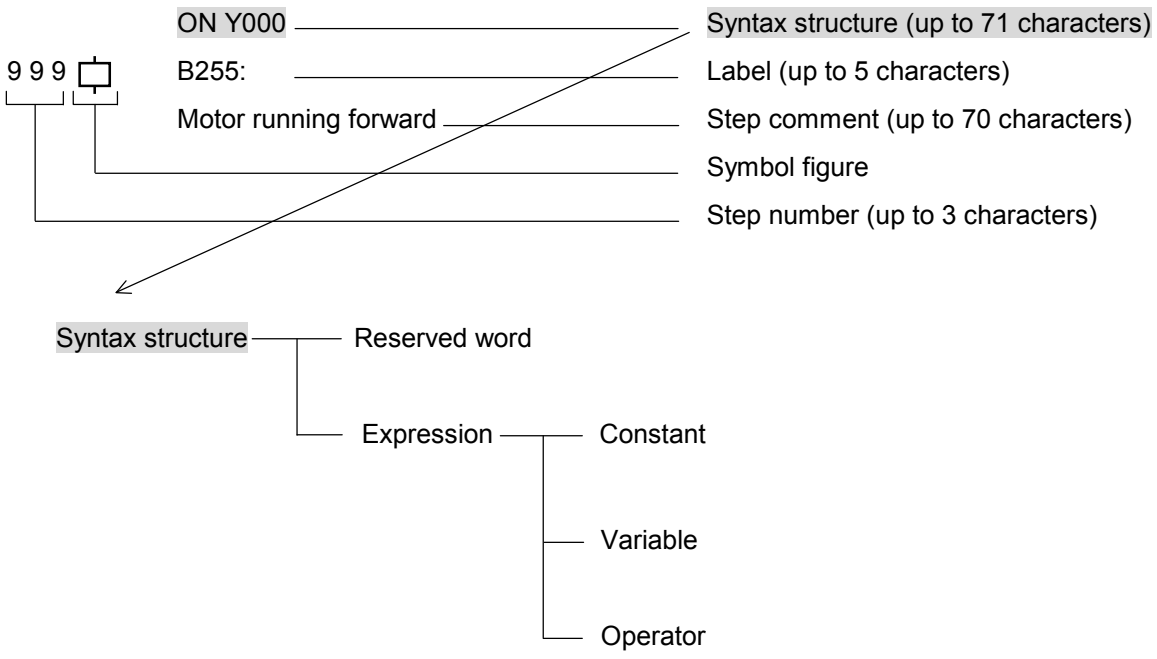
A label consists of a code between B1 and B255 (which can be created for each process and cannot be set to branch to another process). A colon (:) represents a jump destination from a branch figure and can only be added to a step.



Syntax structures

A syntax structure may contain conditional expressions, assignment expressions, and/or control statements. It assists figures and specifies their contents. They include symbol figures that require no syntax structure.

A syntax structure consists of an expression(s) composed of reserved words, constants, variables, and operators.



* In combining syntax structure, label, and step comment, you can enter a total of up to 70 characters. Any logical operator in syntax structure consists of one character for editing purposes, but is calculated as 2 characters for counting purposes.

Reserved words

Note that, since the system gives the reserved word a special meaning, you cannot use it as a symbol name.

List of reserved words

ACT, CLR, MRST, ON, OFF, RST, STP, TASK, TUP, TRS, TCNT,
CNxxx, PTxxx, WTxxx, Bxxx, Pxxx, H????????
Name of applied instruction (see Chapter 5.)

xxx: It means a decimal constant.
????????: It means a hexadecimal constant.

Constants

HI-FLOW allows you to specify long-length constants.

- Constants
  - Bit-types: 0, 1
  - Word-types: Decimals: -32768 to 32767
    Hexadecimals: H0 to HFFFF
  - Long-lengths: Usable only in applied instructions.
    Decimals: [-2147483648] to [2147483647]
    Hexadecimals: [H0] to [HFFFFFFFF]

Variables

HI-FLOW allows you to use real PI/O registers (such as X and Y).
Applied instructions allow you to specify variables indirectly by placing @ before the PI/O and handle variables as long-lengths by [ ].
Below is a list of real PI/O registers usable on HI-FLOW.

Variables	Bit-types	Simple	
		One-dimensional arrays	For example, X000 (5)
	Word-types	Simple	
		One-dimensional arrays	For example, XW000 (FW000), but only in applied instructions. [FW000] handles FW000 and FW001 as long-lengths.
	Long-types	Simple	

Below shows how to handle the range of word-type and long-type values.

Word-types: -32768 to 32767

Long-types: -2147483648 to 2147483647

List of PI/O registers

Item		Symbol	Range	Type	Remark
Registers	External inputs	X	000 to FFF	Bit	
		XW	000 to FF0	Word	
	External outputs	Y	000 to FFF	Bit	
		YW	000 to FF0	Word	
	Communication link registers	G	000 to FFF	Bit	
		GW	000 to FF0	Word	
		A	000 to FFF	Bit	
		AW	000 to FF0	Word	
	Internal registers	R	000 to FFF	Bit	
		RW	000 to FF0	Word	
		K	000 to FFF	Bit	
		KW	000 to FF0	Word	
		M	000 to FFF	Bit	
		MW	000 to FF0	Word	
		E	000 to FFF	Bit	
		EW	000 to FF0	Word	
		Z	000 to 3FF	Bit	
		ZW	000 to 3F0	Word	
		S	000 to BFF	Bit	
		SW	000 to BF0	Word	
	Other registers	J	000 to FFF	Bit	For linkage with a ladder
		JW	000 to FF0	Word	
		Q	000 to FFF	Bit	
		QW	000 to FF0	Word	
		HH	000 to 1FF	Bit	For linkage with other processes
		DW	000 to FFF	Word	
		FW	000 to BFF	Word	
	S10V extended registers	LB	0000 to FFFF	Bit	
		LBW	0000 to FFF0	Word	
		LWW	0000 to FFFF	Word	
		LXW	0000 to 3FFF	Word	
	Timers	WT	000 to 255		Decimal notation
		PT	000 to 255		
	Counters	CN	000 to 127		Decimal notation
	Labels	B	001 to 255 with user-specified labels (up to 6 characters)		Decimal notation, for each process

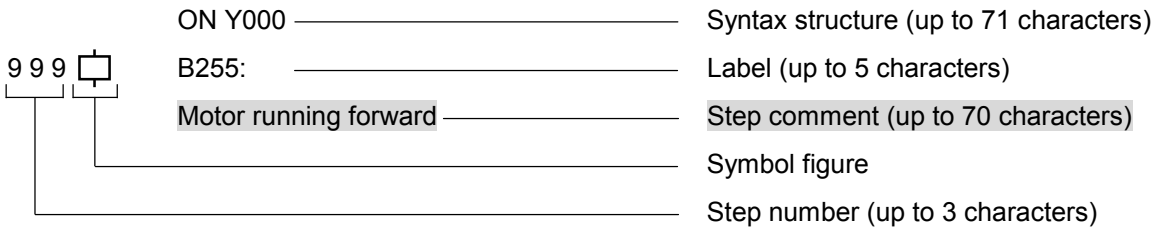
Operators

Operators come in four types: logic, four operations, relations, and parentheses. Four operations are handled

Item		Description	Priority
Operators	Logic	& (AND) | (OR) ~ (NOT) ^ (Exclusive OR)	5
	Four operations	* /	2
		+ -	3
	Relations	=, <>, <, >, >=, <=	4
	Parentheses	Up to 7-fold	1

Step comments

Step comments are written by means of alphabetic characters, numeric characters, and special symbols. The system allows you to enter as many characters as the capacity of each line. This does not necessarily have to be created.



* In combining syntax structure, label, and step comment, you can enter a total of up to 70 characters. Any logical operator in syntax structure consists of one character for editing purposes, but is calculated as 2 characters for counting purposes.

Free labels

HI-FLOW allows you to create jump destination labels in addition to steps. (Such labels can be omitted.) These are called free labels. You are free to give a name other than those of the reserved words in up to six characters, beginning with an alphabet character. Lastly, a colon (:) is required.

A free label can only be added to something other than a step. It will become a jump destination for a branch figure.

LABEL: ——————————— Free labels (up to 6 characters)

Merging position ——————————— Free comments (up to 70 characters)

Free comments

HI-FLOW allows you to create a comment somewhere other than the location of a step. (This can be omitted.) This is called the free comment. The system allows you to use alphabetic characters, numeric characters, and special symbols and enter as many characters as the capacity of one line. This makes it possible to add a comment where it is easy to find.

A free comment can only be added to something other than a step.

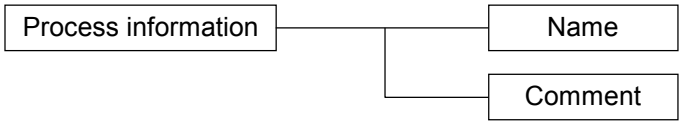
LABEL: ——————————— Free labels (up to 6 characters)

Merging position ——————————— Free comments (up to 70 characters)

* When you combine a free label with a free comment, the system receives up to a total of 70 characters (including a colon (:) as a free label).

3.3 Process Information

A process consists of a program and process information. Process information defines ancillary information regarding a process. This allows you to create a more user-friendly process. Process information consists of two elements and can be changed as desired by means of a process information command.



Names

A name is something included in process information. You can give a specific process a unique name with up to 16 regular-size characters.

Comments

A comment is something included in process information. You can give a specific process a comment with up to 132 regular-size characters.

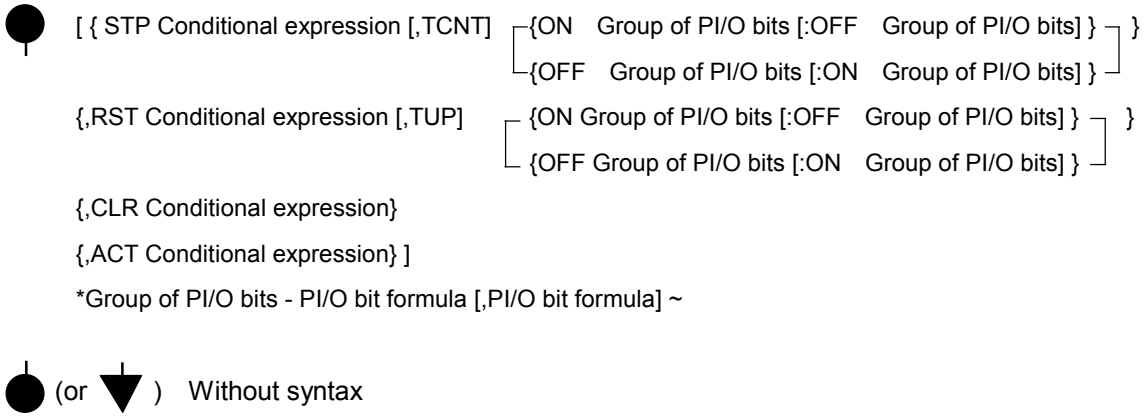
4 DESCRIPTION OF SYNTAX

This chapter describes the types and details of language syntax, including figures and jump destination labels. Here are typical examples.
The [ ] indicates that the item enclosed is omittable. The { } selected. The ~ repeated.

4.1 Process Start and Process End

It means the start or end of a process. The system automatically adds a figure, thus obviating the need of entry.
Process start can be set to conditions for stopping, resetting, restarting, and PI/O initializing a specific process. (See STP, RST, CLR, and ACT.)
Process end performs an operation if all routes other than the route of the current system are finished. If not finished, the system will wait until they are finished. (This is not the case if the route is ended by a non-synchronous process end [see "4.17 Non-synchronous Process End" for more information]). When started, and if the system is specified to master resetting, the system clears the bit-type PI/O to be turned on by the process of the current system, to 0 (ON statement and parallel timer).
The way a timer used in the process of the current system follows the way the system is started. If the system is started with a TUP option specified, the current timer expires. If the system is started with a TRS option specified, the system discontinues the timer. If unspecified, the system continues timer measurement.

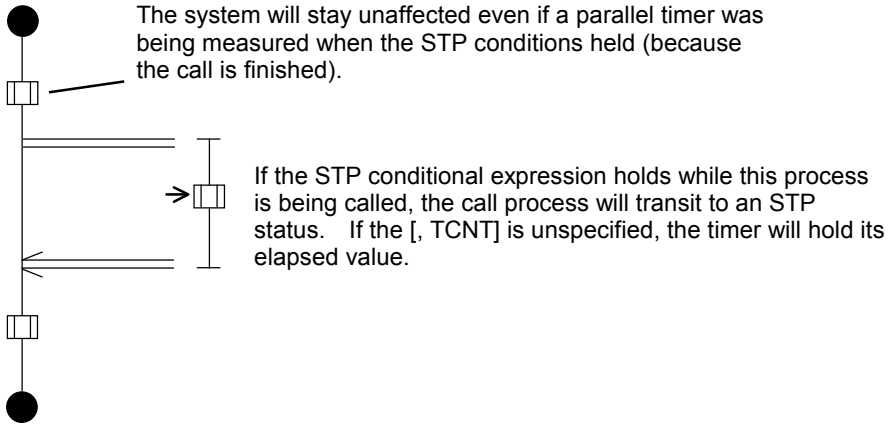
[Syntax]



(or) Without syntax

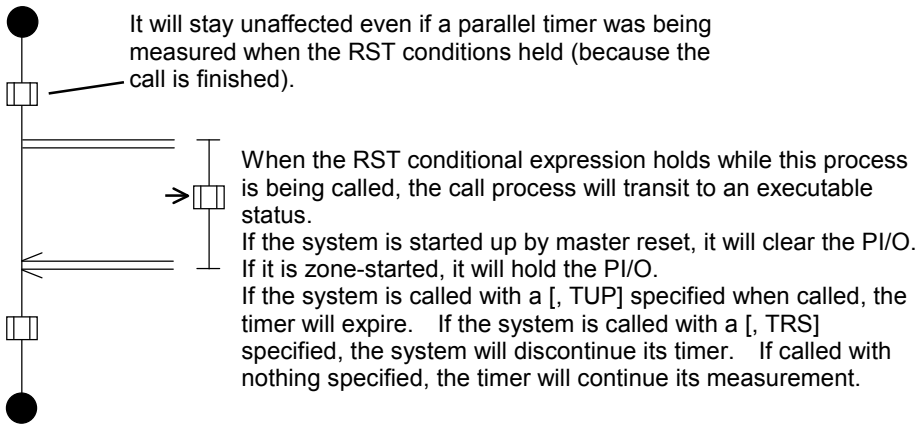
STP

- When the status of a process is “being executed,” and if the conditional expression holds, the system will stop executing the process of the current system at the current position. (It will transit to a stop status.)
- When STP holds, the system will hold the elapsed value of the timer and the bit-type PI/O value of the bit type to be turned on by the process of the current system (ON statement and parallel timer). Note that this cannot prevent events where the system is turned on or off by another process or something similar.
- When STP conditions hold, the system will turn on or off the group of optional PI/O bits. (If the conditions do not hold, the system will turn on or off every scan in reverse of the specification made.)
- Specify a [, TCNT] option, and the timer will continue its measurement when the system transits to a stop status. If unspecified, the timer will hold the elapsed value.
- A process which was called when the STP conditions held will transit to a stop status in a similar manner to the source process. But, the call complete or uncalled process will stay unaffected.



RST

- When the process status is being executed or stopped, and if the conditional expression holds, the system will stop executing its own process and wait at process start (transiting to a reset status).
- When RST holds, the system will hold the bit-type PI/O value to be turned on or off by its own process (ON statement, OFF statement, and parallel timer). Note, however, that this cannot prevent events where the system is turned on or off by another process.
- When RST conditions hold, the system will turn on or off the group of optional PI/O bits. (If the conditions do not hold, the system will turn on or off every scan in reverse of the specification made.)
- The timer will expire at the elapsed value if it is set to an [, TUP] option. If unspecified, the system clears the elapsed value to 0 and discontinues measurement.
- A process which was being called when the RST conditions transit to an executable status. At that time, how the PI/O and timer are handled will be the way the system is started up. The call complete or uncalled process will remain unaffected.



CLR

If the conditions hold in a stop status or reset status, the system will clear to 0 the bit-type PI/O to be turned on by its own process (ON statement, or bit-type PI/O used on the parallel timer).

ACT

If the system is in a stop status or reset status and if the STP conditions or RST conditions do not hold, and if the conditional expression holds, the system will restart executing the process (transiting to being executed).

[Typical programs of process start (●)] ~ is a continuation of the same line.

1. ● STP X000, RST X001, CLR X002, ACT X003

- When X000 is ON, go to a stop status (the elapsed value of the timer is held).
- When X001 is ON, go to a reset status (the timer is discontinued).
- When X002 is ON in stop/reset status, clear to 0 the ON statement used in this process and the bit-type PI/O in the parallel timer.
- When in a stop/reset status and if X000 and X001 are OFF and X003 is ON, go to the process being executed.

2. ● STP G000&X020, TCNT [ON J000:OFF J001] ~, RST Q000, TUP

- When G000 and X020 are both ON, go to a stop status (the timer continues its measurement).
- When transiting to a stop status, turn J000 ON and J001 OFF.
- When Q000 is ON, go to a reset status (timer expiration).
- When the process is being executed, turn OFF each scan J000 and turn ON J001.

3. ● RST FW000<DW000 [OFF G100], ACT FW001=0

- When FW000 becomes smaller than DW000, go to a reset status (timer is discontinued). Then, when transiting to a reset status, turn G100 OFF.
- When in a stop/reset status, and when FW000 is no less than DW000 and FW001 is 0, go to the process being executed.
- When the process is being executed, turn on every scan G100.

4. ● RST Q001, TUP [ON J001, G200], CLR X200

- When Q001 is ON, go to a reset status (timer is discontinued).
- When transiting to a reset status, turn ON J001 and G200.
- When X200 is ON in stop/reset status, clear to 0 the ON statement used in this process and the bit-type PI/O in the parallel timer.
- When the process is being executed, turn OFF every scan J001 and G200.

The STP, RST, CLR, and ACT in process start may take any sequence.

4.2 Route Start and Route End

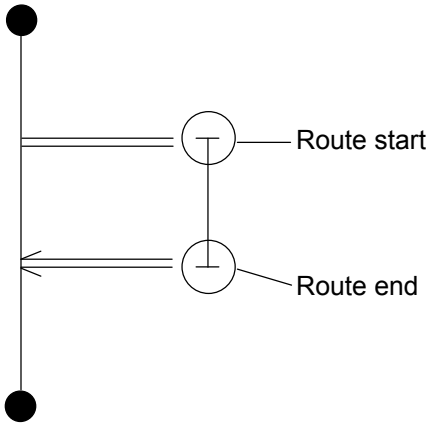
⊤ Without syntax

⊥ Without syntax

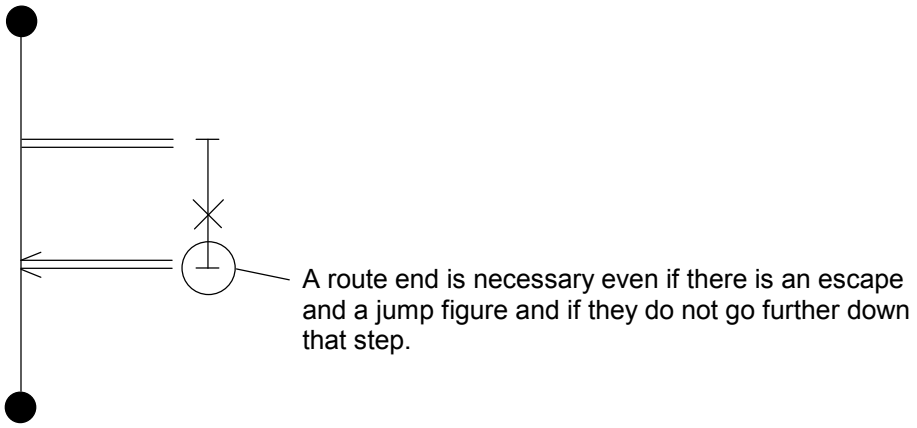
Route start means the start of a subroute, while route end means the end of a subroute. Be sure to use them as a pair.
Creating a subroute builds up a synchronization syntax structure or a selection branch syntax structure.

[Typical programs with route start ⊤ and route end ⊥]

1.



2.



4.3 Wait

At this step, the system waits until the conditions hold for shifting to the next step. The condition for shifting is either a conditional expression or a wait timer (waiting for a specified time to elapse).

[Syntax]

┼ { Conditional expression [, timer, output bit] }
{ WTxxx expression [, conditional expression] }

Conditional expression

- It consists of a bit-type or word-type number and operator.

Timers

- Each of these timers monitors the status until the conditional expression holds. The units are 100 ms.
- Enter a decimal constant.
- The setting range is from 0 to 32767. Setting the system between -32768 and -1 will operate the system on the assumption that it is set between 32768 and 65535.
- The system can monitor up to 64 timers at the same time. Make sure that no more than 64 timers are monitoring at the same time.

Output bits

- This bit will go ON when the conditional expression does not hold even after a time is specified by the timer specified above.
- Registers that can be specified to output bits are bit-type registers as specified below: Y, G, A, R, K, M, E, Z, S, J, Q
- When monitoring starts, this bit is turned off unconditionally.
- The system will not go to the next step unless the conditions hold even after a time is specified by a specific timer.
- The output bits do not get turned off even if the conditions hold after the output bits are turned on.

Wait timer

- Using a wait timer allows you to delay progress for a specified time in a desired step. The system allows you to use WT000 to WT255 (decimal numbers) and to set the delay to any value between 0 and 32767 (decimal) at increments of 100 ms. Setting it between -32768 and -1 will operate the system on the assumption that it is set between 32768 and 65535.

- If the wait timers of the same number wait for a specified time at more than one location, the other steps will turn on the specified PI/O (standard HH1FA) until the step that occupied the timer first opens the timer, and will wait for the timer to open. The result is prolonged delays in other timers.
- The system allows you to set a conditional expression on the wait timer. In that case, the system will wait until the conditional expression continues to hold for a specified time.

[Typical programs of wait (┼)]

1. ┼ X000

   Go to the next step when X000 is ON.

2. ┼ GW000<H2000

   Go to the next step when GW000 becomes smaller than H2000.

3. ┼ X001 (FW000)

   Go to the next step at the turning-on of the X register with the FW000 value as a subscript value at the time of a condition check (which may vary every time).

4. ┼ WT000 (100)

   Go to the next step 10 seconds after the system reaches this step first.

5. ┼ WT255 (10, X01F)

   Go to the next step if X01F remains ON for one consecutive second after arriving at this step.

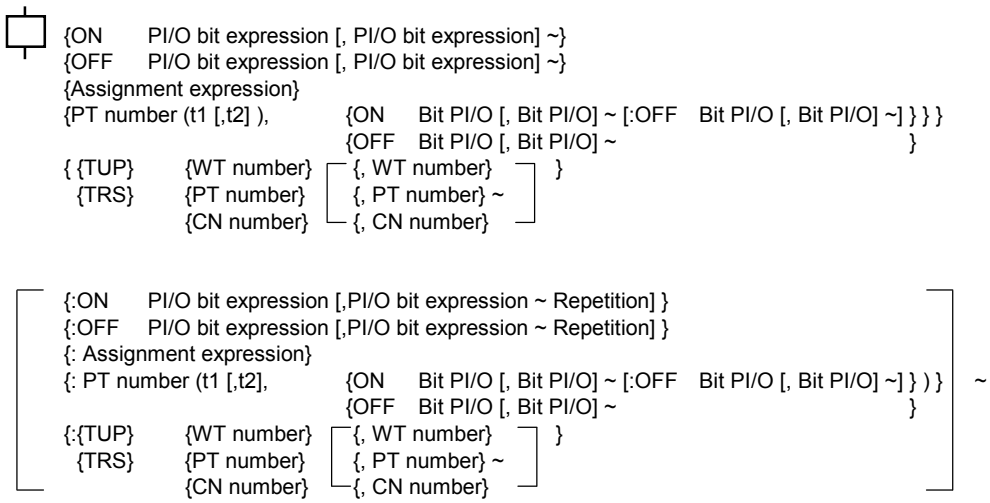
6. ┼ GW000>H2000, 100, Y000

   Go to the next step when GW000 becomes larger than H2000. Turn on Y000 if the GW000 fails to become larger than H2000 within 10 seconds. Do not turn Y000 OFF even if GW000 becomes larger than H2000 after Y000 is turned ON.

4.4 Boxes

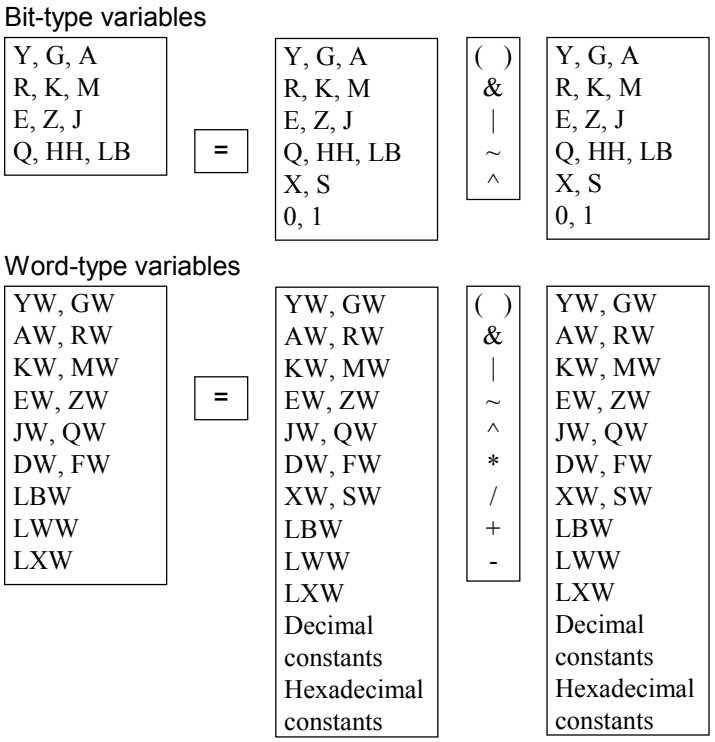
The system performs PI/O output, data processing, and timer control. Separating boxes with a colon (:) produces a complex sentence.

[Syntax]



Assignment expression

Assign the result of logic and four operations to a variable. An expression can take the form of a single-dimensional array, while array subscript values can only take the form of a word type. Below list the variables and operators available.



The system regards operation items and results as uncoded.
In multiplications, both the multiplier and multiplicand are both one-word. The portion that cannot be expressed in one word is rounded off, with the result being one-word.
In divisions, too, both the divisor and the dividend are one-word. The portion that cannot be expressed in one word is rounded off, with the result being one-word. Dividing a number by 0 results in the answer remaining unchanged.
The status of operation results (including normal termination and overflow occurrence) will not be answered back. If answer back is necessary, use an applied instruction.

[Typical programs with assignment statements (⊏)]

1. ⊏ FW000=FW001+FW002

   Add FW001 and FW002 at the particular time, assign the sum to FW000, and go to the next step.

2. ⊏ YW000 (DW001) = HFFFF

   Assign /FFFF to the array of YW000 with DW 001 at the time as a subscript value.

ON statements

Turn on the specified PI/O output bits (Y, G, A, R, K, M, E, Z, J, Q, and HH). Separating them with a comma (,) produces more than one PI/O output. Although the PI/O output bit can take a one-dimensional array, the array subscript value can only be a word type.

[Typical programs with ON statements (⊏)]

1. ⊏ ON Y000, Y00F:OFF Y001

   Turn ON Y000 and Y00F, turn OFF Y001, and go to the next step.

2. ⊏ ON G000 (GW010)

   Turn ON the bit away from G000 by the GW010 value at the particular time and go to the next step.

OFF statements

Turn OFF the specified PI/O bits (Y, G, A, R, K, M, E, Z, J, Q, and HH). Separating it with a comma (,) will produce more than one PI/O output. PI/O output bits can take the form of a one-dimensional array, while array subscript values can only be word types.

[Typical programs with OFF statements (⊓)]

1. ⊓ OFF Y000, Y001

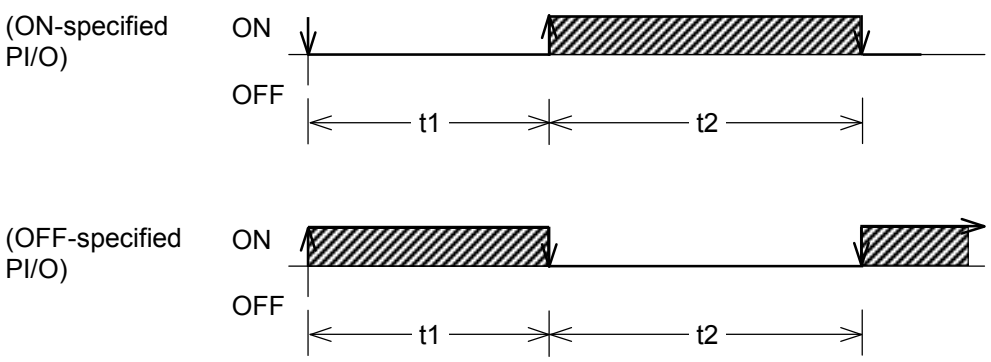
   Turn OFF Y000 and Y001 and go to the next step.

2. ⊓ OFF G000 (GW010)

   Turn OFF the bit separated from G000 by the GW010 value at the particular time, and then go to the next step.

Parallel timers

The system will produce a waveform onto a desired PI/O. The t1 represents rising time, while the t2 represents falling time.
When t1 is 0, the ON-specified PI/O will only fall after the elapse of time t2. The OFF-specified PI/O will only rise after the elapse of time t2. When t2 is 0 or by default, the ON-specified PI/O will only rise after the elapse of time t1. The OFF-specified PI/O will only fall after the elapse of time t2. Soon after giving an instruction for waveform output, the system will go to the next step.
The system allows you to use any number between PT000 and PT255 and to set the time at increments of 100 ms in the range from 0 to 32767, to t1 and t2 respectively. Setting the system between -32768 and -1 will operate the system on the assumption that it is set between 32768 and 65535.
If a specified timer is occupied when the timer is started up, the system will turn ON the specified PI/O (standard HH1F9) and wait until the timer is opened.
The bit PI/O available can take the form of more than one description with a comma (,) and a complex sentence or single-dimensional array with a colon (:). The bit PI/O types available are Y, G, A, R, K, M, E, Z, J, Q, and HH.



[Typical programs with parallel timer (⊓)]

1. ⊓ PT000 (10, 10, ON Y000:OFF Y001)

	On passing by this step (to a next step at once)	1 second later	2 seconds later
Y000	? →OFF	→ON	→OFF
Y001	? →ON	→OFF	→ON

2. ⊓ PT010 (20, ON G000:OFF G001)

	On passing by this step (to a next step at once)	2 seconds later	
Y000	? →OFF	→ON	→
Y001	? →ON	→OFF	→

3. ⊓ PT255 (0, 30, ON J100:OFF J101)

	On passing by this step (to a next step at once)	3 seconds later	
J100	? →ON	→OFF	→
J101	? →OFF	→ON	→

TUP (timer up)

Put the timer in the process of measurement into the expired position.
For a wait timer, set the elapsed value of the timer in the process of measurement to the setting. As a result, the wait status is canceled and step waiting for the timer to expire will go to the next step.
For a parallel timer, set the elapsed value of the timer to t2 (or t1 if t2 is set to its default). As a result, the parallel timer produces a PI/O output earlier than the specified time.
For a loop counter, set the elapsed value of the counter to the final value. As a result, the system will get out at the next loop check.

[Typical programs with timer up (⊓)]

1. ⊓ TUP WT001, WT002, PT001, CN001

Put the wait timers 1 and 2, parallel timer 1, and counter 1 to the up position.

TRS (timer reset)

Reset the timer in the process of measurement.

In the case of a wait timer/loop counter, the system will perform the same operation as when the timer is up. In the case of a parallel timer, reset the elapsed times t1 and t2 of the timer in the process of measurement. The status of the specified PI/O will be held as that when timer reset was issued.

[Typical programs with timer reset (⊓)]

1. ⊓ TRS WT001, WT002, PT001, CN001

   Reset wait timers 1 and 2, parallel timer 1, and counter 1.

4.5 Control Box

The system allows you to start (restart), stop, reset, and clear the PI/O with regard to other processes.

[Syntax]

```
{  ACT  Pxxx { [-Pxxx] [,Step number] [, MRST] [ {, TUP} ] }  }
                                                    {, TRS}
                {, TASK, Factor number            }
{  RST  Pxxx { [-Pxxx] [, TUP]  }  }
                { [, TASK] }
{  STP  Pxxx [-Pxxx] [, TCNT]  }
{  CLR  Pxxx [-Pxxx]  }
```

ACT

	Item	Description
1	Function overview	Start a process with P0 to P255. The process range can be specified with a hyphen (-). If no step number is specified, begin with step 1. The specified step does not have to be the main route. Immediately after startup, go to the next step.
2	Action of the process started	A started process is not executed only once. When the process end is finished, the next scan executes the process again, beginning with the process start. (The condition is similar even if a step is specified.)
3	Startup of the process being executed	Turn on the ACT bit of the result display bit of the control box, then go to the next step (standard HH1FF).
4	Startup of a non-existent process	Turn on the ACT bit of the result display bit of the control box, then go to the next step (standard HH1FF).
5	Startup of a stopped process	The stopped process starts up, resuming the execution.
6	Startup of a reset process	The reset process starts up, resuming the execution with the process start.
7	Indication of the timer status	When set to a TUP option, the system will put into the up position the parallel timer occupied in its own process when executing a process end or escape and when shifting to an executable status. When set to a , TRS option, the system will reset the parallel timer occupied by its own process when executing a process end or escape and when shifting to an executable status.
8	Startup of master reset specification	When set to a , MRST option, the system will clear to 0 the bit-type PI/O turned on in its own process when executing a process end or escape and when shifting to an executable status.
9	Startup of CPMS tasks	When set to a , TASK and factor number option, the system will use Pxxx as a CPMS task (1 to 127) and issue an RLEAS, QUEUE macro.

RST

	Item	Description
1	Function overview	The system will reset a process specified with P0 to P255. The range can be set with a hyphen (-). Immediately after issuance, the system will go to the next step.
2	Action of a reset process	The system will abort executing a specified process, transit to a reset status, and wait for re-execution at process start (will start ACT from another process or re-execute the process when the ACT condition holds at its own process start).
3	Indication of the timer status	When set to a , TUP option, the system will put into the up position the parallel timer occupied by the process. If unspecified, the system will reset the timer. This option will only be effective in the specified process and will not affect the call process.
4	PI/O of the process to be reset	If a master reset is started, the system will clear to 0 the bit PI/O turned on or off in its own process.
5	Issuance to a stopped process	The system will abort executing a specified process, transit to a reset status, and wait for re-execution at process start (will start ACT from another process or re-execute the process when the ACT condition holds at its own process start).
6	Issuance to a non-existent process	The system will turn on the RST bit of the result display bit of the control box, then move on to the next step (standard HH1FD).
7	Issuance of a reset to the own process	The system specifies its own process number by a parameter.
8	Stoppage of the CPMS task	When set to a , TASK option, the system will issue an ABORT macro with Pxxx as a CPMS task.

STP

	Item	Description
1	Function overview	The system will stop a process specified with PO to P255. The range can be specified with a hyphen (-). Immediately after issuance, the system will go to the next step.
2	Action of a stopped process	The system will stop executing a specified process and transit to a stopped status. It will wait for re-execution at the current position.
3	Re-execution conditions	The system will start ACT from another process and re-execute the process when the ACT conditions hold for starting its own process.
4	Indication of the timer status	When set to a , TCNT option, the system will continue to measure the parallel timer occupied by the process. When unspecified, the system will stop its timer measurement. This option is effective for all processes strung by call by a specified process.
5	PI/O of the process to be stopped	When a master reset is started, the system will clear to 0 the bit PI/O turned on or off by its own process.
6	Issuance to a non-existent process	The system will turn on the STP bit of the result display bit of the control box and go to the next step (standard HH1FE).
7	Issuance to a reset process	The system will turn on the STP bit of the result display bit of the control box and go to the next step (standard HH1FE).
8	Issuance of a stop to the own process	The system will specify its own process number by a parameter.

CLR

	Item	Description
1	Function overview	The system will clear to 0 the bit PI/O turned on or off by a process specified with P0 to P255. Immediately after issuance, the system will go to the next step. The system will only allow the stoppage and reset statuses of the specified process. Note that the system will clear, without checking, the PI/O use status in other processes. The range can be specified with a hyphen (-).
2	Issuance to a non-existent process	The system will turn on the CLR bit of the result display bit of the control box, and then go to the next step (standard HH1FC).
3	Issuance to a process being executed	The system will turn on the CLR bit of the result display bit of the control box, and then go to the next step (standard HH1FC).
4	Issuance to an unstarted process	The system will turn on the CLR bit of the result display bit of the control box, and then go to the next step (standard HH1FC).

[Typical programs with the control box (■)]

1. ■ ACT P1-P5, MRST

   Start a master reset on processes 1 to 5, beginning with step 1, then go to the next step. The started process will cause the parallel timer to continue its measurement when executing the process end or escape and when transiting to a process-executable status.

2. ■ ACT P100, 5, TUP

   Start the zone on process 100, beginning with step 5, then go to the next step. The started process will cause the parallel timer to go into the up position when executing the process end or escape and when transiting a process-executable status.

3. ■ ACT P80, TASK, 3

   Issue an RLEAS macro with regard to CPMS task 80 and issue a QUEUE macro with factor 3, and then go to the next step.

4. ■ RST P10

   Reset process 10, and then go to the next step. When the RST is in the issued status, the parallel timer in the process of measurement will be reset.

5. ■ RST P11, TUP

   Reset process 11, and then go to the next step. When an RST is issued, the parallel timer in the process of measurement will expire.

6. RST P12, TASK

Issue an ABORT macro to CPMS task 12, and then go to the next step.

7. STP P50

Put process 50 into a stopped status, and then go to the next step. When an STP is issued, the parallel/wait timer in the process of measurement will stop.

8. STP P51, TCNT

Put process 51 into a stopped status, and then go to the next step. When an STP is issued, the parallel/wait timer in the process of measurement will continue its measurement.

9. CLR P40

Clear to 0 the bit-type PI/O used in process 40, and then go to the next step.

4.6 Repeat Start and Repeat End

The system will execute the process repeatedly between the repeat start and repeat end. A syntax error will occur if the number of repeat starts is not the same as that of repeat ends in the same route. The system will add an increment to the initial value every time it repeats the process. It will continue repeating until the value becomes larger than the final value. If the initial value is larger than the final one, the system will go to the next step without executing the process between the repeat start and the repeat end. Omitting the increment will result in the increment becoming 1. If the increment is 0, there will be an infinite loop.
The setting range of the initial value, final value, and increments is from 0 to 32767. Setting the range between -32768 and -1 will operate the system on the assumption that it is set between 32768 and 65535.

[Syntax]

CNxxx (Initial value, final value {, increment})
(xxx is a decimal between 000 and 127)

Without syntax

[Typical programs with repeat start ( ) and repeat end ( )]

1.

CN000 (1, 10)

Repeat the process between repeat start and repeat end ten times, then go to the step following the repeat end. Immediately after executing the repeat end, the system will execute the repeat start.

2.

CN127 (1, 5, 2)

Repeat the process between repeat start and repeat end three times, then go to the step following the repeat end.

3.

CN001 (FW000, FW001,FW002)

The value between FW000 and FW002 that was used when the system passed through the repeat start for the first time will be the initial value, final value, or an increment.

4.7 If

The system will judge whether a specific conditional expression is true or false, and then perform a corresponding operation. If the conditional expression holds, the system will execute the portion up to true, , comma (,) and semi-colon (;). If the condition does not hold, the system will execute the portion after false and semi-colon (;). If the system omits the portion after the semi-colon (;), and if the conditional expression does not hold, it will go to the next step. If a label is specified after the comma (,) and semi-colon (;), it will branch to that label.

[Syntax]

◇Conditional expression, { Jump destination label(Bxxx) }
{ Free label }
{ON/OFF} [{:ON/OFF}]
{Assignment expression} {:Assignment expression}
{ACT statement} {:ACT statement}
{STP statement} {:STP statement} ~
{RST statement} {:RST statement}
{CLR statement} {:CLR statement}
{TUP statement} {:TUP statement}
{TRS statement} {:TRS statement}
{PT statement} {:PT statement}]

[{ ; Jump destination label (Bxxx) }]
{ ; Free label }
{;ON/OFF} [{:ON/OFF}]
{Assignment expression} {:Assignment expression}
{;ACT statement} {:ACT statement}
{;STP statement} {:STP statement} ~
{;RST statement} {:RST statement}
{;CLR statement} {:CLR statement}
{;TUP statement} {:TUP statement}
{;TRS statement} {:TRS statement}
{;PT statement} {:PT statement}]]

(xxx is a decimal between 1 and 255.)

<Notice>

The system does not allow branching to another process but does allow branching to another route. However, note that, in actually executing an operation, the system may not function normally in any of the following cases:

- Branching from loop start to the inside of the loop end
- Branching from inside the parallel processing
- Branching into parallel processing
- Branching to the route already being executed

[Typical programs with if (◇)]

1. ◇ X000, B1;LABEL

   When X000 is ON, jump to a step where a B1 label is present. When it is OFF, the system will jump to the step following the place where a LABEL label is present.

2. ◇ H0<> (YW000&H3000), ON Q005

   If the logical product of YW000 and H3000 is not 0, turn ON Q005. If it is 0, do nothing and go to the next step.

3. ◇ Q000, FW100=FW100+1;ACT P10

   If Q000 is ON, add 1 to FW000 and go to the next step. If it is OFF, conduct an ACT start on process 10 and go to the next step.

4. ◇ GW000=4, STP P6:RST P7;EW000=8:ON J000

   When GW000 is 4, stop process 6, reset process 7, and go to the next step.
   When GW000 is not 4, set EW to 8, turn on J000, and go to the next step.

5. ◇ X010, ON J000, J001, J002, J003;ERRLB

   When X010 is ON, turn ON J000, J001, J002, and J003 and go to the next step. When X010 is OFF, jump to the step following the place where an ERRLB label is present.

4.8 Jump

The system will branch unconditionally to a specified label in the process. The system allows you to specify labels from B1 to B255 for each process. HI-FLOW specifies free labels (which must be up to 6 characters and which you are free to name and can add only to an entity other than steps).

[Syntax]

↳ { Jump destination label (Bxxx) }
{ Free label }

<Notice>

The system does not allow branching to another process but does allow branching to another route. However, note that, when actually executing an operation, the system may not function correctly in any of the following cases:

- Branching from the loop start to the inside of the loop end
- Branching from inside the parallel processing
- Branching into parallel processing
- Branching a route already being executed

[Typical programs with jump (↳)]

1. ↳ B1

   Jump to a step where a B1 label is present, then execute the operation immediately, beginning with that step.

2. ↳ ERRBLK

   Jump to the step following the place where a LABEL label is present, then execute the operation immediately, beginning with that step.

4.9 Escape

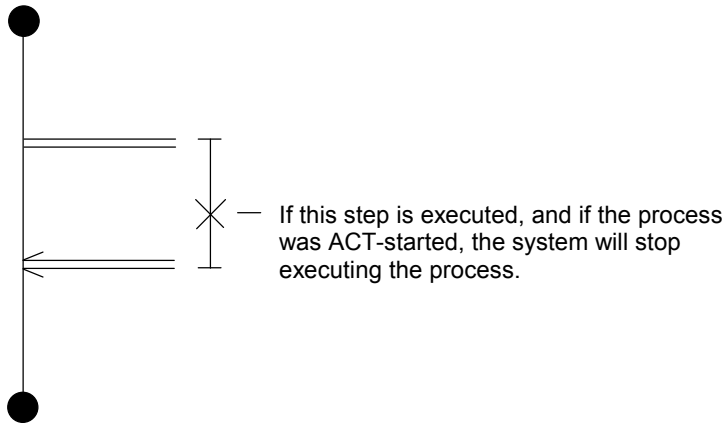
The system will shut down its own process.
If it is the main process, the system will shut down all routes and transit to an executable status. At that time, if a process (or processes) is being called, the system will make all of them escape. The timers in the system's own process are used in the same way as when the system is started up (TUP and TRS options).
Subprocesses are basically handled in the same way as the main process. The system will restore the executed place to the main process with the same scan.
When started by master reset, the system will clear to 0 the bit-type PI/O to be turned on by its own process (ON statement and parallel timer).

[Syntax]

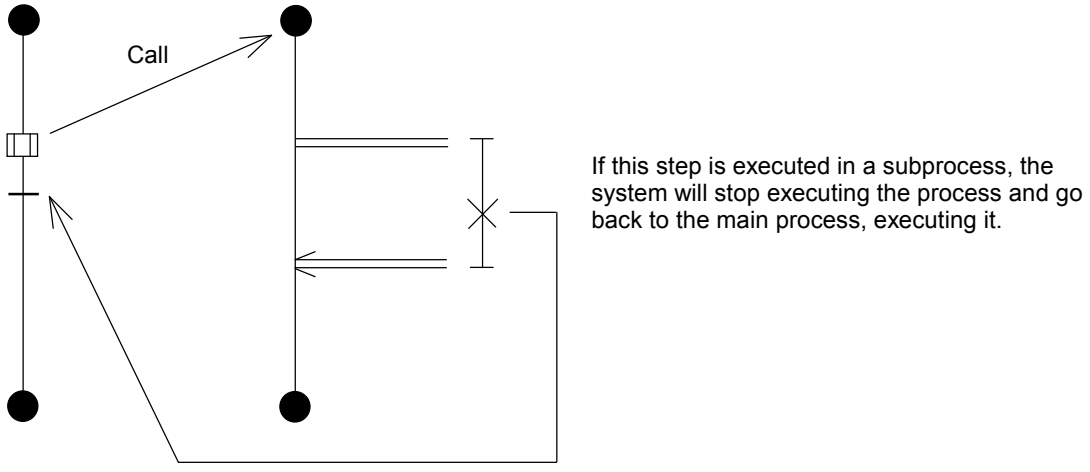
Without syntax

[Typical programs with escape ( )]

1.



2.



4.10 Para Start and Para End

A pair consisting of para start and para end represents a portion to be synchronized.
The para start will start a synchronized subroute, and then go to the step following the system's own route.
The para end indicates that the system will execute the step following its own route after all merging routes are finished.
In conventional practice, the system used to monitor the end of the subroute where the para end merged (that is, the main route was being executed), so that the execution of the next step is delayed by one scan. The present redesign ensures that both the para and the route ends check if each of them merged at the last, and that the system executes the next step of the place where the main route merges if it is the last, and will end the execution of its own route if not the last (the main route is not necessarily being executed). This prevents scan delays.

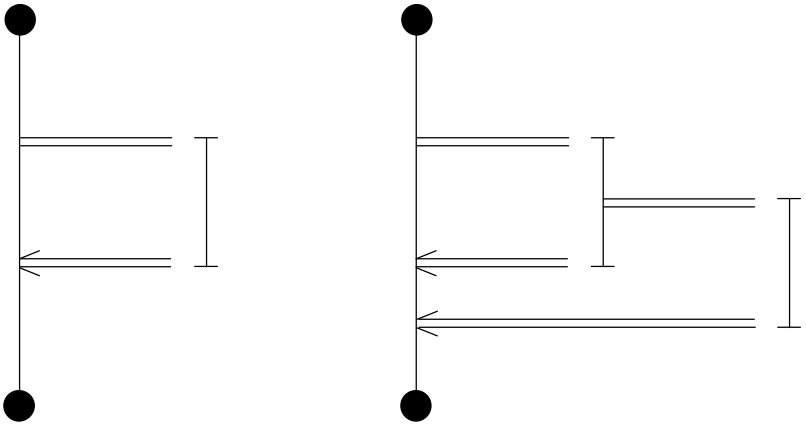
[Syntax]

Without syntax

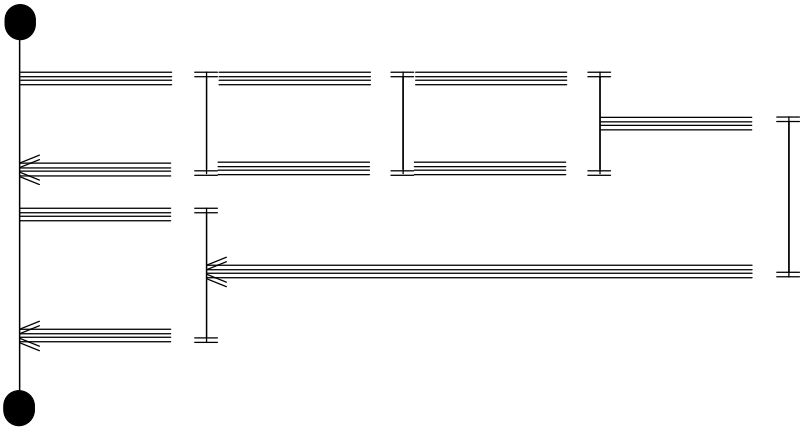
Without syntax

[Typical programs with para start ( ) and para end ( )]

1.



2.



4.11 Select, Cell Wait, and Select End

A set of select, cell wait, and select end represents a portion of selective branching.
The select will start the selective branching route, and then go to the cell wait of the system's own route. (The select and the cell wait or the route start and the cell wait must be consecutive.)
The cell wait will end the execution of another route when the conditional expression of the system's own route holds, and will go to the step following the system's own route. The present redesign is such that, when a subroute is selected, the system will terminate the main route (will only execute the route selected).
The system will check the conditional expression from the left route of the screen, so that, if more than one condition holds with the same scan, the system will select the route at the extreme left.
In conventional practice, the system used to monitor the end of the subroute where the select end merged (that is, the main route was being executed). As a result, the execution of the next step is delayed by one scan. The present redesign ensures that, when a subroute is selected, the route end of that route will start the main route and execute the step following the merging portion, thus resolving the one-scan delay. Furthermore, in conventional practice, the select end and select has to be present in the same route. The present redesign is such that they do not have to be present in the same route (do not have to merge into the source route).

<Notice>
The cell wait must be at the step following the select.

[Syntax]

Without syntax

Conditional expression [, timer, output bit]

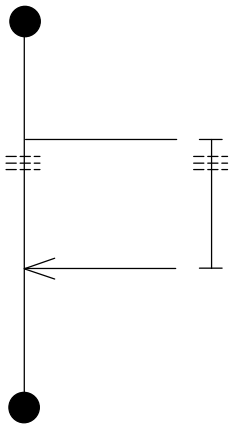
Without syntax

- Timer
- Output bit

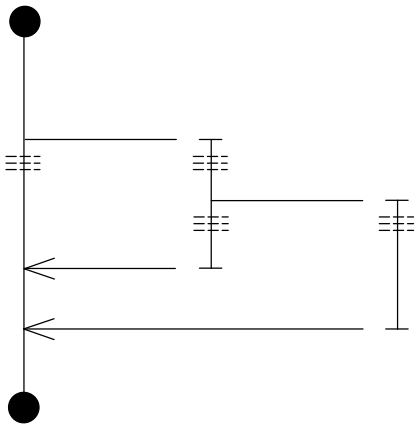
* For timers and output bits, see "4.3 Wait."

[Typical programs with select ( ), cell wait ( ), and select end ( )]

1.



2.



4.12 Multi-entry

When a conditional expression is configured in the same figure as the select end, the system will regard it as a multi-entry.
When a process is being executed, and when a conditional expression holds, the system will re-execute the operation beginning with the step where the multi-entry is present. (Even when executing the process for the first time, the system will begin with that process when the conditional expression holds.) A check of the conditional expression is conducted at the first point of the scan and the system may be delayed by up to one scan.
The system will begin with the smallest-step condition when conducting a check. When more than one condition holds in the same scan, the system will re-execute the operation, beginning with the step having the smallest step number.
A multi-entry can be configured at the subroute.
When conditions hold and the system executes an operation, it will initialize all routes other than those equipped with timers (PT and WT), counter (CN), called process, and multi-entry. But, it will hold the PI/O value.

[Syntax]

⊧ Conditional expression

<Notice>

- Note that configuring a multi-entry inside the loop end at the loop start may cause the system to malfunction.
- The system does not allow you to configure a multi-entry in the subroute of a synchronization syntax.

[Typical programs with multi-entry (⊧)]

1. ⊧ X000

   Re-execute the operation, beginning with this step, when X000 is ON.

2. ⊧ GW000<H2000

   When GW000 is smaller than H2000, re-execute the operation, beginning with this step.

4.13 Call

The system can conduct a subroutine call for a process specified by P0 through P255. The [, step number] option starts executing the operation, beginning with a specified step. (Omitting it will cause the system to begin with the process start.)
If no process is specified, no step is specified, or if the system calls its own process, the system will turn on the CALL bit of the result display bit of the control box, and then go to the next step.
When a specified process is already being executed, the system will continue to wait until it can call the process (shift to an executable status). It will ACT-start and can call the process being reset.
The system can call another process with the subprocess, and can nest up to 16 of them.
The [,MRST] option conducts a master reset call. When a master reset call is made, the system will clear to 0 the bit PI/O turned on by its own process when the system terminates the call process, executes the escape, and shifts to an executable status.

The [, TUP] option will cause the parallel timer occupied by the system's own process to become up when the system executes the process end or escape and when it shifts to an execution-specifiable status.
The [, TRS] option resets the parallel timer occupied by the system's own process when the system executes the process end and escape and when it shifts to an executable status. If no such thing is specified, the system will continue to measure its parallel timer after the end of the process.

[Syntax]

Pxxx [,Step number] [, MRST] { [, TUP] }
{ [, TUP] }

[Typical programs with call ( )]

1. P1

   The system will make a zone call on process 1, from step 1. The process called will cause the parallel timer to continue its measurement when the system executes the process end and escape and when it shifts to a process executable status.

2. P2, 5, MRST

   The system makes a master reset call on process 2 from step 5. The process called will cause the parallel timer to continue its measurement when the system executes the process end and escape and when it shifts to a process executable status.

3. P3, TUP

   The system will make a zone call on process 3 from step 1. The process called will cause the parallel timer to become up when the system executes the process end and escape and when it transits to a process executable status.

4.14 Function

This function is designed to complement the function of operation and data processing supported by the box. For details, see Chapter 5.

[Syntax]

Name of applied instruction [,Parameter] ~

4.15 Wait with Precondition

Wait remains the same until the conditions for shifting hold. After the conditions hold, and before the system goes to the next step, and if the last step is an ON statement or a process call, the system first turns off the PI/O before continuing to the next step. The system will go on without doing anything if the last step is not an ON statement or a process call. (It is the same as a wait.)
Note that the system will not clear the last condition of the source of the branch if it begins with this step by branching. This function is for conforming to the SFC standards.

[Syntax]

{ Conditional expression }
{ WTxxx (formula [, SB] [,Conditional expression]) }

4.16 Motion

This feature is provided to facilitate motion control under HI-FLOW. For details, see Chapter 6.

[Syntax]

Name of applied instruction

Common parameter
Axis number(s) [,Axis number(s)] ~
Parameters for the axis (or axes) [,Parameters for the axis (or axes)] ~

4.17 Non-synchronous Process End

A non-synchronous process end, used in conjunction with a process start, causes the process to be terminated without waiting for any of the given non-synchronous branching routes to reach their ends. To make a non-synchronous process end function asynchronously with all of the given non-synchronous routes, use the non-synchronous process end and non-synchronous routes in such a way that the latter do not merge to the main route -- the route from which they have branched -- at its route end.

If the above process is initially started by the ACT statement at its process start, then its main route proceeds until it reaches the process end, and, in the meantime, the given non-synchronous routes are started as requested. When the main route reaches the process end, the HI-FLOW system terminates the process even if any one or ones of the given non-synchronous routes have not reached their ends. Then, the process is started again and continues as far as the route start of one of the non-synchronous routes that was previously on the way to its route end. At the route start, the HI-FLOW system checks if the non-synchronous route has reached its route end. If not, the HI-FLOW system does not start it again.

If the above process is initially started by calling as a subroutine, then it is not terminated immediately when its main route reaches the process end, but instead it is placed in a wait state until all of the non-synchronous routes reach their ends, as in cases where the main route is ended by a (synchronous) process end.

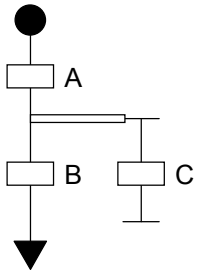
[Syntax]

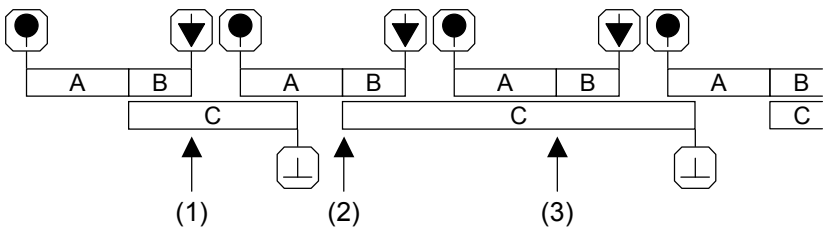
Without syntax

[Sample programs]

1. Sample program using only one non-synchronous route

<Sample circuit> <Timing chart>

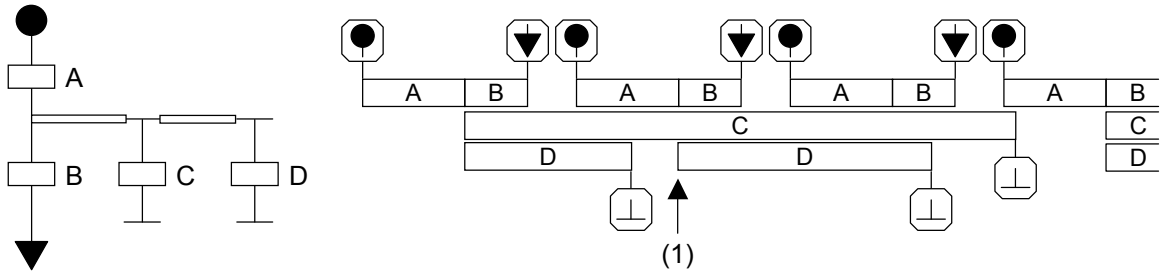




(1) When the main route reaches its non-synchronous process end, the process is terminated, but the non-synchronous route is left undisturbed because it is still on the way to its route end.
(2) Because the non-synchronous route has already reached its end, it is started again at its route start.
(3) Although the process has reached the route start of the non-synchronous route, the non-synchronous route is left undisturbed because it is still on the way to its route end.

2. Sample program using two non-synchronous routes

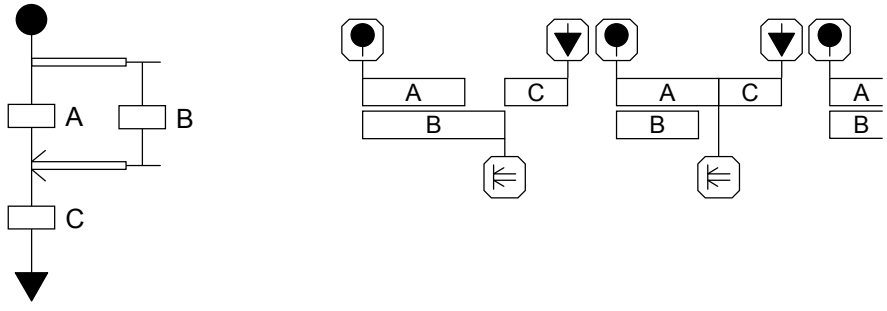
<Sample circuit> <Timing chart>



(1) When the process again reaches the route start of the non-synchronous route crossing through box D, it starts that non-synchronous route because it has already reached its route end. However, the non-synchronous route crossing through box C is left undisturbed because it is still on the way to its route end.

3. Sample program using both a main route ended by non-synchronous process end and a subroute with its route end merging to that main route

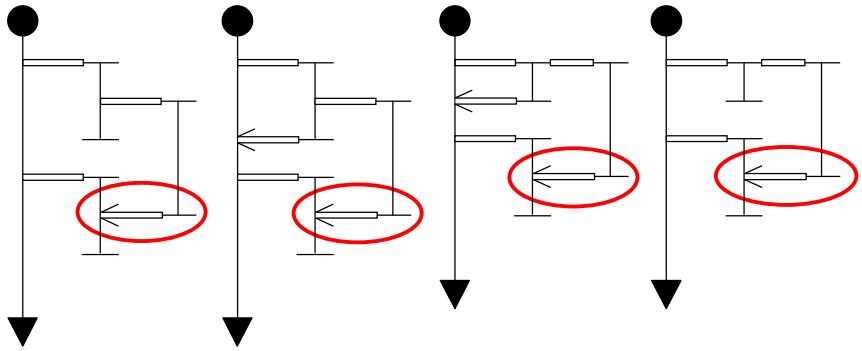
<Sample circuit> <Timing chart>



The effect of this sample program is the same as that of a program using a main route ended by (synchronous) process end. In the above sample program, when the main route and subroute proceed to the para end normally, the main route proceeds further to the non-synchronous process end.

<Notice>

(1) The programming patterns shown below are not supported, in each of which a subroute merges to a non-synchronous route. If such a programming pattern is executed and an attempt is made to start the merging subroute a first time, the attempt will fail. The reason for this is that, while the non-synchronous route is on the way to its route end, the merging subroute is also considered to be on the way to its route end.



(2) If a process ended by non-synchronous process end is transmitted to a HI-FLOW system version not supporting the non-synchronous process end (Ver-Rev 02-03 or earlier), that version will receive it successfully, but it is not capable of displaying it on-screen. In addition, if an attempt is made to transmit the received process to the PCs, it will fail. This is because the unsupported feature will be detected as an error during compilation, which is usually made before such an attempt. In these cases, the non-synchronous process end could be changed to a (synchronous) process end to solve the problem; however, if it is so changed, the non-synchronous route(s) will not work as originally intended.

(3) If a master reset specification is given at the start of a process, and bit-type PI/O data is used, then the bit-type PI/O data may be zero-cleared.

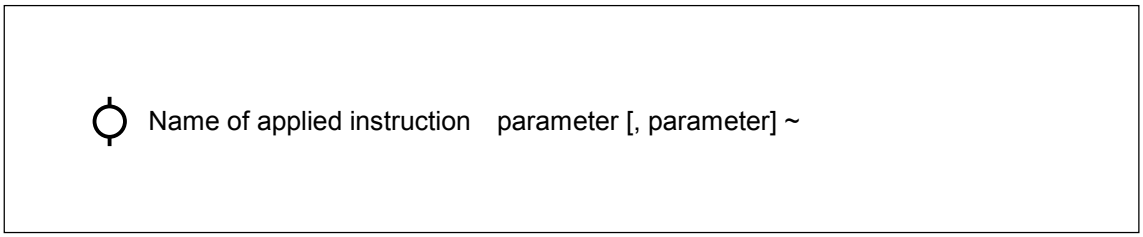
5 APPLIED INSTRUCTIONS

5.1 Overview

The function of operation and that of data processing supported by HI-FLOW language syntax are four operations, logical operations, and assignment only (word length only). The PC HI-FLOW then supports the applied instructions of functions similarly to the ladder diagram.

5.2 How to Use It

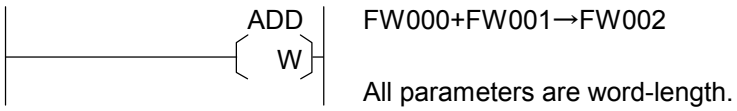
Applied instructions are programmed as follows:



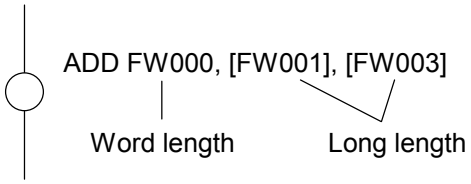
5.3 Parameters

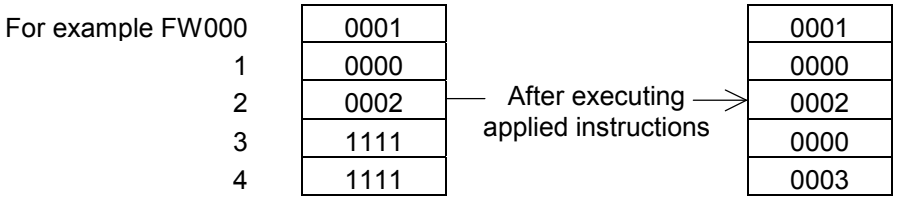
In applied instructions of HI-FLOW, each applied instruction and its applicable parameter type do not have to correspond, unlike the operation function of the ladder.

- Ladder



- HI-FLOW





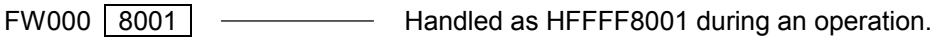
Parameters generally come in three categories: source, destination, and result. They are expressed as S, D, and R respectively.
Parameters come in three categories: bit-type PI/O, word-type PI/O, and constant.
For applied instructions of HI-FLOW, the system allows you to specify an addressing mode for parameters. Addressing modes come in four categories as listed below.

1. Specification of direct word length: Describes it just like the parameter.
2. Specification of direct long length: Encloses the parameter in [ ] (brackets).
3. Specification of indirect word length: Adds @ before the description of 1.
4. Specification of indirect long length: Adds @ before the description of 2.

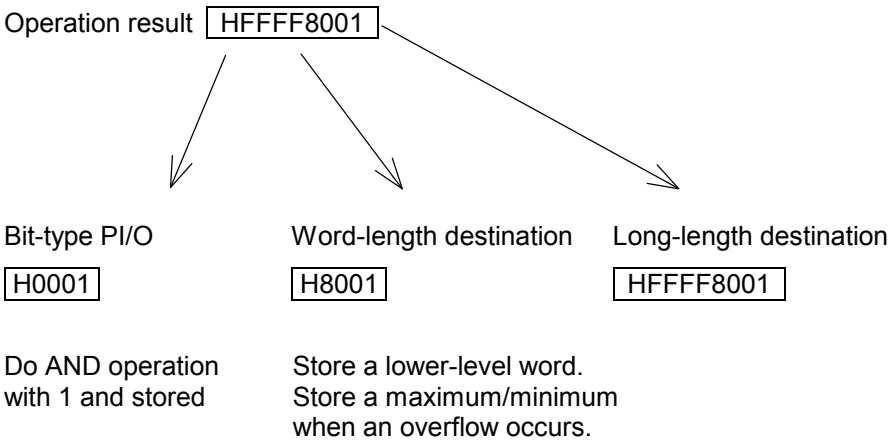
Addressing mode	Parameters		
	Bit-type PI/O	Word-type PI/O	Constant
	X000 Data 1 X001 Data 2 Data 1, 2 Data a Data b	FW000 Data 3 FW001 Data 4 Data 3, 4 Data c Data d	XXXX YYYYYYYY XXXX Data e Data f YYYYYYYY Data g Data h
1. Direct word length	AND result of data 1 and 1	Data 3	XXXX, but long-length YYYYYYYY is a low-level word only.
Example	X000	FW000	1230 H20000000
2. Direct long length	AND result of data 1, 2 and 1	Data 3, 4	XXXX, YYYYYYYY However, XXXX is handled as a long-length.
Example	[X000]	[FW000]	[H1234] [H20000000]
3. Indirect word length	Parameter error	Data c However, an error occurs when data 3,4 is an odd number.	For XXXX, data e. For YYYYYYYY, data g. If XXXX and YYYYYYYY are odd numbers, it is an error.
Example		@FW000	@HFFF0 @H180000
4. Indirect long length	Parameter error	Data c, d However, an error occurs when data 3,4 is an odd number.	For XXXX, data e, f. For YYYYYYYY, data g, h. If XXXX and YYYYYYYY are odd numbers, it is an error.
Example		@[FW000]	@[HFFF0] @[H180000]

5.4 Type Conversion in Operations

When the system takes in a parameter value for performing an operation, it expands all their codes to long-lengths.



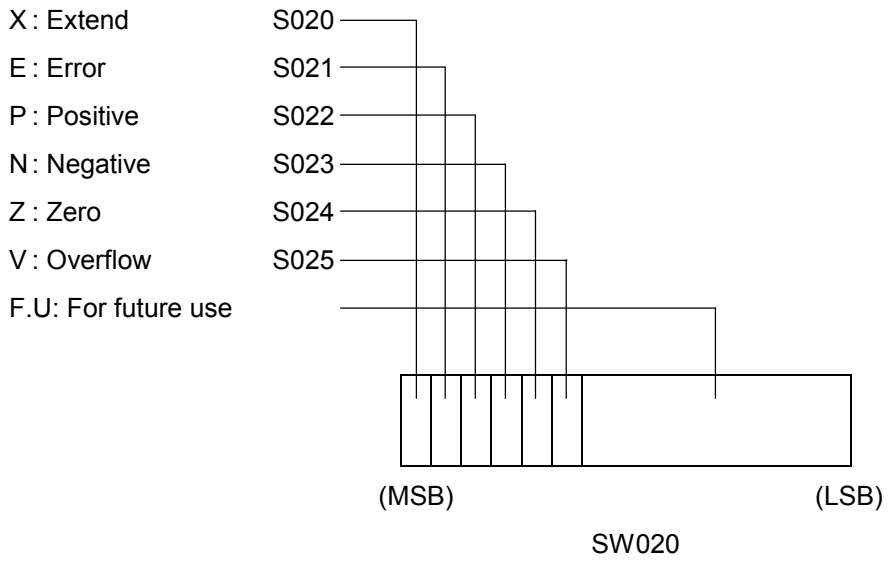
When storing an operation result, the system converts the type of the result according to the destination.



5.5 System Error Flags

Various flags are set to SW020 according to the execution results of applied instructions of HI-FLOW.

Flag types



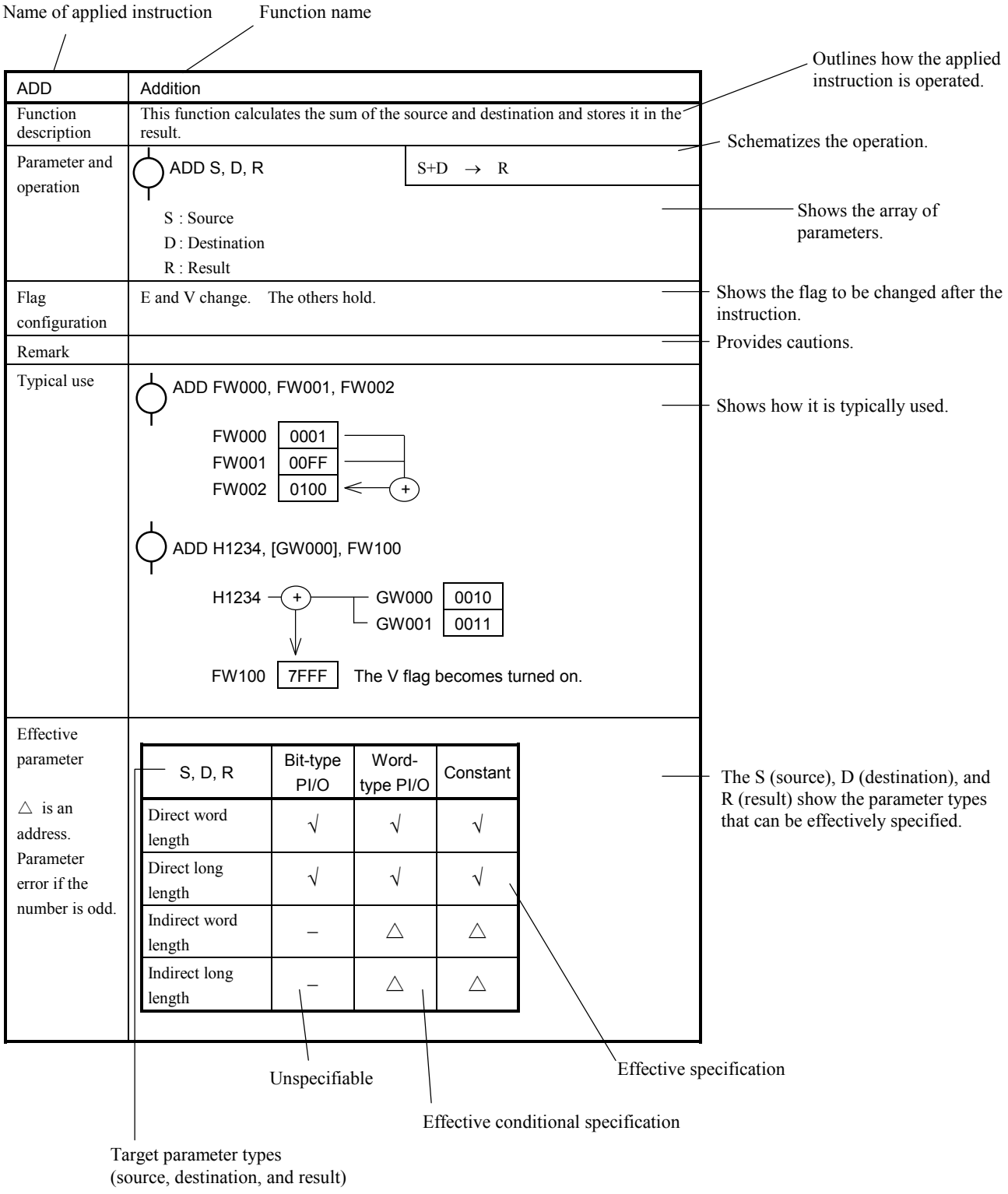
Each flag is configured according to the configuration conditions of a flag for each applied instruction. However, when the conditions listed below hold, the flags specified below are configured in common with all applied instructions.

- Error flags ..... When the number of parameters of applied instructions used differs
  When the CPU is memory-protected, and when the address and PI/O specified by result (R) indicate the inside of the protect area
  When a specified PI/O is defective (such as when it is unserviceable)

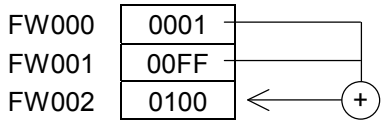
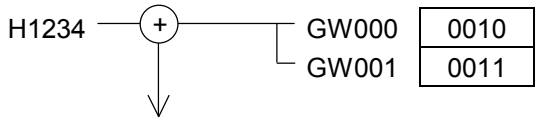
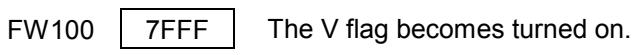
- Overflow flags.... When an operation result exceeds the range (word or long) specified by result (R). The operation result specifies the limit value of each size.
  Word length: Positive overflow/7FFF
  Negative overflow/8000
  Long length: Positive overflow/7FFFFFFF
  Negative overflow/80000000

5.6 Function Description

This section specifies the applied instructions. Here is the way they will be described.



ADD	Addition
Function description	This function calculates the sum of the source and destination and stores it in the result.
Parameter and operation	ADD S, D, R S+D → R S : Source D : Destination R : Result
Flag configuration	The E and V will change. The others will become turned off.
Remark	

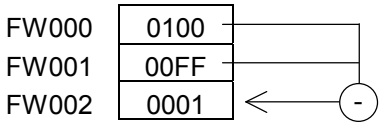
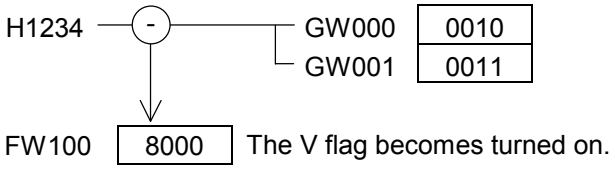
Typical use	ADD FW000, FW001, FW002
	 
	
	ADD H1234, [GW000], FW100
	 
	
	 

Effective parameter △ is an address. Parameter error if the number is odd.	

S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

SUB	Subtraction
Function description	This function subtracts the contents of the destination from the source and stores it in the result.
Parameter and operation	SUB S, D, R S-D → R S : Source D : Destination R : Result
Flag configuration	The E and V will change. The others will become turned off.
Remark	

Typical use	SUB FW000, FW001, FW002
	 
	
	SUB H1234, [GW000], FW100
	 
	
	The V flag becomes turned on.

Effective parameter △ is an address. Parameter error if the number is odd.	

S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

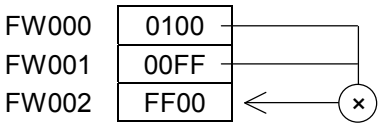
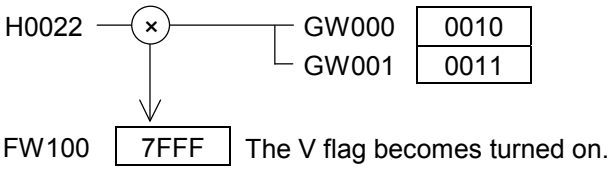
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

<table>
<tr><td>INC</td><td>+1 (Increment)</td></tr>
<tr><td>Function description</td><td>This function adds 1 to the contents of the source.</td></tr>
<tr><td>Parameter and operation</td><td>INC S
S : Source
S+1 → S</td></tr>
<tr><td>Flag configuration</td><td>The E and V will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td></td></tr>
<tr><td>Typical use</td><td>INC FW000
FW000 +1
INC [GW000]
GW000
GW001 +1
The system will increment GW000 and GW001,
regarding them as long variables.</td></tr>
<tr><td>Effective parameter
△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>DEC</td><td>-1 (decrement)</td></tr>
<tr><td>Function description</td><td>This function subtracts 1 from the contents of the source.</td></tr>
<tr><td>Parameter and operation</td><td>DEC S
S : Source
S-1 → S</td></tr>
<tr><td>Flag configuration</td><td>The E and V will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td></td></tr>
<tr><td>Typical use</td><td>DEC FW000
FW000 -1
DEC [GW000]
GW000
GW001 -1
The system will decrement GW000 and GW001,
regarding them as long variables.</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

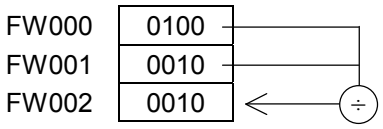
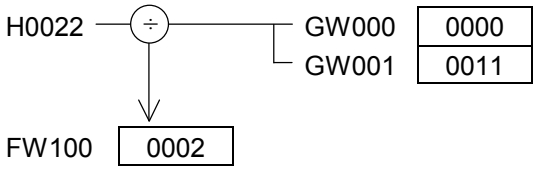
MUL	Multiplication
Function description	This function multiplies the contents of the source and destination and stores them in the result.
Parameter and operation	MUL S, D, R S×D → R S : Source D : Destination R : Result
Flag configuration	The E and V will change. The others will become turned off.
Remark	

Typical use	MUL FW000, FW001, FW002
	 
	MUL H22, [GW000], FW100
	 
	The V flag becomes turned on.

Effective parameter △ is an address. Parameter error if the number is odd.	

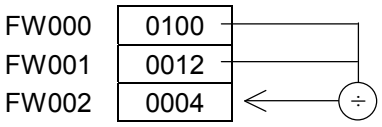
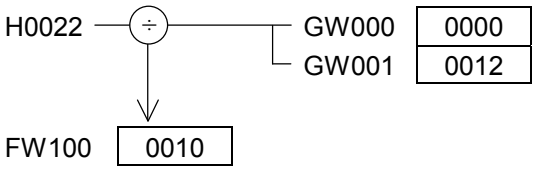
S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

DIV	Division
Function description	This function divides the source by the contents of the destination and stores the quotient in the result.
Parameter and operation	DIV S, D, R S÷D → R S : Source D : Destination R : Result
Flag configuration	The E and V will change. The others will become turned off.
Remark	When D = 0, the system will turn on the E flag and do nothing.
Typical use	DIV FW000, FW001, FW002  DIV H22, [GW000], FW100 
Effective parameter △ is an address. Parameter error if the number is odd.	

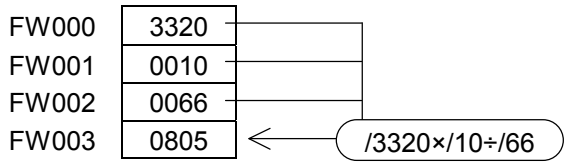
S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

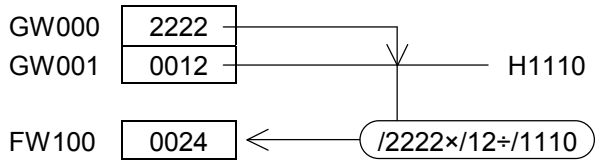
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

MOD	Remainder
Function description	This function divides the source by the contents of the destination and stores the remainder in the result.
Parameter and operation	MOD S, D, R Remainder of S/D → R S : Source D : Destination R : Result
Flag configuration	The E and V will change. The others will become turned off.
Remark	When D = 0, the system will turn on the E flag and do nothing. R = 0 when overflowed.
Typical use	MOD FW000, FW001, FW002  MOD H22, [GW000], FW100 
Effective parameter △ is an address. Parameter error if the number is odd.	

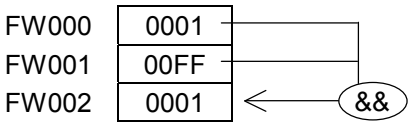
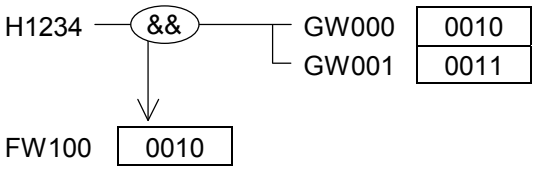
S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

<table>
<tr><td>SCL</td><td>Scale conversion</td></tr>
<tr><td>Function description</td><td>This function converts the scale of the source by the contents of the destination and stores it in the result.</td></tr>
<tr><td>Parameter and operation</td><td>SCL S, D1, D2, R
S×D1÷D2 → R
S : Source
D1: Destination 1 D2: Destination 2
R : Result</td></tr>
<tr><td>Flag configuration</td><td>The E and V will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When a multiplication overflow occurs, the system will write the overflow value in the result and terminate the writing. When D2 = 0, the system will turn on the E flag and do nothing. R = 0 when overflowed.</td></tr>
<tr><td>Typical use</td><td>SCL FW000, FW001, FW002, FW003
 

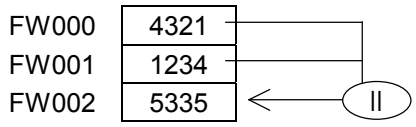
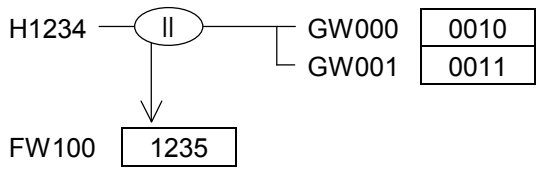
SCL GW000, GW001, H1110, FW100
 
</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S, D1, D2</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

AND	Logical product
Function description	This function stores in the result the logical product of the source and the contents of the destination.
Parameter and operation	AND S, D, R S && D → R S : Source D : Destination R : Result
Flag configuration	The E will change. The others will become turned off.
Remark	When the R is word length, the system will write a lower-level word of the operation result.
Typical use	AND FW000, FW001, FW002  AND H1234, [GW000], FW100 
Effective parameter △ is an address. Parameter error if the number is odd.	

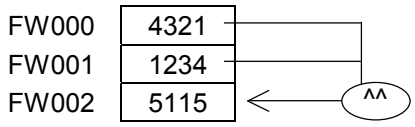
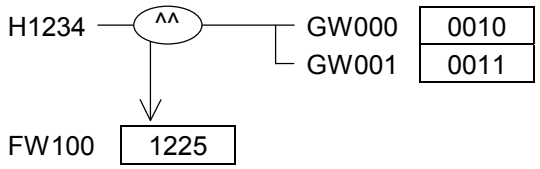
S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

OR	Logical sum
Function description	This function stores in the result the logical addition of the source and the contents of the destination.
Parameter and operation	OR S, D, R S ‖ D → R S : Source D : Destination R : Result
Flag configuration	The E will change. The others will become turned off.
Remark	When the R is word length, the system will write a lower-level word of the operation result.
Typical use	OR FW000, FW001, FW002  OR H1234, [GW000], FW100 
Effective parameter △ is an address. Parameter error if the number is odd.	

S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

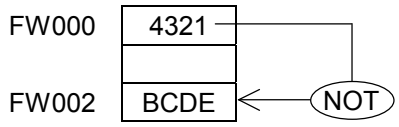
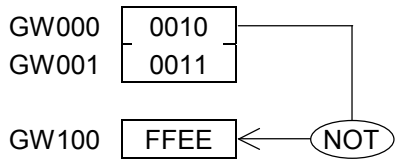
EOR	Exclusive OR
Function description	This function stores in the result the exclusive OR of the source and the contents of the destination.
Parameter and operation	EOR S, D, R S ^^ D → R S : Source D : Destination R : Result
Flag configuration	The E will change. The others will become turned off.
Remark	When the R is word length, the system will write a lower-level word of the operation result.
Typical use	EOR FW000, FW001, FW002  EOR H1234, [GW000], FW100 

Effective parameter

△ is an address. Parameter error if the number is odd.

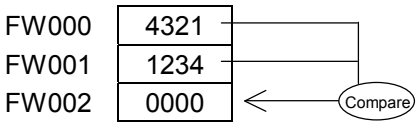
S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

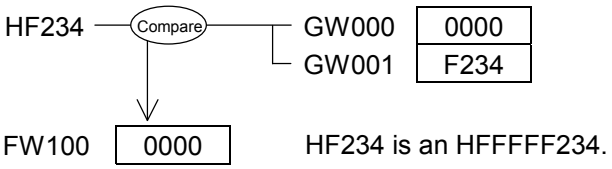
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

NOT	Negation
Function description	This function stores in the result the negation (bit reversion) of the contents of the source.
Parameter and operation	NOT S, R S (bit reversion) → R S : Source D : Destination
Flag configuration	The E will change. The others will become turned off.
Remark	
Typical use	NOT FW000, FW002  NOT [GW000], FW100 
Effective parameter △ is an address. Parameter error if the number is odd.	

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

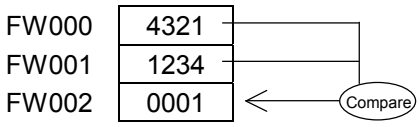
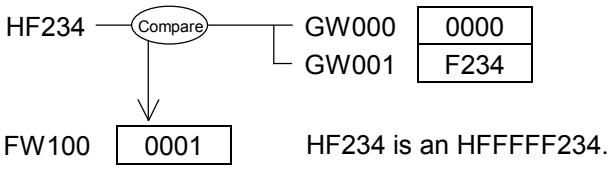
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

<table>
<tr><td>EQU</td><td>Compare to see if equal</td></tr>
<tr><td>Function description</td><td>This function compares the source with the contents of the destination. If they are equal, this function stores a 1 in the result. If not, it stores a 0.</td></tr>
<tr><td>Parameter and operation</td><td>EQU S, D, R
S : Source
D : Destination
R : Result
S = D 1 → R
S ≠ D 0 → R</td></tr>
<tr><td>Flag configuration</td><td>The E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>Word-length data is code-extended to long length and compared.</td></tr>
<tr><td>Typical use</td><td>EQU FW000, FW001, FW002
 

EQU HF234, [GW000], FW100
 

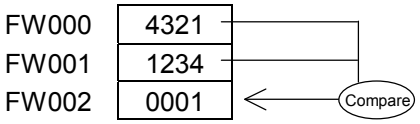
HF234 is an HFFFFF234.</td></tr>
<tr><td>Effective parameter

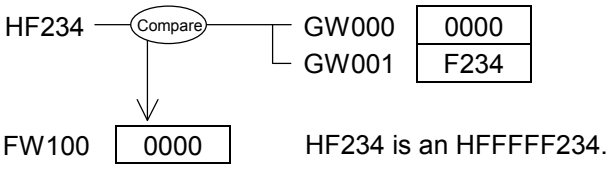
△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S, D</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

NEQ	Compare to see if unequal
Function description	This function compares the source with the contents of the destination. If they are equal, this function stores a 1 in the result. If not, it stores a 0.
Parameter and operation	NEQ S, D, R S : Source D : Destination R : Result S ≠ D 1 → R S = D 0 → R
Flag configuration	The E will change. The others will become turned off.
Remark	Word-length data is code-extended to long length and compared.
Typical use	NEQ FW000, FW001, FW002  NEQ HF234, [GW000], FW100  HF234 is an HFFFFF234.
Effective parameter △ is an address. Parameter error if the number is odd.	

S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

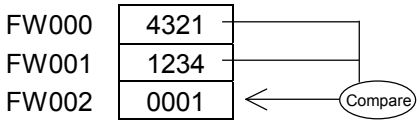
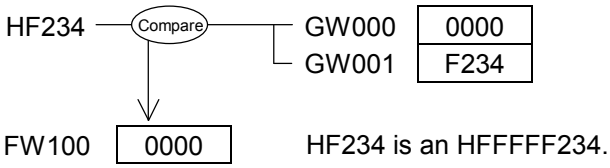
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

<table>
<tr><td>GT</td><td>Compare to see if larger</td></tr>
<tr><td>Function description</td><td>This function compares the source with the contents of the destination. If the source is larger, this function stores a 1 in the result. If not, it stores a 0.</td></tr>
<tr><td>Parameter and operation</td><td>GT S, D, R
S : Source
D : Destination
R : Result
S > D 1 → R
S ≤ D 0 → R</td></tr>
<tr><td>Flag configuration</td><td>The E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>Word-length data is code-extended to long length and compared.</td></tr>
<tr><td>Typical use</td><td>GT FW000, FW001, FW002
 

GT H1234, [GW000], FW100
 

HF234 is an HFFFFF234.</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S, D</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

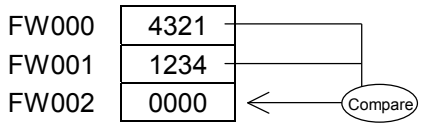
GE	Compare to see if equal or larger
Function description	This function compares the source with the contents of the destination. If the source is equal or larger, this function stores a 1 in the result. If not, it stores a 0.
Parameter and operation	GE S, D, R S : Source D : Destination R : Result S ≥ D 1 → R S < D 0 → R
Flag configuration	The E will change. The others will become turned off.
Remark	Word-length data is code-extended to long length and compared.
Typical use	GE FW000, FW001, FW002  GE H1234, [GW000], FW100  HF234 is an HFFFFF234.
Effective parameter △ is an address. Parameter error if the number is odd.	

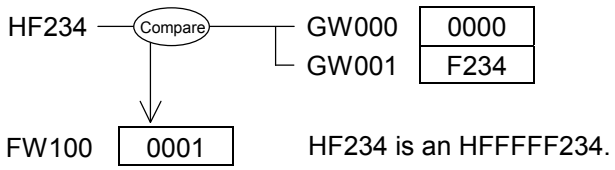
S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

<table>
<tr><td>LT</td><td>Compare to see if smaller</td></tr>
<tr><td>Function description</td><td>This function compares the source with the contents of the destination. If the source is smaller, this function stores a 1 in the result. If not, it stores a 0.</td></tr>
<tr><td>Parameter and operation</td><td>LT S, D, R

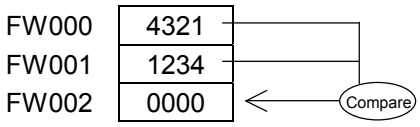
S : Source
D : Destination
R : Result

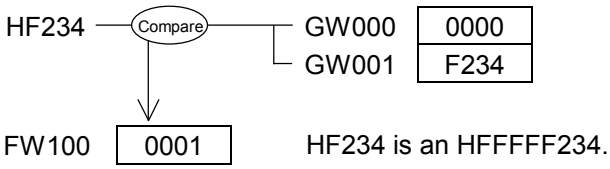
S < D 1 → R
S ≥ D 0 → R</td></tr>
<tr><td>Flag configuration</td><td>The E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>Word-length data is code-extended to long length and compared.</td></tr>
<tr><td>Typical use</td><td>LT FW000, FW001, FW002
 

LT H1234, [GW000], FW100
 

HF234 is an HFFFFF234.</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S, D</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>LE</td><td>Compare to see if equal or smaller</td></tr>
<tr><td>Function description</td><td>This function compares the source with the contents of the destination. If the source is equal or smaller, this function stores a 1 in the result. If not, it stores a 0.</td></tr>
<tr><td>Parameter and operation</td><td>LE S, D, R
S : Source
D : Destination
R : Result
S ≤ D 1 → R
S > D 0 → R</td></tr>
<tr><td>Flag configuration</td><td>The E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>Word-length data is code-extended to long length and compared.</td></tr>
<tr><td>Typical use</td><td>LE FW000, FW001, FW002
 

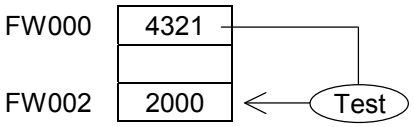
LE H1234, [GW000], FW100
 

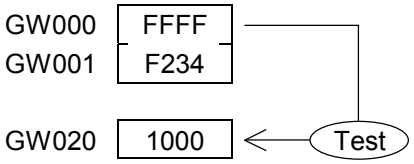
HF234 is an HFFFFF234.</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S, D</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

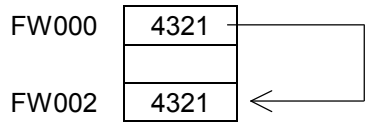
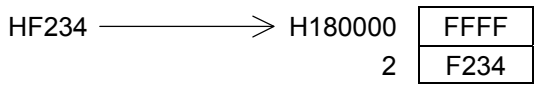
<table>
<tr><td>TST</td><td>Code test</td></tr>
<tr><td>Function description</td><td>This function tests the contents of the source and configures the flags P, Z, and N.</td></tr>
<tr><td>Parameter and operation</td><td>TST S

S : Source

S > 0 : P=1, Z=0, N=0
S = 0 : P=0, Z=1, N=0
S < 0 : P=0, Z=0, N=1</td></tr>
<tr><td>Flag configuration</td><td>The E, P, Z and N will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>Word-length data is code-extended to long length and tested.</td></tr>
<tr><td>Typical use</td><td>TST FW000
 

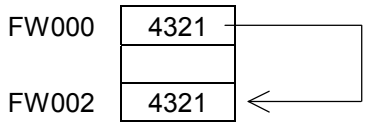
TST [GW000]
 
</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

MOV	Transfer
Function description	This function transfers the contents of the source to the destination.
Parameter and operation	MOV S, D S → D S : Source D : Destination
Flag configuration	The E will change. The others will become turned off.
Remark	If sizes differ in transfer, the system will convert the type.
Typical use	MOV FW000, FW002  MOV HF234, @ [H180000]  HF234 is an HFFFFF234.
Effective parameter △ is an address. Parameter error if the number is odd.	

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

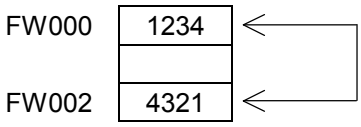
D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

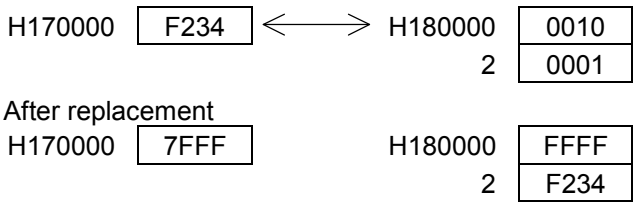
MOM	Collective transfer
Function description	This function transfers the contents of the source to the destination by sending n elements worth (word and long) collectively.
Parameter and operation	MOM S, n, D S : Source D : Destination n : Number of elements to be transferred S1 → D1 ⋮ Sn → Dn
Flag configuration	The E and V will change. The others will become turned off.
Remark	The system will conduct no operation when n ≤ 0 and n > 256. If S is a constant, the system will convert the constant to D type and configure it as such. If S and D are different in type, the system will convert their types and configure them as such.
Typical use	MOM FW000, 1, FW002  MOM FW000, 2, @ [H180000]  HF234 is an HFFFFF234, H0001 is an H00000001.
Effective parameter △ is an address. Parameter error if the number is odd.	

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

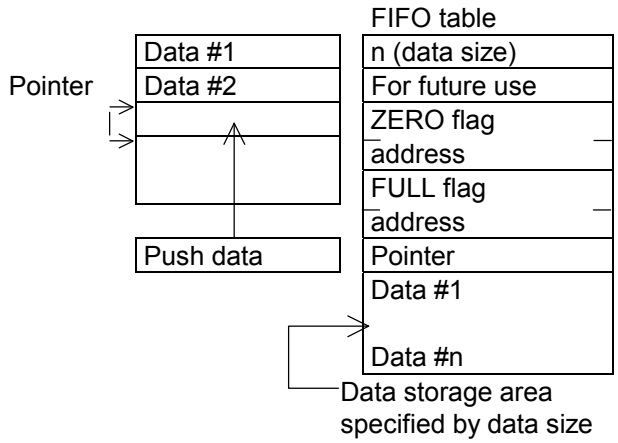
<table>
<tr><td>EXC</td><td>Replacement</td></tr>
<tr><td>Function description</td><td>This function replaces the contents of the source with the destination.</td></tr>
<tr><td>Parameter and operation</td><td>EXC S, D
S ←→ D

S : Source
D : Destination</td></tr>
<tr><td>Flag configuration</td><td>The E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>If sizes differ in transfer, the system will convert their types and replace them.</td></tr>
<tr><td>Typical use</td><td>EXC FW000, FW002
 

EXC @H170000, @ [H180000]
 
</td></tr>
<tr><td>Effective parameter

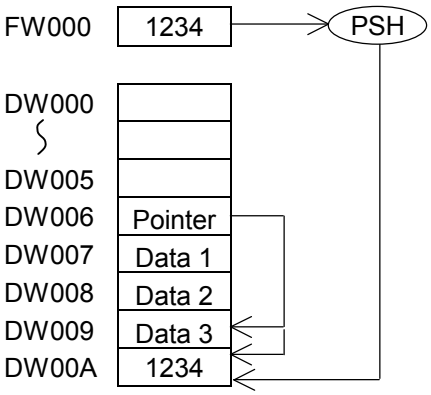
△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S, D</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

PSH	FIFO write
Function description	This function pushes the contents of the source to the FIFO table. The data length of the FIFO table is word only.
Parameter and operation	PSH S, TB S: Source TB: Starting address of the FIFO table
Flag configuration	The E will change. The others will become turned off.
Remark	The system will perform no operation when n ≤ 0 and n > 256. It will perform no operation when the pointer < 0 or the data size < pointer. When the pointer = data size, the system will turn on the FULL flag and perform no operation. After the push, and when the system increments the pointer and it becomes n, the FULL flag will be turned on. If not, the 0 flag will be turned off, while the FULL flag will be turned off. If the TB is a constant, the system will regard it as a table address.



Typical use

PSH FW000, DW000



Effective parameter

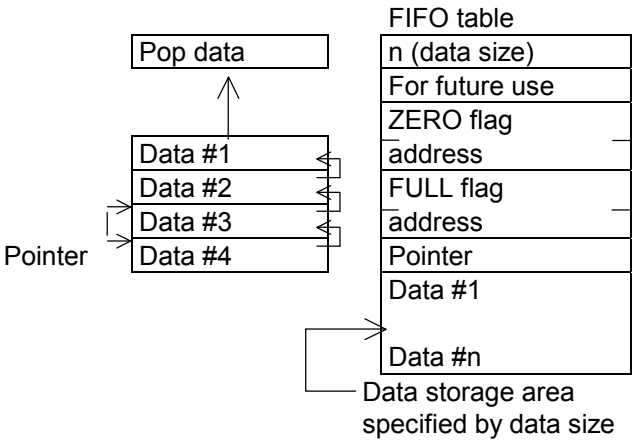
△ is an address. Parameter error if the number is odd.

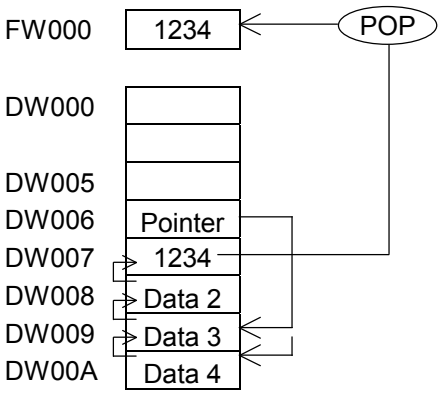
S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

TB	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	–	√	△
Direct long length	–	√	△
Indirect word length	–	△	△
Indirect long length	–	△	△

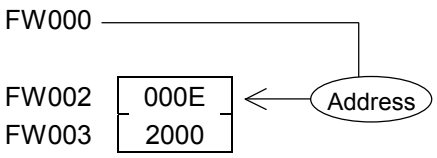
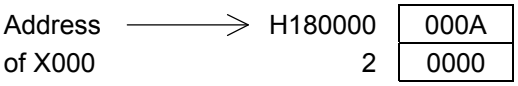
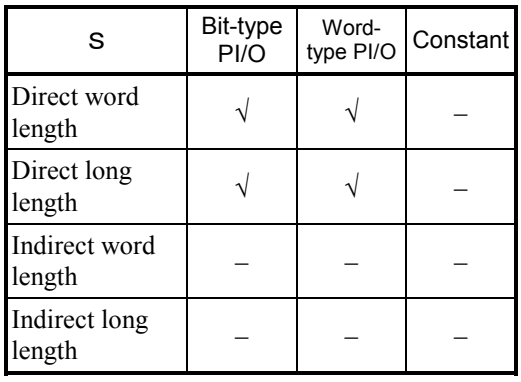
<table>
<tr><td>POP</td><td>FIFO read</td></tr>
<tr><td>Function description</td><td>This function pops the FIFO table and stores pop data in the destination. The data length of the FIFO table is word only.</td></tr>
<tr><td>Parameter and operation</td><td>POP TB, D

D: Destination
TB: Starting address of the FIFO table

 
</td></tr>
<tr><td>Flag configuration</td><td>The E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>The system will perform no operation when n ≤ 0 and n > 256. The system will perform no operation when the pointer < 0 or the data size < pointer. When the pointer = 0, the system will turn on the 0 flag and perform no operation. If the system decrements the pointer and it becomes 0, the 0 flag will be turned on. If not, the 0 flag will be turned off, while the FULL flag will be turned off. If the TB is a constant, the system will regard it as a table address.</td></tr>
<tr><td>Typical use</td><td>POP DW000, FW000

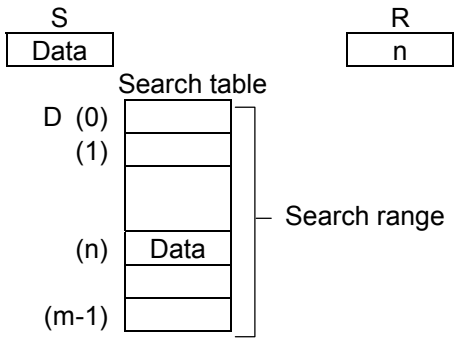
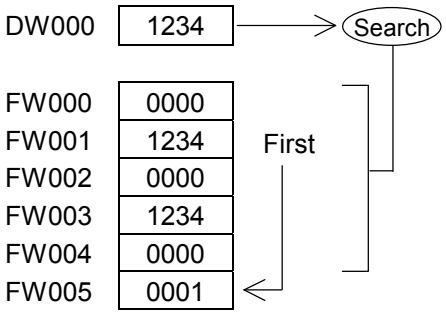
 
</td></tr>
<tr><td>Effective parameter

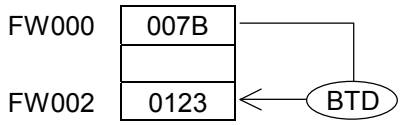
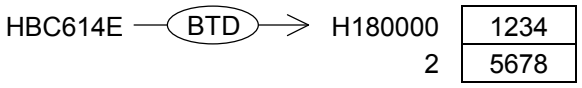
△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>TB</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>–</td><td>√</td><td>△</td></tr>
<tr><td>Direct long length</td><td>–</td><td>√</td><td>△</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>D</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

AST	Address set
Function description	This function transfers the address data of the source to the destination. The PI/O alone is effective.
Parameter and operation	AST S, D Address of S → D S : Source D : Destination
Flag configuration	The E will change. The others will become turned off.
Remark	Note that, if D is specified as word length, the address value will be type-converted to word length.
Typical use	AST FW000, [FW002]  AST X000, @ [H180000] 
Effective parameter △ is an address. Parameter error if the number is odd.	

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	–	–
Indirect long length	–	–	–

D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	–	–	–
Direct long length	√	√	–
Indirect word length	–	–	–
Indirect long length	–	△	△

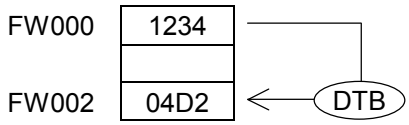
SCH	Search
Function description	This function searches the range of a specified distance (in meters) from the destination for the contents of the source and stores in the result the number (n) of steps from the destination.
Parameter and operation	SCH S, D, m, R S : Source D : Destination m : Number of search steps R : Result 
Flag configuration	The E will change. The others will become turned off.
Remark	The system will perform no operation when m ≤ 0 and m > 256. The matching data is the first thing found. If the search range contains no matching data, the result will be set to -1. If the search data type (long or word) differs, an error will occur. The n begins with 0.
Typical use	SCH DW000, FW000, 5, FW005 
Effective parameter △ is an address. Parameter error if the number is odd.	<table><tr><th>S, m</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr><tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr><tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr><tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr><tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr></table> <table><tr><th>D, R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr><tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr><tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr><tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr><tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr></table>

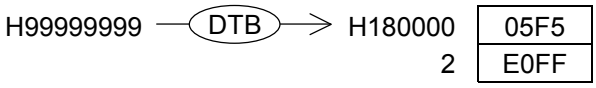
BTD	Binary → BCD conversion
Function description	This function converts the contents of the source from binary to BCD and stores them in the result.
Parameter and operation	BTD S, R S (binary) → R (BCD) S : Source R : Result
Flag configuration	The E and V will change. The others will become turned off.
Remark	When S < 0, the E flag will be turned on and the V flag will be turned off. The system will perform no operation. When overflowed, the system sets it to H9999 or H99999999.
Typical use	BTD FW000, FW002  BTD HBC614E, @ [H180000] 
Effective parameter △ is an address. Parameter error if the number is odd.	See tables below

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

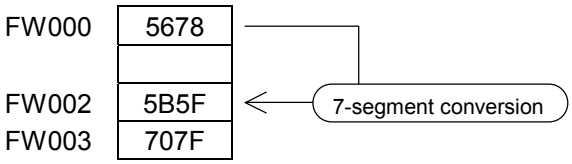
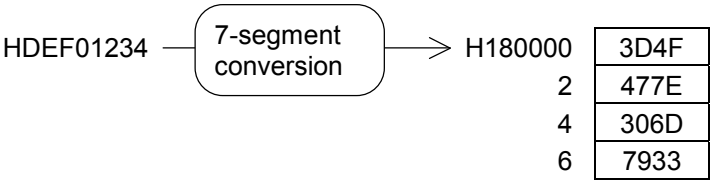
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

<table>
<tr><td>DTB</td><td>BCD → binary conversion</td></tr>
<tr><td>Function description</td><td>This function converts the contents of the source from BCD to binary and stores them in the result.</td></tr>
<tr><td>Parameter and operation</td><td>DTB S, R
S (BCD) → R (binary)

S : Source
R : Result</td></tr>
<tr><td>Flag configuration</td><td>The E and V will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When anything between A and F is used in S, the E flag will be turned on and the system will perform no operation.</td></tr>
<tr><td>Typical use</td><td>DTB FW000, FW002
 

DTB H99999999, @ [H180000]
 
</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

SEG	Binary → 7-segment conversion
Function description	This function converts the contents of the source from binary to 7-segment data and stores them in the result.
Parameter and operation	SEG S, R S (binary) → R (7-segment data) S : Source R : Result
Flag configuration	The E will change. The others will become turned off.
Remark	The size × 2 of the S is written in the R.
Typical use	SEG FW000, FW002  SEG HDEF01234, @ [H180000] 

[7-segment table]

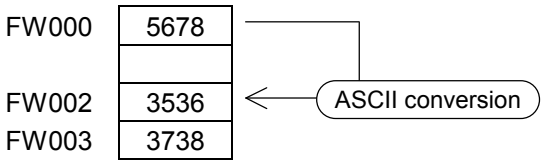
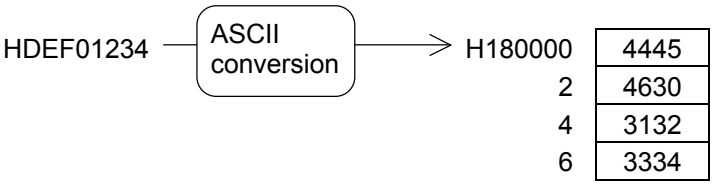
No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Data	7E	30	6D	79	33	5B	5F	70	7F	7B	77	1F	4E	3D	4F	47

Effective parameter

△ is an address. Parameter error if the number is odd.

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

ASP	Binary → ASCII conversion pack mode
Function description	This function converts the contents of the source from binary to ASCII data and stores them in the result in the pack mode.
Parameter and operation	ASP S, R S : Source R : Result S (binary) → R (ASCII pack)
Flag configuration	The E will change. The others will become turned off.
Remark	The size × 2 of the S is written in the R.
Typical use	ASP FW000, FW002  ASP HDEF01234, @ [H180000]  [ASCII, binary table]

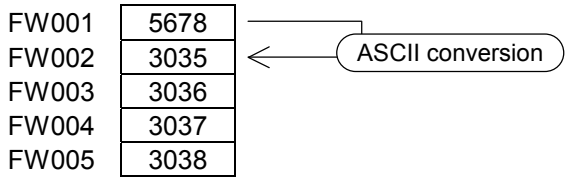
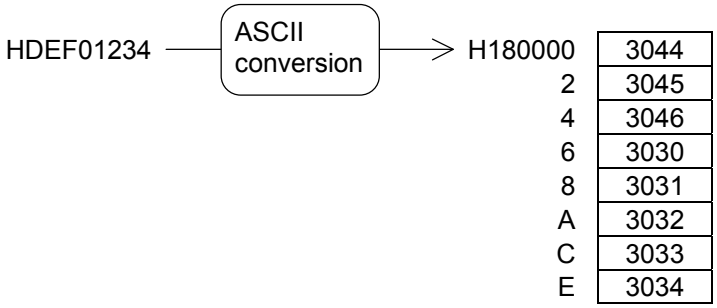
Binary	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	30	31	32	33	34	35	36	37	38	39	41	42	43	44	45	46

Effective parameter

△ is an address. Parameter error if the number is odd.

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

ASU	Binary → ASCII conversion unpack mode
Function description	This function converts the contents of the source from binary to ASCII data and stores them in the result in the unpack mode.
Parameter and operation	ASU S, R S : Source R : Result S (binary) → R (ASCII unpack)
Flag configuration	The E will change. The others will become turned off.
Remark	The size × 4 of the S will be written in the R.
Typical use	ASU FW001, FW002  ASU HDEF01234, @ [H180000]  [ASCII, binary table]

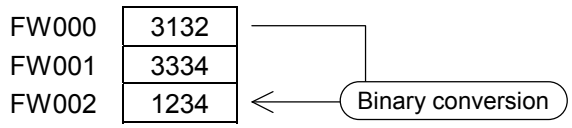
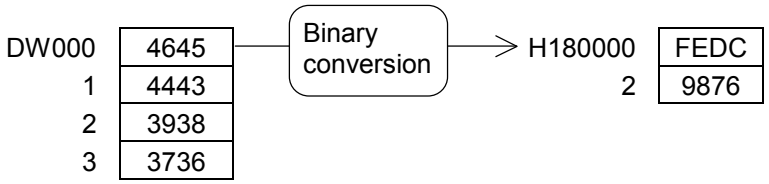
Binary	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	30	31	32	33	34	35	36	37	38	39	41	42	43	44	45	46

Effective parameter

△ is an address. Parameter error if the number is odd.

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

APB	ASCII → binary conversion pack mode
Function description	This function converts the contents of the source from ASCII data (pack mode) to binary and stores them in the result.
Parameter and operation	APB S, R S : Source R : Result S (ASCII pack) → R (binary)
Flag configuration	The E will change. The others will become turned off.
Remark	The size × 2 of R will be taken from S and converted. If S contains any data from H30 to 39 or H41 to 46, the E flag will be turned on and the system will perform no operation.
Typical use	APB FW000, FW002  APB DW000, @ [H180000] 

[ASCII, binary table]

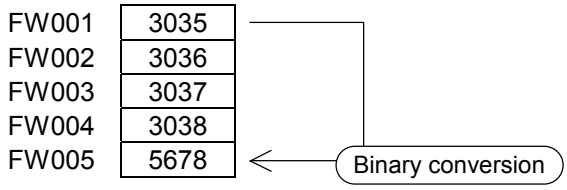
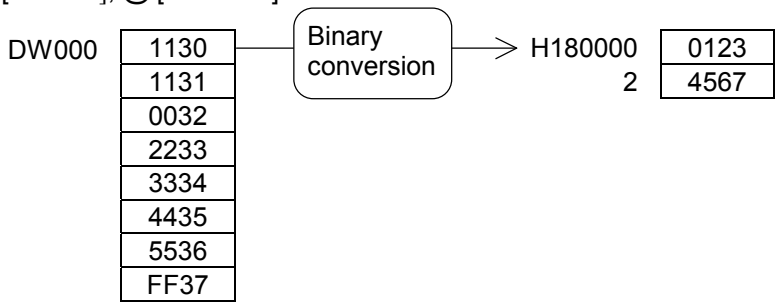
Binary	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	30	31	32	33	34	35	36	37	38	39	41	42	43	44	45	46

Effective parameter

△ is an address. Parameter error if the number is odd.

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	–	√	–
Direct long length	–	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

AUB	ASCII → binary conversion unpack mode
Function description	This function converts the contents of the source from ASCII data (unpack mode) to binary and stores them in the result.
Parameter and operation	AUB S, R S : Source R : Result S (ASCII unpack) → R (binary)
Flag configuration	The E will change. The others will become turned off.
Remark	The size × 4 of R will be taken from S and converted. If S contains any data from H30 to 39 or H41 to 46, the E flag will be turned on and the system will perform no operation.
Typical use	AUB FW001, FW002  AUB [DW000], @ [H180000]  [ASCII, binary table]

Binary	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	30	31	32	33	34	35	36	37	38	39	41	42	43	44	45	46

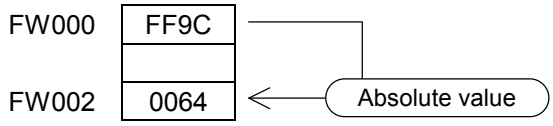
Effective parameter

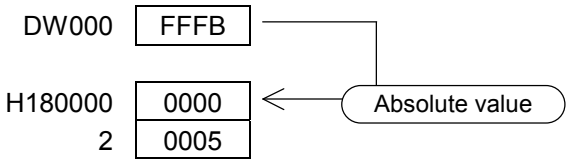
△ is an address. Parameter error if the number is odd.

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	–	√	–
Direct long length	–	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

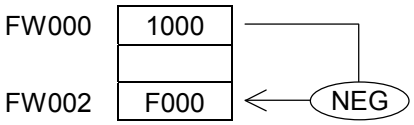
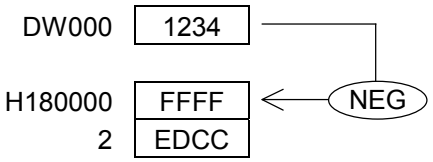
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

<table>
<tr><td>ABS</td><td>Absolute value</td></tr>
<tr><td>Function description</td><td>This function stores the absolute values in the source in the result.</td></tr>
<tr><td>Parameter and operation</td><td>ABS S, R
|S| → R

S : Source
R : Result</td></tr>
<tr><td>Flag configuration</td><td>The E and V will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When overflowed, the system will set the result to H7FFFFFFF.</td></tr>
<tr><td>Typical use</td><td>ABS FW000, FW002
 

ABS DW000, @ [H180000]
 
</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

NEG	Code conversion
Function description	This function converts the code in the source and stores it in the result.
Parameter and operation	NEG S, R -S → R S : Source R : Result
Flag configuration	The E and V will change. The others will become turned off.
Remark	When overflowed, the system will set the result to H7FFF and H7FFFFFFF.
Typical use	NEG FW000, FW002  NEG DW000, @ [H180000] 

Effective parameter

△ is an address. Parameter error if the number is odd.

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

DCD	Decode
Function description	This function decodes the contents of the source and stores the finding in the result.
Parameter and operation	DCD S, R S : Source R : Result

S | | n | →

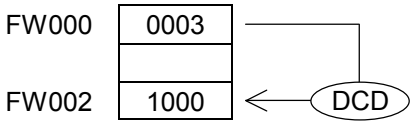
	0		n		(LSB)
R	0	~ 0	1	0	~ 0

With n specified in S, the system will turn on the bit of the bit number n as counted from the MSB of R (counted from 0 onwards).

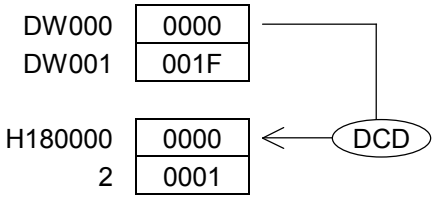
Flag configuration	The E will change. The others will become turned off.
Remark	The effective bit of S will depend on the specified size of R when word is specified as four low-level bits and when long is specified as five low-level bits.

Typical use

DCD FW000, FW002



DCD [DW000], @ [H180000]



Effective parameter

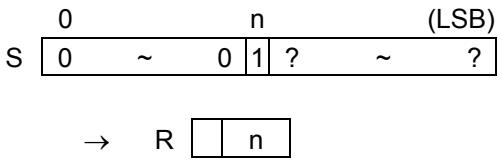
△ is an address. Parameter error if the number is odd.

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

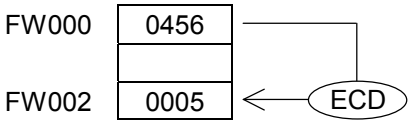
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

<table>
<tr><td>ECD</td><td>Encode</td></tr>
<tr><td>Function description</td><td>This function encodes the contents of the source and stores the finding in the result.</td></tr>
<tr><td>Parameter and operation</td><td>ECD S, R

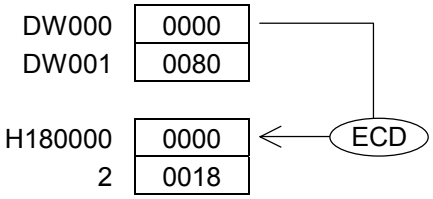
S : Source
R : Result

The system will count the items, starting from the MSB of S (counting it from 0 onwards) and stores in R the n where the first 1 is detected.</td></tr>
<tr><td>Flag configuration</td><td>The E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>The system will perform no operation when S = 0. The bit to be encoded will only be effective for the bit where the first 1 is detected by the MSB.</td></tr>
<tr><td>Typical use</td><td>ECD FW000, FW002

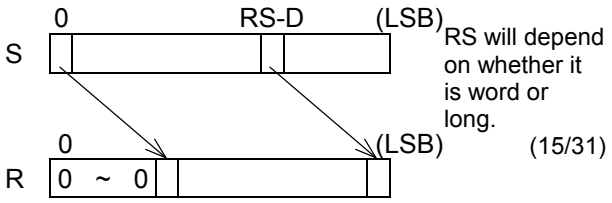
 

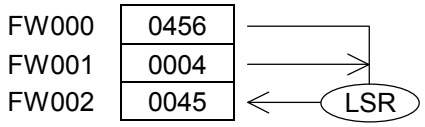
ECD [DW000], @ [H180000]

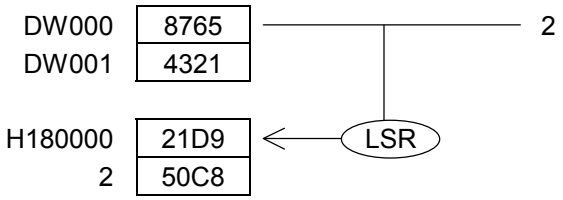
 
</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

LSR	Logic right-shift
Function description	This function right-shifts the contents of the source with the contents of the destination and stores the finding in the result.
Parameter and operation	LSR S, D, R S : Source R : Result D : Destination
Flag configuration	The E will change. The others will become turned off.
Remark	The effective bit of D will be four low-level bits when S is word length and five low-level bits when it is long length.
Typical use	LSR FW000, FW001, FW002 LSR [DW000], 2, @ [H180000]





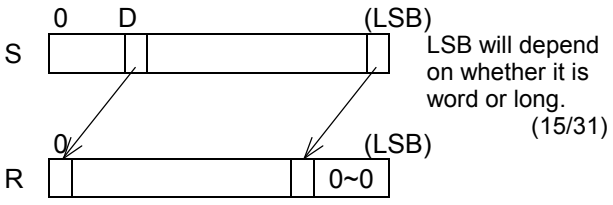
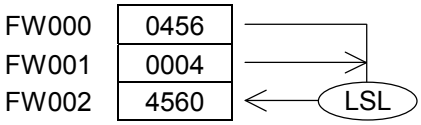
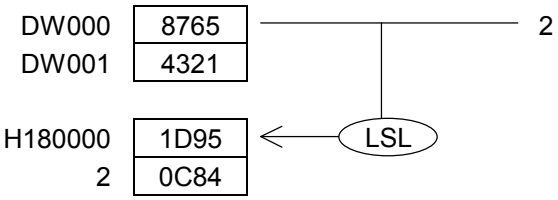
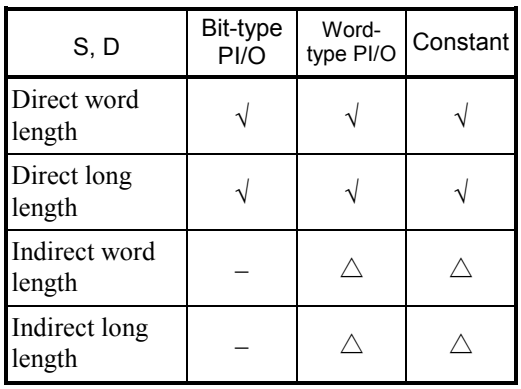
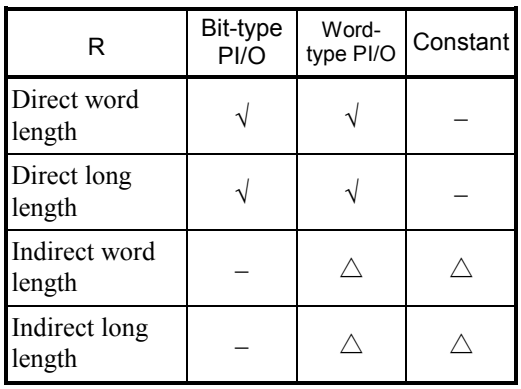


Effective parameter

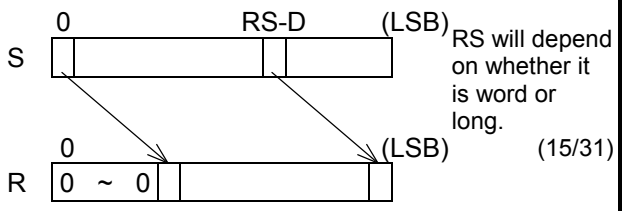
△ is an address. Parameter error if the number is odd.

S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

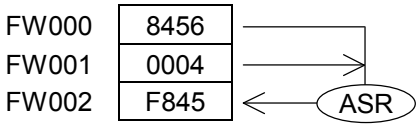
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

LSL	Logic left-shift
Function description	This function left-shifts the contents of the source with the contents of the destination and stores the finding in the result.
Parameter and operation	LSL S, D, R S : Source R : Result D : Destination 
Flag configuration	The E will change. The others will become turned off.
Remark	The effective bit of D will be four low-level bits when S is word length and five low-level bits when it is long length.
Typical use	LSL FW000, FW001, FW002  LSL [DW000], 2, @ [H180000] 
Effective parameter △ is an address. Parameter error if the number is odd.	 

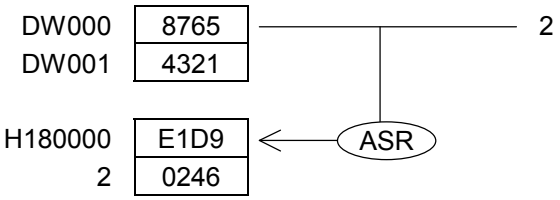
ASR	Arithmetic right-shift
Function description	This function right-shifts (holds the code bit) the contents of the source with the contents of the destination and stores the finding in the result.
Parameter and operation	ASR S, D, R S : Source R : Result D : Destination



Flag configuration	The E and V will change. The others will become turned off.
Remark	When R is word length, the system sets it to a low-level word. The effective bit of D will be four low-level bits when S is word length, and five low-level bits when S is long length.
Typical use	ASR FW000, FW001, FW002



ASR [DW000], 2, @ [H180000]

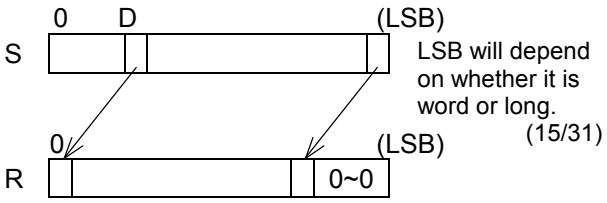
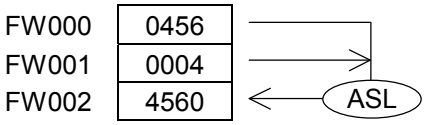
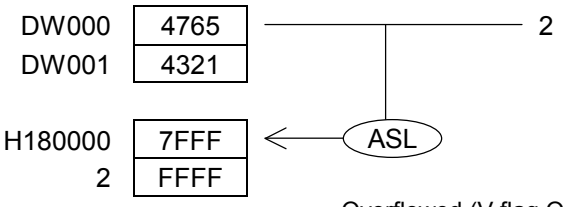


Effective parameter

△ is an address. Parameter error if the number is odd.

S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

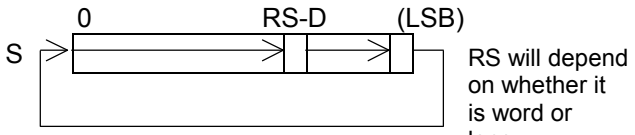
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

ASL	Arithmetic left-shift
Function description	This function left-shifts the contents of the source with the contents of the destination and stores the finding in the result. When overflowed, the system will set it to full scale.
Parameter and operation	ASL S, D, R S : Source R : Result D : Destination 
Flag configuration	The E and V will change. The others will become turned off.
Remark	The effective bit of D will be four low-level bits when S is word length, and five low-level bits when S is long length.
Typical use	ASL FW000, FW001, FW002  ASL [DW000], 2, @ [H180000] 
Effective parameter △ is an address. Parameter error if the number is odd.	

S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

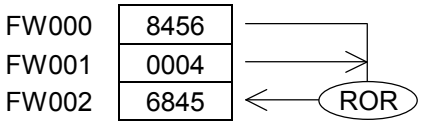
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

ROR	CW rotation
Function description	This function rotates the contents of the source clockwise with the contents of the destination and stores the finding in the result.
Parameter and operation	ROR S, D, R S : Source R : Result D : Destination

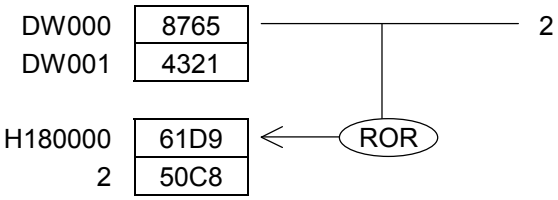


(15/31)

Flag configuration	The E will change. The others will become turned off.
Remark	The effective bit of D will be four low-level bits when S is word length, and five low-level bits when S is long length.
Typical use	ROR FW000, FW001, FW002



ROR [DW000], 2, @ [H180000]

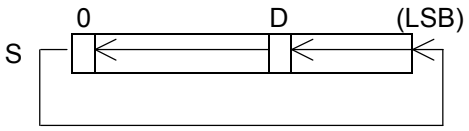
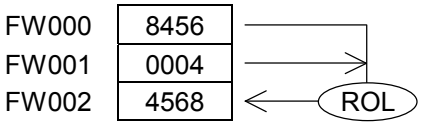
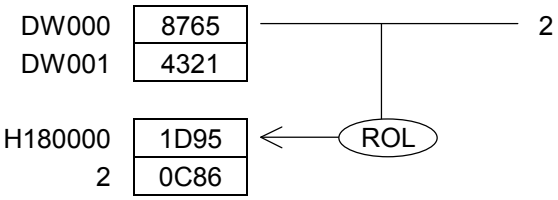


Effective parameter

△ is an address. Parameter error if the number is odd.

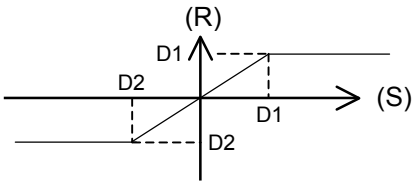
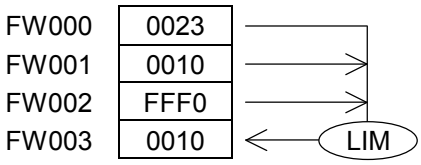
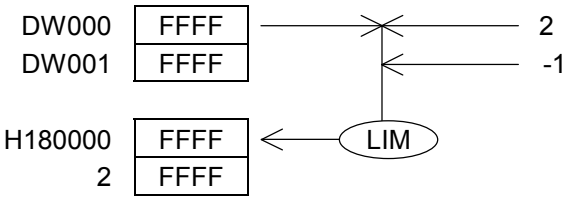
S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

ROL	CCW rotation
Function description	This function rotates the contents of the source counterclockwise with the contents of the destination and stores the finding in the result.
Parameter and operation	ROL S, D, R S : Source R : Result D : Destination
	 RS will depend on whether it is word or long. (15/31)
Flag configuration	The E will change. The others will become turned off.
Remark	The effective bit of D will be four low-level bits when S is word length, and five low-level bits when S is long length.
Typical use	ROL FW000, FW001, FW002  ROL [DW000], 2, @ [H180000] 
Effective parameter △ is an address. Parameter error if the number is odd.	

S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

LIM	Limiter
Function description	This function compares the contents of the source with the contents of the boundary values (destinations D1 and D2) and stores the finding in the result.
Parameter and operation	LIM S, D1, D2, R S : Source R : Result D1, D2: Destination 
Flag configuration	The E and V will change. The others will become turned off.
Remark	When D1 < D2, the E flag will be turned on.
Typical use	LIM FW000, FW001, FW002, FW003  LIM [DW000], 2, -1, @ [H180000] 

Effective parameter

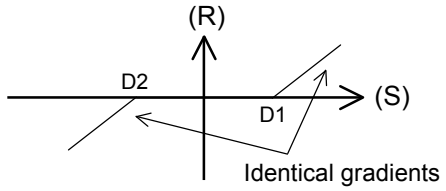
△ is an address. Parameter error if the number is odd.

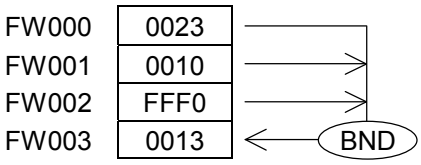
S, D1, D2	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

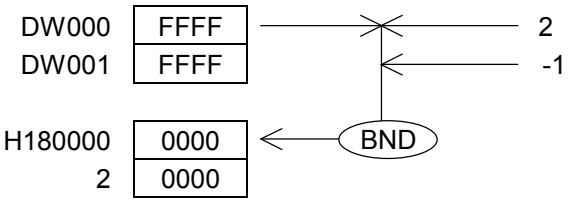
<table>
<tr><td>BND</td><td>Dead band</td></tr>
<tr><td>Function description</td><td>This function compares the contents of the source with the contents of the boundary values (destinations D1 and D2) and stores them in the result, regarding the boundary range as a dead band (data 0).</td></tr>
<tr><td>Parameter and operation</td><td>BND S, D1, D2, R

S : Source
R : Result
D1, D2: Destination

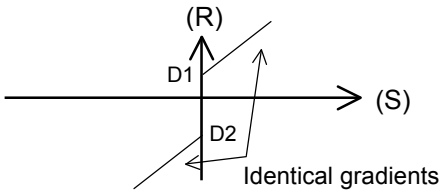
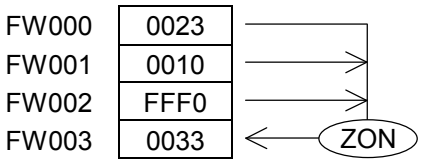
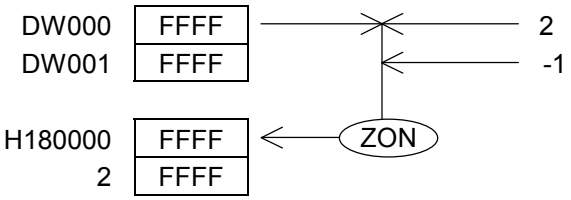
 
</td></tr>
<tr><td>Flag configuration</td><td>The E and V will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When D1 < D2, the E flag will be turned on.</td></tr>
<tr><td>Typical use</td><td>BND FW000, FW001, FW002, FW003

BND [DW000], 2, -1, @ [H180000]

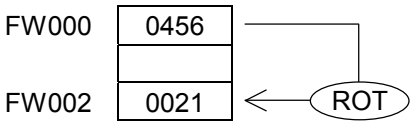
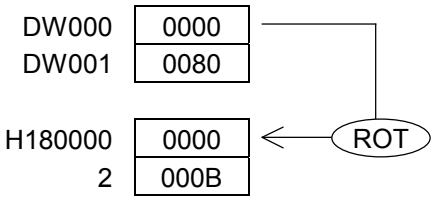
 
</td></tr>
<tr><td>Effective parameter

△ is an address.
Parameter error if the number is odd.</td><td>
<table>
<tr><th>S, D1, D2</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

ZON	Dead zone
Function description	This function adds a bias (destinations D1 and D2) to the contents of the source depending on whether it is positive or negative and stores the finding in the result.
Parameter and operation	ZON S, D1, D2, R S : Source R : Result D1, D2: Destination 
Flag configuration	The E and V will change. The others will become turned off.
Remark	When D1 < D2, the E flag will be turned on.
Typical use	ZON FW000, FW001, FW002, FW003  ZON [DW000], 2, -1, @ [H180000] 
Effective parameter △ is an address. Parameter error if the number is odd.	

S, D1, D2	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

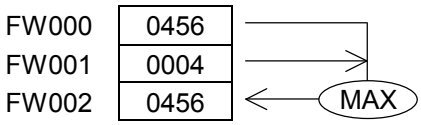
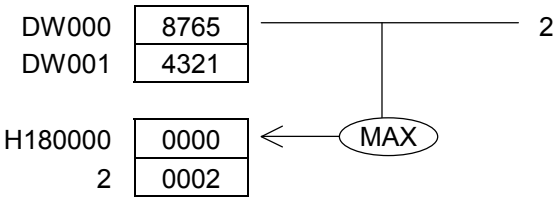
ROT	Square root
Function description	This function stores the square root (the integer portion only) of the contents of the source in the result.
Parameter and operation	ROT S,R S : Source R : Result When S ≥ 0, Square root of S → R When S < 0 0 → R
Flag configuration	The E and V will change. The others will become turned off.
Remark	
Typical use	ROT FW000, FW002  ROT [DW000], @ [H180000] 

Effective parameter

△ is an address. Parameter error if the number is odd.

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

MAX	Maximum value	
Function description	This function compares the contents of the source with those of the destination in size and stores the larger value in the result.	
Parameter and operation	MAX S, D, R S : Source R : Result D : Destination	When S ≥ D, S → R When S < D D → R
Flag configuration	The E and V will change. The others will become turned off.	
Remark		
Typical use	MAX FW000, FW001, FW002  MAX [DW000], 2, @ [H180000]  	

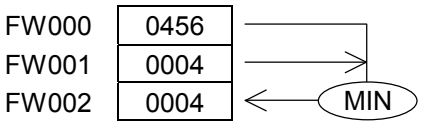
Effective parameter

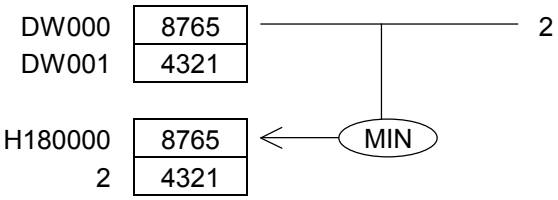
△ is an address. Parameter error if the number is odd.

S, D	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	√
Direct long length	√	√	√
Indirect word length	–	△	△
Indirect long length	–	△	△

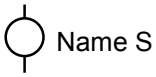
R	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	√	√	–
Direct long length	√	√	–
Indirect word length	–	△	△
Indirect long length	–	△	△

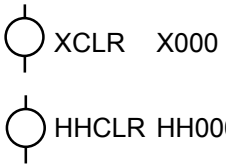
<table>
<tr><td>MIN</td><td colspan="2">Minimum value</td></tr>
<tr><td>Function description</td><td colspan="2">This function compares the contents of the source with those of the destination in size and stores the smaller value in the result.</td></tr>
<tr><td>Parameter and operation</td><td>MIN S, D, R

S : Source
R : Result
D : Destination</td><td>When S ≤ D, S → R
When S > D D → R</td></tr>
<tr><td>Flag configuration</td><td colspan="2">The E and V will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td colspan="2"></td></tr>
<tr><td>Typical use</td><td colspan="2">MIN FW000, FW001, FW002
 

MIN [DW000], 2, @ [H180000]
 
</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td colspan="2">
<table>
<tr><th>S, D</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
<table>
<tr><th>R</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Direct long length</td><td>√</td><td>√</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>△</td><td>△</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>CLR</td><td>Clear</td></tr>
<tr><td>Function description</td><td>This function clears the specified I/O area. The TCLR, UCLR, and CCLR will clear the discrete value area as well.</td></tr>
<tr><td>Parameter and operation</td><td> 

Name: Name of each CLR instruction
S: Source (the top of the I/O area to be cleared)</td></tr>
<tr><td>Flag configuration</td><td>The system will set all flags to 0.</td></tr>
<tr><td>Remark</td><td></td></tr>
<tr><td>Description</td><td>
<table>
<tr><th>Name</th><th>Function</th></tr>
<tr><td>XCLR</td><td>Clears X000 through XFFF.</td></tr>
<tr><td>YCLR</td><td>Clears Y000 through YFFF.</td></tr>
<tr><td>GCLR</td><td>Clears G000 through GFFF.</td></tr>
<tr><td>RCLR</td><td>Clears R000 through RFFF.</td></tr>
<tr><td>KCLR</td><td>Clears K000 through KFFF.</td></tr>
<tr><td>TCLR</td><td>Clears T000 through T3FF.
Clears the measurement of T.</td></tr>
<tr><td>UCLR</td><td>Clears U000 through U3FF.
Clears the measurement of U.</td></tr>
<tr><td>CCLR</td><td>Clears C000 through C3FF.
Clears the measurement of C.</td></tr>
<tr><td>VCLR</td><td>Clears V000 through VFFF.</td></tr>
<tr><td>ECLR</td><td>Clears E000 through EFFF.</td></tr>
<tr><td>FCLR</td><td>Clears S020 through S027.</td></tr>
<tr><td>JCLR</td><td>Clears J000 through JFFF.</td></tr>
<tr><td>QCLR</td><td>Clears Q000 through QFFF.</td></tr>
<tr><td>HHCLR</td><td>Clears HH000 through HH1FF.</td></tr>
</table>
</td></tr>
<tr><td>Typical use</td><td> 
</td></tr>
</table>

5.7 Applied Instructions for Ethernet Communication

5.7.1 Function overview

When engaging in TCP or UDP communication with the HI-FLOW program, use applied instructions for Ethernet communication.
For applied instructions for Ethernet communication, the HI-FLOW system supports the following interfaces:

Instruction	Function
TOP	Open a TCP connection (client)
TPOP	Open a TCP connection (server)
TCLO	Close a TCP connection
TRCV	Receive a TCP
TSND	Send a TCP
UOP	Open a UDP
UCLO	Close a UDP
URCV	Receive a UDP
USND	Send a UDP

The communication specifications that comply with the system expansion operation function are shown below.

Item	Specification	Remarks
Number of sockets usable simultaneously	CME: 16	The sum of TCP and UDP communication
	ET.NET (main): 16	
	ET.NET (sub): 16	
	OPTET (Module 0 to 3) common: 32	
Size of data communicated	TCP communication: 0 to 4,096 bytes	
	UDP communication: 0 to 1,472 bytes	
Port No.	1 to 65535	It is recommended to use 10000 to 59999. The system reserves settings for 60000 and later.

To use the Ethernet communication operation function, a module of the following version or later must be used according to the LPU unit configuration.

Configuration	Prerequisite module
CMU only	CMU (LQP520): Ver-Rev 03-01 or later
CMU+ET.NET	LPU (LQP510): Ver-Rev 02-02 or later
	CMU (LQP520): Ver-Rev 04-00 or later
	ET.NET (LQE720): Ver-Rev 01-00 or later
CMU+OPTET	LPU (LQP510): Ver-Rev 02-02 or later
	CMU (LQP520): Ver-Rev 06-00 or later
	OPTET (LQE710): Ver-Rev 01-00 or later

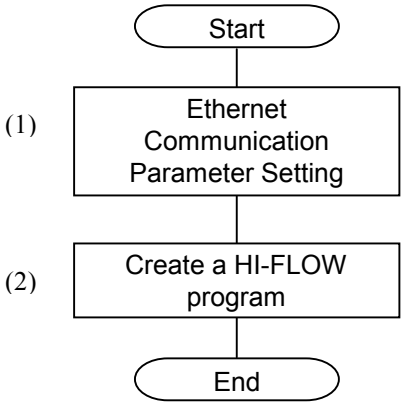
Note that Ver-Rev above applies to the microprogram of each module indicated in “Module List” of the S10V basic system.

Execute an applied instruction for Ethernet communication, and the system will set the execution results to system registers S9C0 to S9FF, and S690 to S6AF for each management number.
The system will set a 0 to the system register corresponding to the particular management number when normally terminated. It will then set a 1 when abnormally terminated.
The management number is a number corresponding to the applicable socket.

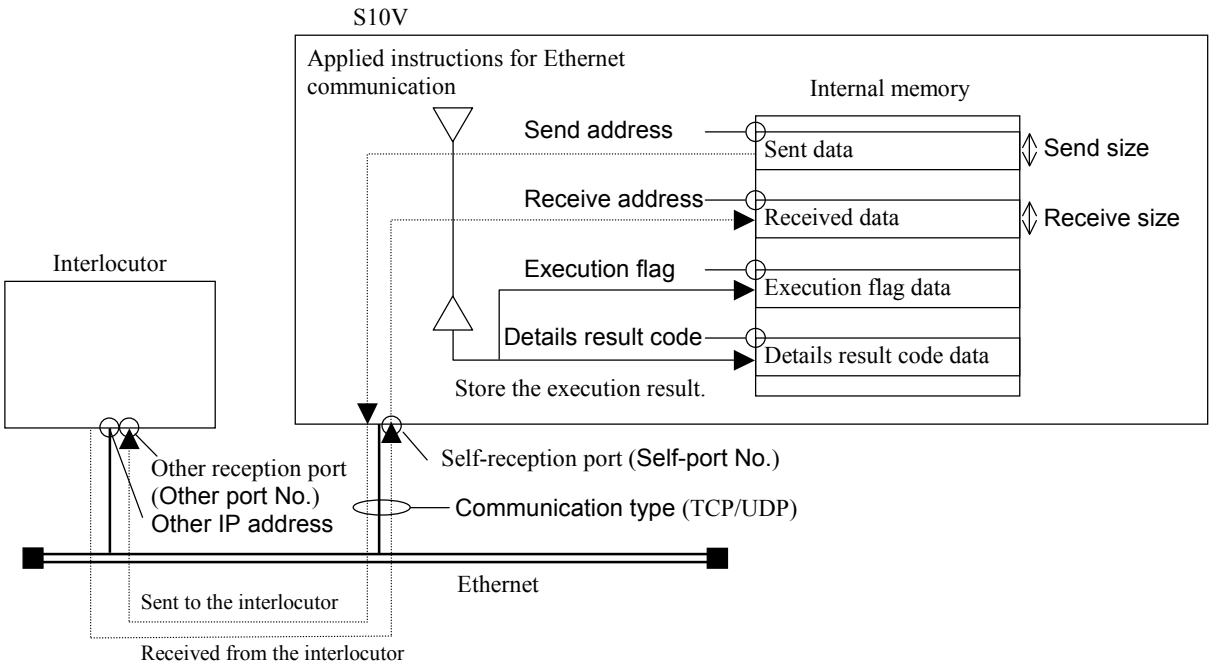
Register		Management number	Remark
Word	Bit		
SW9C0	S9C0	1	CMU for Ethernet communication
	S9C1	2	
	⟆	⟆	
	S9CE	15	
	S9CF	16	
SW9D0	S9D0	17	ET.NET (main) for Ethernet communication
	S9D1	18	
	⟆	⟆	
	S9DE	31	
	S9DF	32	
SW9E0	S9E0	33	ET.NET (sub) for Ethernet communication
	S9E1	34	
	⟆	⟆	
	S9EE	47	
	S9EF	48	
SW9F0	S9F0	For future use	
	⟆		
	S9FF		
SW690	S690	49	OPTET for Ethernet communication
	S691	50	
	⟆	⟆	
	S69E	63	
	S69F	64	
SW6A0	S6A0	65	OPTET for Ethernet communication
	S6A1	66	
	⟆	⟆	
	S6AE	79	
	S6AF	80	

5.7.2 How to use applied instructions

Applied instructions for Ethernet communication function according to the parameters configured on [Set Ethernet Communication] window of the HI-FLOW system. Therefore parameter information on [Set Ethernet Communication] window must be configured before using each instruction.

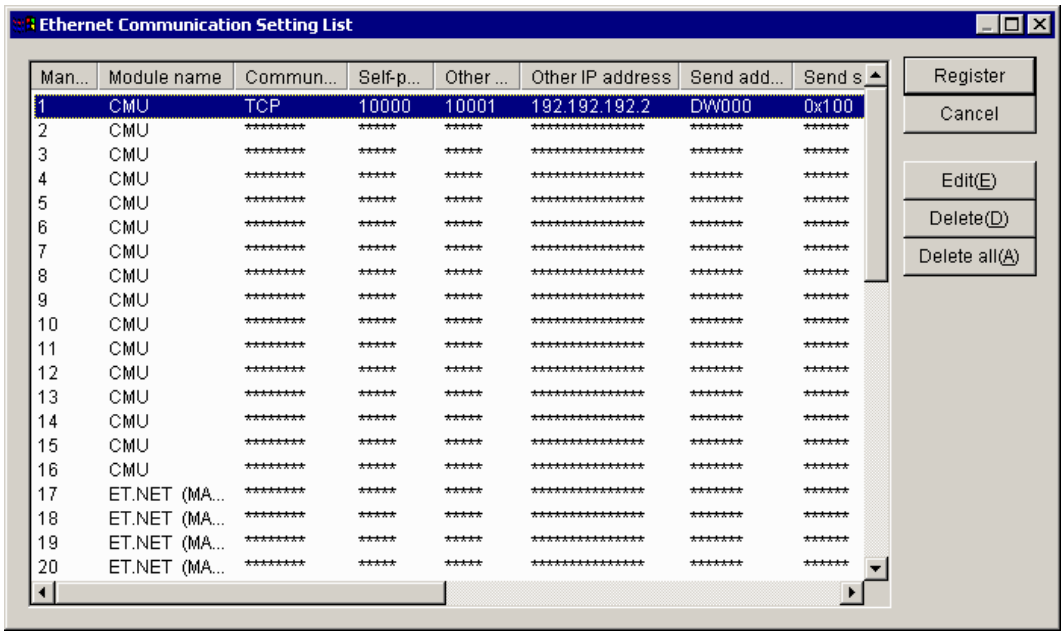


For parameter information, configure the Ethernet settings according to the chart below.
The items in bold letters in the chart are information to be configured on [Set Ethernet Communication] window. On details of the settings on [Set Ethernet Communication] window, see "(1) Configuring ethernet communication parameter."



(1) Configuring ethernet communication parameter

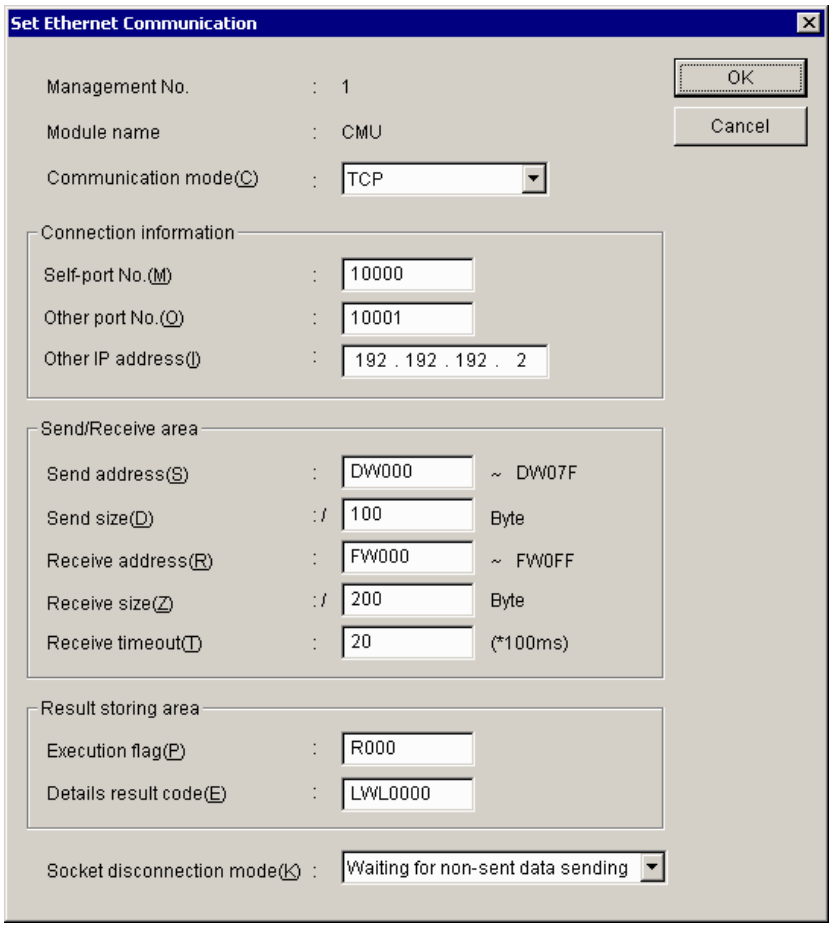
To configure parameter information on [Set Ethernet Communication] window, select HI-FLOW Process Sheet/HI-FLOW Sheet and then [Utility] - [Set Ethernet Communication]. [Set Ethernet Communication] window can only be configured when online. The following [Ethernet Communication Setting List] window will appear.



[Ethernet Communication Setting List] window

Specify the line of parameter information on [Ethernet Communication Setting List] window, and then click the Edit button or double-click the line of parameter information on [Ethernet Communication Setting List] window to display [Set Ethernet Communication] window for the specified line.

For details of the settings, refer to “SOFTWARE MANUAL OPERATION HI-FLOW For Windows® (Manual number SVE-3-132).”



[Set Ethernet Communication] window

Here is a description of the parameter information on [Set Ethernet Communication] window.

Management No.: Displays management numbers specified on [Ethernet Communication Setting List] window.

Module name: Displays the module for communication specified on [Ethernet Communication Setting List] window.
The module name is fixed according to the management number and the module shown below will be displayed.
When you want to use an OPTET module whose management number is in the range 49 to 80, specify its module number.

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main)
33 to 48	ET.NET (sub)
49 to 80	OPTET (Module 0 to 3)

Communication mode: While in the combo box, select “TCP” or “UDP.” It is “TCP” by default.

Self-port No.: Specify a port number for communication as a decimal number. (The specification range is between 1 and 65535.) It is blank by default. (Using a number between 10000 and 59999 is recommended. The system reserves numbers for 60000 and above.)

Other port No.: Specify the port number of the interlocutor as a decimal number. (The specification range is between 1 and 65535.) It is blank by default. (Using a number between 10000 and 59999 is recommended.) The system reserves a number for 60000 and above.

Other IP address: Specify the IP address of the interlocutor. It is blank by default. To broadcast data by UDP transmission, specify the node address as 255, as in 255.255.255.255.

Send address: Specify the top address of sent data in word form (registers for long word and float only are in long word and float forms) of PI/O. The system does not allow you to specify a bit-type register, specify an area unassigned as a PI/O, or span two or more registers. It is blank by default. The send address and send size are used to calculate the final address of sent data and display it.

Send size: Specify a send size for data in a hexadecimal number. It is blank by default. The unit is the byte. For each communication type, the system allows you to specify either of the following sizes:
TCP: 0x0 to 0x1000 (0 to 4,096)
UDP: 0x0 to 0x5C0 (0 to 1,472)

Receive address: Specify the top address of the area for storing received data in word form (registers for long word and float only are in long word and float forms) of PI/O. The system does not allow you to specify a bit-type register, specify an area unassigned as PI/O, or span two or more registers. It is blank by default. The receive address and the receive size are used to calculate the final address of received data and display it.

Receive size: Specify a receive size for data in a hexadecimal number. It is blank by default and the units are bytes. For each communication type, the system allows you to specify either of the following sizes:
TCP: 0x0 to 0x1000 (0 to 4,096)
UDP: 0x0 to 0x5C0 (0 to 1,472)

Receive timeout: Set a wait time for received data to arrive in case data cannot be received when a reception instruction is issued. Specify a range between 0 and 100 (0 and 10 seconds) at increments of 100 ms. (0 means no timeout.) It is set to 10 (1 second) by default. Set a timeout setting. If a reception instruction causes a reception timeout, the reception instruction will cause an error with no reception data (EWOULDBLOCK).

Execution flag: Specify with a bit-type register that specifies whether an applied instruction for Ethernet communication is being processed. It is blank by default.

Details result code: Specify with a long-type register an area for storing a detailed result code for the execution result of an applied instruction for Ethernet communication. It is blank by default.

Socket disconnection mode: Can only be specified when the communication method is "TCP." Select "Waiting for non-sent data sending" or "Non-sent data destruction" from the combo box. It is "Waiting for non-sent data sending" by default. Here are the options and their meanings:

Waiting for non-sent data sending…If data has not yet been sent, the system will wait until the data flows. Any unread data will be discarded.

Non-sent data destruction…If data has not yet been sent, the system will disconnect the channel and relieve the socket without waiting for the data to flow. In that case, the TCP of the interlocutor host will receive an RST. Since the disconnection takes place differently from the way it usually occurs, be careful as to how the system functions (the method of reporting when an RST is received by the UP) when the interlocutor host receives an RST. Any unread received data will be discarded.

Here are registers specifiable on [Set Ethernet Communication] window.

List of configurable registers

(1/2)

No.	Item	Symbol	Send address	Received address	Execution flag	Details result code
1	External input	X	√	√	√	√
2	External output	Y	√	√	√	√
3	Internal register	R	√	√	√	√
4	Keep relay	K	√	√	√	√
5	On-delay timer	T	√	√	√	√
6	One-shot timer	U	√	√	√	√
7	Up-down counter	C	√	√	√	√
8	Global link register	G	√	√	√	√
9	Nesting coil	N	√	√	√	√
10	Process register	P	√	√	√	√
11	Event register	E	√	√	√	√
12	Edge contact	V	√	√	√	√
13	Z register	Z	√	√	√	√
14	System register	S	√	√	√	√
15	Data register	DW	√	√	–	√
16	Work register	FW	√	√	–	√
17	Internal register	M	√	√	√	√
18	Internal register (long/word)	BD	–	–	–	–
19	For fast remote I/O input	I	√	√	–	√
20	For fast remote I/O output	O	√	√	–	√
21	Between HI-FLOW and ladder	J	√	√	√	√
22	shared data register	Q	√	√	√	√

√: Specifiable
–: Unspecifiable

(2/2)

No.	Item	Symbol	Send address	Received address	Execution flag	Details result code
23	Work register	LB	√	√	√	√
24	Word-only work register	LW	√	√	–	√
25	Work register for long word only	LL	√	√	–	√
26	Work register for single-precision floating-point only	LF	√	√	–	√
27	Word-only work register (held after blackout)	LX	√	√	–	√
28	Work register for long word only (held after blackout)	LM	√	√	–	√
29	Work register for single-precision floating-point only (held after blackout)	LG	√	√	–	√
30	Work register for ladder converter only	LR	√	√	√	√
31	Work register for ladder converter only (for edge contact)	LV	√	√	√	√

√: Specifiable
–: Unspecifiable

The following are detailed result codes of the applied instructions for Ethernet communication.

List of details result codes

(1/2)

Value	Meaning	Corrective action
0	Normal (TOP, TPOP, TCLO, UOP, UCLO)	–
0 to 4,096	Normal (size of sent/received data) (TRCV, TSND, URCV, USND)	–
0x80000005 (EIO)	Major hazard on the adaptor (device)	Reference the corrective action mentioned in the error log information (*1).
0x8000000D (EACCES)	Specifies a broadcast address for the destination IP address.	The Ethernet communication setting is incorrect. Check the setting.
0x80000016 (EINVAL)	Specifies a disconnected socket or a value with a negative buffer length.	The Ethernet communication setting is incorrect. Check the setting.
0x800000DA (EMSGSIZE)	Length of sent data out of the range	The Ethernet communication setting is incorrect. Check the setting.
0x800000E2 (EADDRINUSE)	The port number is occupied by another socket.	Check the port number being used.
0x800000E3 (EADDRNOTAVAIL)	Invalid port number or IP address	The Ethernet communication setting is incorrect. Check the setting.
0x800000E4 (ENETDOWN)	Device uninitialized or stopped	Reference the corrective action mentioned in the error log information (*1).
0x800000E5 (ENETUNREACH)	No information about the route of the destination IP address	Check the route information setting of the CMU module/ET.NET module. (*2)
0x800000E7 (ECONNABORTED)	Connection disconnected	• Check the cable wiring. • Check the program of the connection destination host.
0x800000E8 (ECONNRESET)	Connection reset from the TCP of the destination host	Check the program of the connection destination host.
0x800000E9 (ENOBUFS)	Failure in securing memory	Reference the corrective action mentioned in the error log information (*1).
0x800000EB (ENOTCONN)	Sent to an unconnected socket	TOP/TROP execution failed. Check the program.
0x800000EC (ESHUTDOWN)	Socket relieved from other tasks	Confirm that the same control number is not being used in the ladder/HI-FLOW program.
0x800000EE (ETIMEDOUT)	Connection request timeout	• Check the cable wiring. • Check the program of the connection destination host.
0x800000EF (ECONNREFUSED)	Destination socket nonexistent (server task unbound)	Check the program of the connection destination host.
0x800000F6 (EWOULDBLOCK)	No data received, or transmission failure because of the TCP transmission window being full	Check the program.
0x800000F9 (ENSOCK)	Exceeds the number of openable sockets	Check the program to ensure that the number of sockets being used simultaneously is 16 or less per module.
0x800000FB (ECARDOFF)	A card OFF is inserted, resulting in the unavailability of the adaptor (device)	TOP/TROP/UOP is not yet executed or its execution failed. Check the program.
0x80000516 (EBADF)	Socket unopened	

(*1) For how to reference the error log information, refer to "USER'S MANUAL BASIC MODULES (Manual number SVE-1-100)."

(*2) Use the setting tool of each module to set the route information.

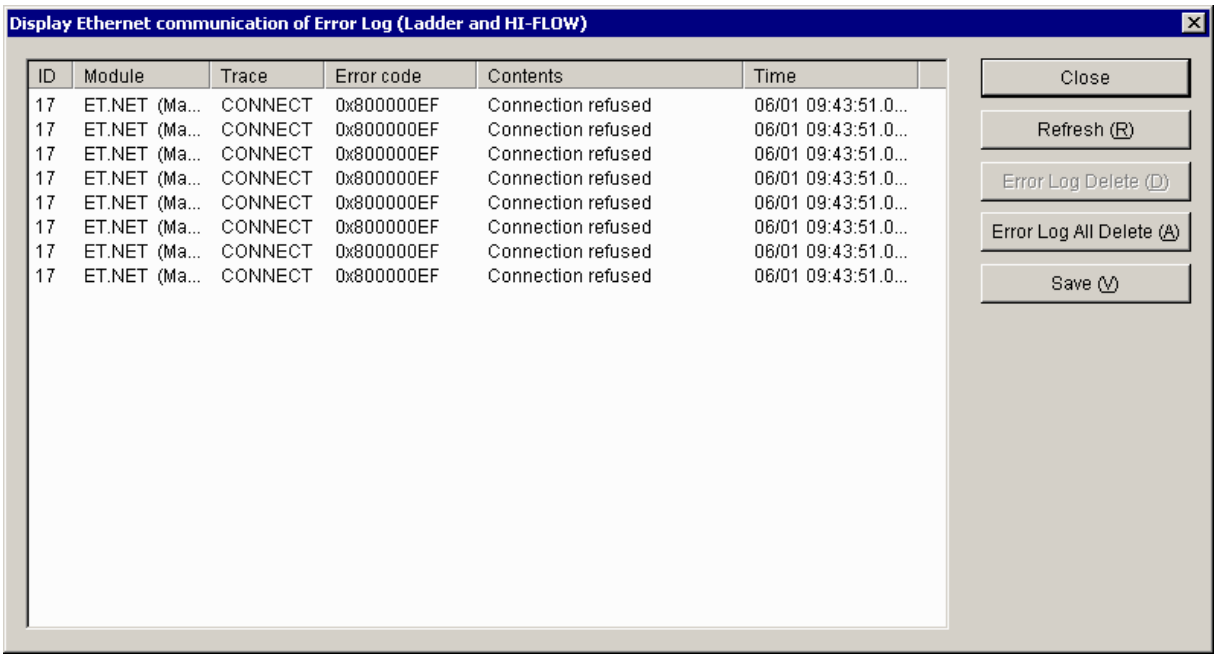
(2/2)

Value	Meaning	Corrective action
0xFFFFFFFB	The Ethernet module is down.	Reset the LPU and restart the ET.NET module. If the same error occurs after restart, the ET.NET module may be faulty. Replace the module.
0xFFFFFFFC	Ethernet module not mounted	Check the mounted status of the CMU module/ET.NET module.
0xFFFFFFFD	Failure in task startup	Check whether the version of the CMU module is applicable to the applied instructions used for Ethernet communication.
0xFFFFFFFE	Communication identifier error (control number being used)	Check whether there is any ladder/HI-FLOW that uses the Ethernet communication setting of the same control number.
0xFFFFFFFF	Mismatch in type being used (different parameter information transmission method and communication type)	Check whether the communication method of the Ethernet communication setting is the same as the communication type of the ladder/HI-FLOW program.

Error types
0x8XXXXXXX: CPMS socket macro error (adds 0x80000000 to the actual CPMS macro error)
0xFXXXXXXX: Error in the system program or task

The error log at Ethernet communication is described below.
Eight cases of an error trace log are collected for each control number. These can be referenced from the “Display Ethernet communication of Error Log (Ladder and HI-FLOW)” window of the basic system. For how to reference error log information from the “Display Ethernet communication of Error Log (Ladder and HI-FLOW)” window, refer to “USER’S MANUAL BASIC MODULES (Manual number SVE-1-100).”

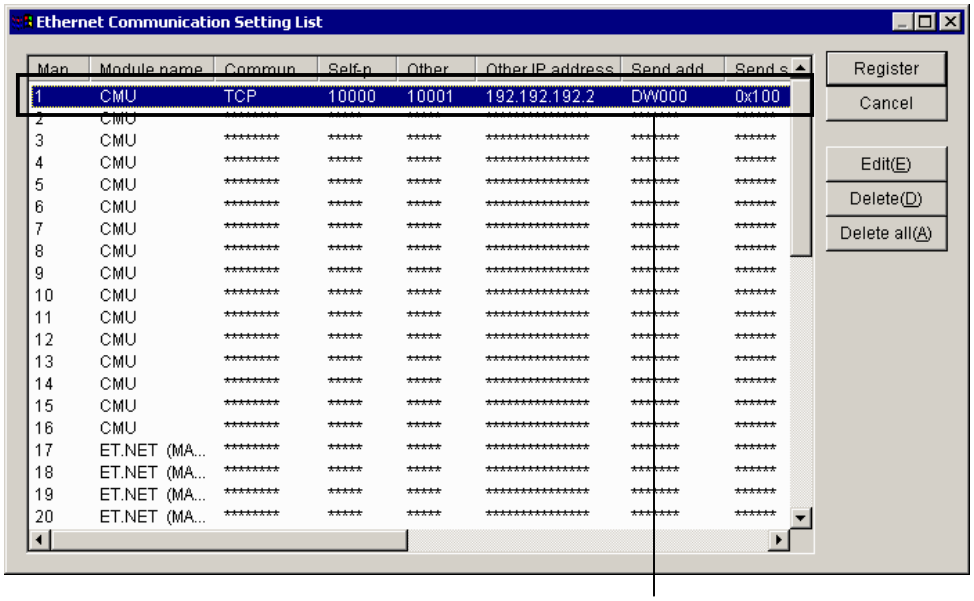
<Display Ethernet communication of Error Log (Ladder and HI-FLOW) window>

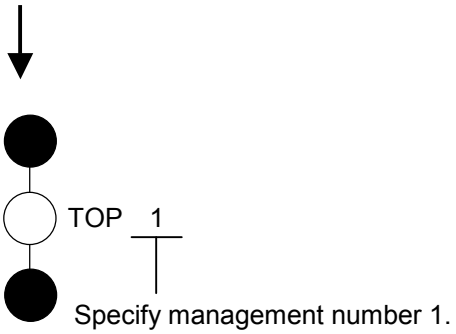


(2) Creating a HI-FLOW program

For parameters of applied instructions for Ethernet communication, specify only the management numbers specified on [Set Ethernet Communication] window and create a HI-FLOW program. Applied instructions for Ethernet communication function according to the information specified on [Set Ethernet Communication] window by management number.

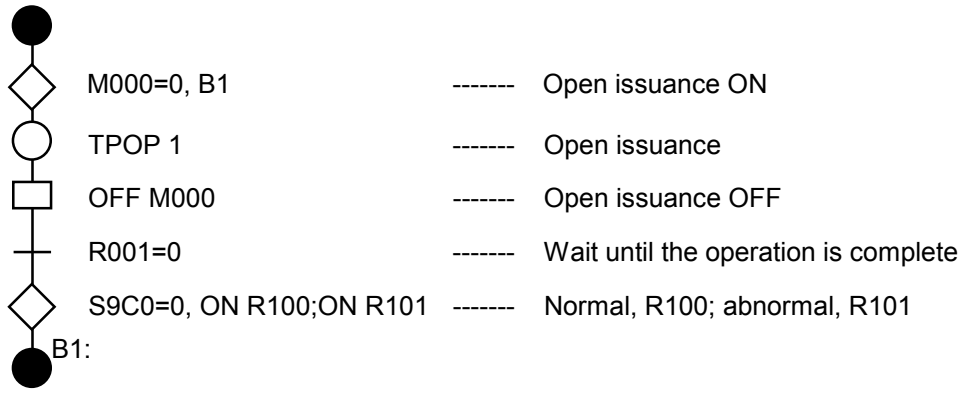
<A typical implementation>





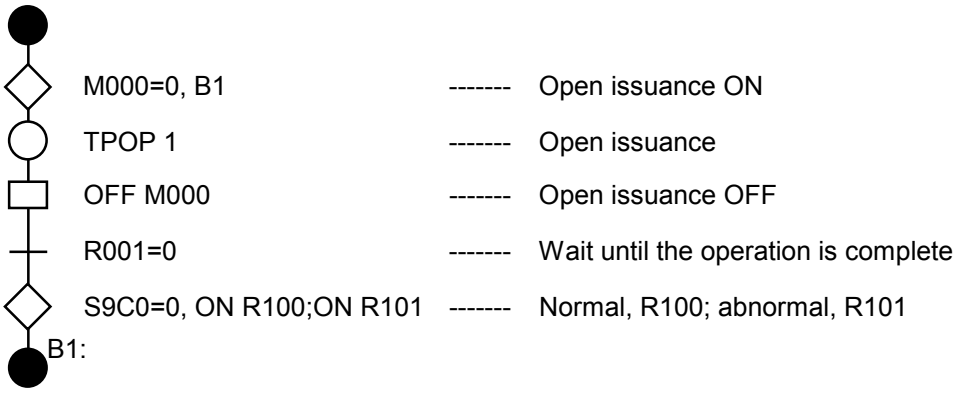
5.7.3 Function description

This section gives explanations in the same manner as "5.6 Function Description."
The system registers in the description (S9C0 to S9FF) are for storing the execution results of applied instructions for Ethernet communication.

<table>
<tr><td>TOP</td><td>Open a TCP connection (client)</td></tr>
<tr><td>Function description</td><td>This function opens a TCP connection as a server.</td></tr>
<tr><td>Parameter and operation</td><td>TPOP S
S: Source (communication identifier)
Specify the control number shown below according to the module.
<table>
<tr><th>Management No.</th><th>Module name</th></tr>
<tr><td>1 to 16</td><td>CMU</td></tr>
<tr><td>17 to 32</td><td>ET.NET (main)</td></tr>
<tr><td>33 to 48</td><td>ET.NET (sub)</td></tr>
<tr><td>49 to 80</td><td>OPTET (Module 0 to 3)</td></tr>
</table>
Up to 16 TCP and UDP connections per module can be used for communication if the module used is other than OPTET.
If the module is an OPTET module and a total of four OPTET modules are used in the user system, then up to 32 TCP and UDP connections can be used for communication via the OPTET modules.</td></tr>
<tr><td>Flag configuration</td><td>When an error occurs in the number of parameters or data type, the E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When a TCP connection is successfully opened (server), the system will set the system register to 0 and set the details result code to 0. If it fails, the system will set the system register to 1 and set the details result code to an error number (not 0).
Determine with the system register whether the operation succeeded or failed.
To identify the cause of the error in the case of a failure, see the Details Result Codes.</td></tr>
<tr><td>Typical use</td><td>Opened when management number = 1, execution flag = R001
 
</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>–</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>–</td><td>–</td></tr>
</table>
</td></tr>
</table>

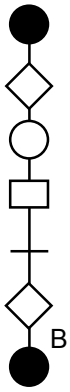
<table>
<tr><td>TPOP</td><td>Open a TCP connection (server)</td></tr>
<tr><td>Function description</td><td>This function opens a TCP connection as a server.</td></tr>
<tr><td>Parameter and operation</td><td> 

S: Source (communication identifier)
Specify the control number shown below according to the module.
<table>
<tr><th>Management No.</th><th>Module name</th></tr>
<tr><td>1 to 16</td><td>CMU</td></tr>
<tr><td>17 to 32</td><td>ET.NET (main)</td></tr>
<tr><td>33 to 48</td><td>ET.NET (sub)</td></tr>
<tr><td>49 to 80</td><td>OPTET (Module 0 to 3)</td></tr>
</table>
Up to 16 TCP and UDP connections per module can be used for communication if the module used is other than OPTET.
If the module is an OPTET module and a total of four OPTET modules are used in the user system, then up to 32 TCP and UDP connections can be used for communication via the OPTET modules.</td></tr>
<tr><td>Flag configuration</td><td>When an error occurs in the number of parameters or data type, the E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When a TCP connection is successfully opened (server), the system will set the system register to 0 and set the details result code to 0. If it fails, the system will set the system register to 1 and set the details result code to an error number (not 0).
Determine with the system register whether the operation succeeded or failed.
To identify the cause of the error in the case of a failure, see the Details Result Codes.</td></tr>
<tr><td>Typical use</td><td>Opened when management number = 1, execution flag = R001
 
</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>–</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>–</td><td>–</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>TCLO</td><td>Close a TCP connection</td></tr>
<tr><td>Function description</td><td>This function closes a TCP connection.</td></tr>
<tr><td>Parameter and operation</td><td>

TCLO S

S: Source (communication identifier)
Specify the control number shown below according to the module.
<table>
<tr><th>Management No.</th><th>Module name</th></tr>
<tr><td>1 to 16</td><td>CMU</td></tr>
<tr><td>17 to 32</td><td>ET.NET (main)</td></tr>
<tr><td>33 to 48</td><td>ET.NET (sub)</td></tr>
<tr><td>49 to 80</td><td>OPTET (Module 0 to 3)</td></tr>
</table>
Up to 16 TCP and UDP connections per module can be used for communication if the module used is other than OPTET.
If the module is an OPTET module and a total of four OPTET modules are used in the user system, then up to 32 TCP and UDP connections can be used for communication via the OPTET modules.
</td></tr>
<tr><td>Flag configuration</td><td>When an error occurs in the number of parameters or data type, the E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When a TCP connection is successfully closed, the system will set the system register to 0 and set the details result code to 0. If it fails, the system will set the system register to 1 and set the details result code to an error number (not 0).
Determine with the system register whether the operation succeeded or failed.
To identify the cause of the error in the case of a failure, see the Details Result Codes.</td></tr>
<tr><td>Typical use</td><td>
Opened when management number = 1, execution flag = R001


</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>–</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>–</td><td>–</td></tr>
</table>
</td></tr>
</table>

(1/2)

<table>
<tr><td>TRCV</td><td>Receive via TCP</td></tr>
<tr><td>Function description</td><td>This function receives data according to the Ethernet settings.</td></tr>
<tr><td>Parameter and operation</td><td> 

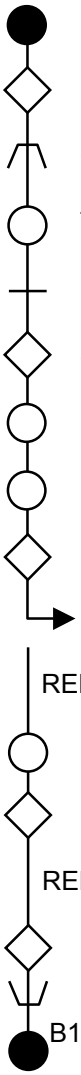
S: Source (communication identifier)

Specify the control number shown below according to the module.
<table>
<tr><th>Management No.</th><th>Module name</th></tr>
<tr><td>1 to 16</td><td>CMU</td></tr>
<tr><td>17 to 32</td><td>ET.NET (main)</td></tr>
<tr><td>33 to 48</td><td>ET.NET (sub)</td></tr>
<tr><td>49 to 80</td><td>OPTET (Module 0 to 3)</td></tr>
</table>
Up to 16 TCP and UDP connections per module can be used for communication if the module used is other than OPTET.
If the module is an OPTET module and a total of four OPTET modules are used in the user system, then up to 32 TCP and UDP connections can be used for communication via the OPTET modules.</td></tr>
<tr><td>Flag configuration</td><td>When an error occurs in the number of parameters or data type, the E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When a TCP is successfully received, the system will set the received data size into the details result code and set the details result code to 0. If it fails, the system will set the system register to 1 and set the details result code to an error number (negative value). Determine with the system register whether the operation succeeded or failed.
To identify the cause of the error in the case of a failure, see the Details Result Codes. If the details result code is EWOULDBLOCK, the system allows you to reissue a TRCV. If the details result code is EWOULDBLOCK and if you wish to continue to receive data, reissue a TRCV.</td></tr>
</table>

(2/2)

Typical use

The system will receive data with management number = 1, operation complete flag = R001, details result code = LWL0000, and receive size = 1,024 bytes. If no data is received, the system retries the reception.

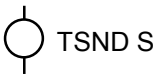


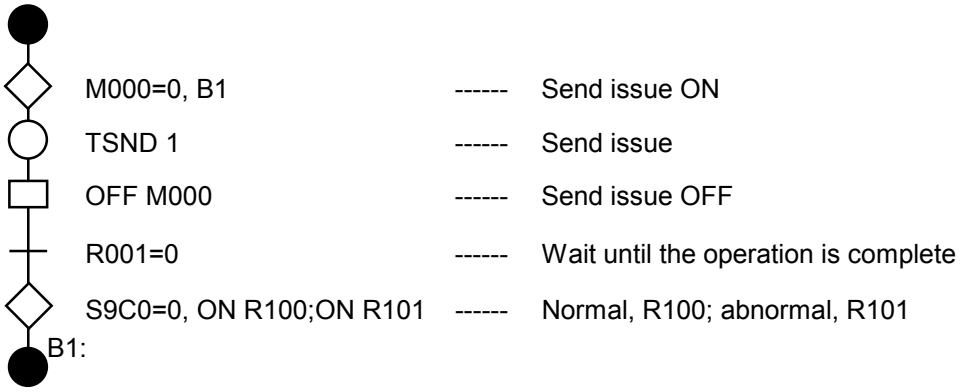
```
M000=0, B1          -------  Reception issuance ON
CN000 (0, 1, 0)     -------  Endless loop
TRCV 1              -------  Reception issuance
R001=0              -------  Wait for the operation to be completed
S9C0, RERR          -------  Abnormal termination (jump to the determined details result code)
EQU [LWW0000], [1024], R104  -------  Normal termination (receive size = received data)
NEQ [LWW0000], [1024], R106  -------  Normal termination (receive size > received data)
R104 | R106, OFF M000  -------  Reception issuance OFF
REND                -------  Jump to the determined reception end
RERR:               -------  Label for determining the details result code
EQU [LWW0000], H800000F6, R003  -------  Determine whether it is EWOULDBLOCK
R003=0, OFF M000:ON R103  -------  Reception ends in abnormal termination, reception retry in EWOULDBLOCK
REND:               -------  Label for determining reception end
M000=0, B1          -------  Reception termination when reception issuance is off
B1:
```

Effective parameter

△ is an address. Parameter error if the number is odd.

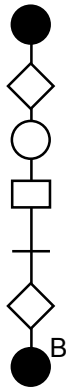
S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	–	√	√
Direct long length	–	–	–
Indirect word length	–	△	△
Indirect long length	–	–	–

<table>
<tr><td>TSND</td><td>Send via TCP</td></tr>
<tr><td>Function description</td><td>This function sends data according to the Ethernet settings.</td></tr>
<tr><td>Parameter and operation</td><td> 

S: Source (communication identifier)
Specify the control number shown below according to the module.
<table>
<tr><th>Management No.</th><th>Module name</th></tr>
<tr><td>1 to 16</td><td>CMU</td></tr>
<tr><td>17 to 32</td><td>ET.NET (main)</td></tr>
<tr><td>33 to 48</td><td>ET.NET (sub)</td></tr>
<tr><td>49 to 80</td><td>OPTET (Module 0 to 3)</td></tr>
</table>
Up to 16 TCP and UDP connections per module can be used for communication if the module used is other than OPTET.
If the module is an OPTET module and a total of four OPTET modules are used in the user system, then up to 32 TCP and UDP connections can be used for communication via the OPTET modules.</td></tr>
<tr><td>Flag configuration</td><td>When an error occurs in the number of parameters or data type, the E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When a TCP is successfully sent, the system will set the details result code to 0 and set the details result code to 0. If it fails, the system will set the system register to 1 and set the details result code to an error number (not 0).
Determine with the system register whether the operation succeeded or failed.
To identify the cause of the error in the case of a failure, see the Details Result Codes.</td></tr>
<tr><td>Typical use</td><td>Opened when management number = 1, execution flag = R001
 
</td></tr>
<tr><td>Effective parameter

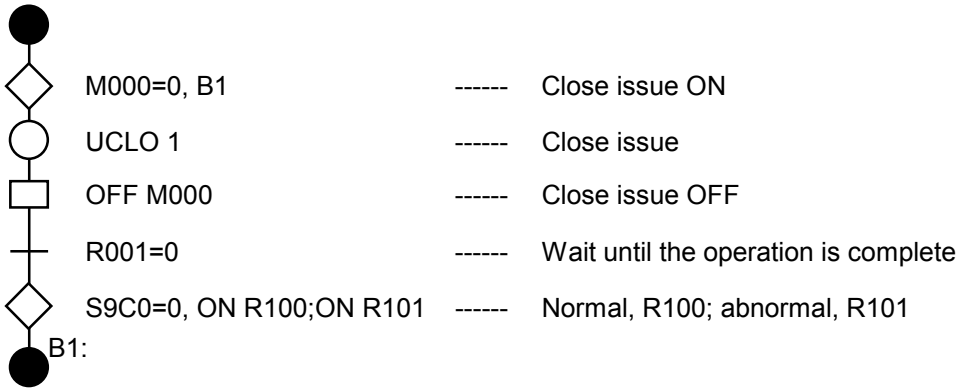
△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>–</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>–</td><td>–</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>UOP</td><td>Open a UDP connection</td></tr>
<tr><td>Function description</td><td>This function opens a UDP connection.</td></tr>
<tr><td>Parameter and operation</td><td> 
UOP S

S: Source (communication identifier)
Specify the control number shown below according to the module.
<table><tr><th>Management No.</th><th>Module name</th></tr><tr><td>1 to 16</td><td>CMU</td></tr><tr><td>17 to 32</td><td>ET.NET (main)</td></tr><tr><td>33 to 48</td><td>ET.NET (sub)</td></tr><tr><td>49 to 80</td><td>OPTET (Module 0 to 3)</td></tr></table>Up to 16 TCP and UDP connections per module can be used for communication if the module used is other than OPTET.
If the module is an OPTET module and a total of four OPTET modules are used in the user system, then up to 32 TCP and UDP connections can be used for communication via the OPTET modules.</td></tr>
<tr><td>Flag configuration</td><td>When an error occurs in the number of parameters or data type, the E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When a UDP is successfully opened, the system will set the details result code to 0 and set the details result code to 0. If it fails, the system will set the system register to 1 and set the details result code to an error number (not 0).
Determine with the system register whether the operation succeeded or failed.
To identify the cause of the error in the case of a failure, see the Details Result Codes.</td></tr>
<tr><td>Typical use</td><td>Opened when management number = 1, execution flag = R001
 
M000=0, B1 ------ Open issuance ON
UOP 1 ------ Open issuance
OFF M000 ------ Open issuance OFF
R001=0 ------ Wait until the operation is complete
S9C0=0, ON R100;ON R101 ------ Normal, R100; abnormal, R101
B1:</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td><table><tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr><tr><td>Direct word length</td><td>–</td><td>√</td><td>√</td></tr><tr><td>Direct long length</td><td>–</td><td>–</td><td>–</td></tr><tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr><tr><td>Indirect long length</td><td>–</td><td>–</td><td>–</td></tr></table></td></tr>
</table>

<table>
<tr><td>UCLO</td><td>Close a UDP connection</td></tr>
<tr><td>Function description</td><td>This function closes a UDP connection.</td></tr>
<tr><td>Parameter and operation</td><td> 

S: Source (communication identifier)
Specify the control number shown below according to the module.
<table>
<tr><th>Management No.</th><th>Module name</th></tr>
<tr><td>1 to 16</td><td>CMU</td></tr>
<tr><td>17 to 32</td><td>ET.NET (main)</td></tr>
<tr><td>33 to 48</td><td>ET.NET (sub)</td></tr>
<tr><td>49 to 80</td><td>OPTET (Module 0 to 3)</td></tr>
</table>
Up to 16 TCP and UDP connections per module can be used for communication if the module used is other than OPTET.
If the module is an OPTET module and a total of four OPTET modules are used in the user system, then up to 32 TCP and UDP connections can be used for communication via the OPTET modules.</td></tr>
<tr><td>Flag configuration</td><td>When an error occurs in the number of parameters or data type, the E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When a UDP is successfully closed, the system will set the details result code to 0 and set the details result code to 0. If it fails, the system will set the system register to 1 and set the details result code to an error number (not 0).
Determine with the system register whether the operation succeeded or failed.
To identify the cause of the error in the case of a failure, see the Details Result Codes.</td></tr>
<tr><td>Typical use</td><td>Opened when management number = 1, execution flag = R001
 
</td></tr>
<tr><td>Effective parameter

△ is an address. Parameter error if the number is odd.</td><td>
<table>
<tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr>
<tr><td>Direct word length</td><td>–</td><td>√</td><td>√</td></tr>
<tr><td>Direct long length</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr>
<tr><td>Indirect long length</td><td>–</td><td>–</td><td>–</td></tr>
</table>
</td></tr>
</table>

(1/2)

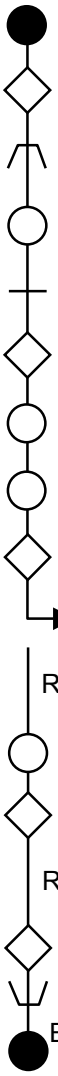
<table>
<tr><td>URCV</td><td>Receive via UDP</td></tr>
<tr><td>Function description</td><td>This function receives data according to the Ethernet settings.</td></tr>
<tr><td>Parameter and operation</td><td>  URCV S

S: Source (communication identifier)
Specify the control number shown below according to the module.
<table>
<tr><th>Management No.</th><th>Module name</th></tr>
<tr><td>1 to 16</td><td>CMU</td></tr>
<tr><td>17 to 32</td><td>ET.NET (main)</td></tr>
<tr><td>33 to 48</td><td>ET.NET (sub)</td></tr>
<tr><td>49 to 80</td><td>OPTET (Module 0 to 3)</td></tr>
</table>
Up to 16 TCP and UDP connections per module can be used for communication if the module used is other than OPTET.
If the module is an OPTET module and a total of four OPTET modules are used in the user system, then up to 32 TCP and UDP connections can be used for communication via the OPTET modules.</td></tr>
<tr><td>Flag configuration</td><td>When an error occurs in the number of parameters or data type, the E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When a TCP is successfully received, the system will set the received data size into the details result code and set the details result code to 0. If it fails, the system will set the system register to 1 and set the details result code to an error number (negative value). Determine with the system register whether the operation succeeded or failed.
To identify the cause of the error in the case of a failure, see the Details Result Codes. If the details result code is EWOULDBLOCK, the system allows you to reissue a TRCV. If the details result code is EWOULDBLOCK and if you wish to continue to receive data, reissue a TRCV.</td></tr>
</table>

(2/2)

Typical use

The system will receive data with management number = 1, operation complete flag = R001, details result code = LWL0000, and receive size = 1,024 bytes. If no data is received, the system retries the reception.



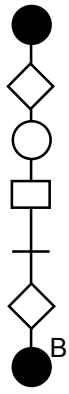
```
M000=0, B1          ------  Reception issuance ON
CN000 (0, 1, 0)     ------  Endless loop
TRCV 1              ------  Reception issuance
R001=0              ------  Wait for the operation to be completed
S9C0, RERR          ------  Abnormal termination (jump to the determined details result code)
EQU [LWW0000], [1024], R104  ------  Normal termination (receive size = received data)
NEQ [LWW0000], [1024], R106  ------  Normal termination (receive size > received data)
R104 | R106, OFF M000  ------  Reception issuance OFF
REND                ------  Jump to the determined reception end
RERR:               ------  Label for determining the details result code
EQU [LWW0000], H800000F6, R003  ------  Determine whether it is EWOULDBLOCK
R003=0, OFF M000:ON R103  ------  Reception ends in abnormal termination, reception retry in EWOULDBLOCK
REND:               ------  Label for determining reception end
M000=0, B1          ------  Reception termination when reception issuance is off
B1:
```

Effective parameter

△ is an address. Parameter error if the number is odd.

S	Bit-type PI/O	Word-type PI/O	Constant
Direct word length	–	√	√
Direct long length	–	–	–
Indirect word length	–	△	△
Indirect long length	–	–	–

<table>
<tr><td>USND</td><td>Send via UDP</td></tr>
<tr><td>Function description</td><td>This function sends data according to the Ethernet settings.</td></tr>
<tr><td>Parameter and operation</td><td>  USND S

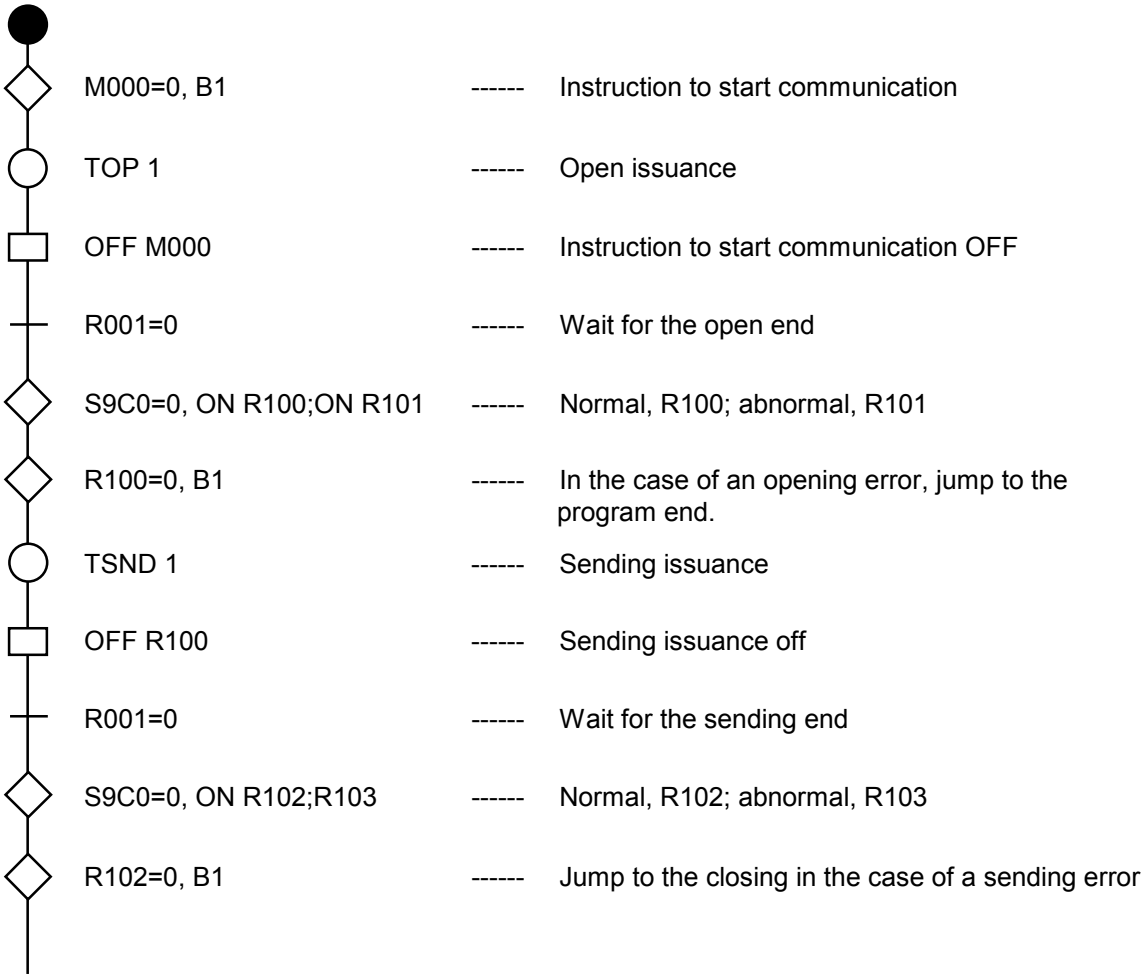
S: Source (communication identifier)
Specify the control number shown below according to the module.
<table><tr><th>Management No.</th><th>Module name</th></tr><tr><td>1 to 16</td><td>CMU</td></tr><tr><td>17 to 32</td><td>ET.NET (main)</td></tr><tr><td>33 to 48</td><td>ET.NET (sub)</td></tr><tr><td>49 to 80</td><td>OPTET (Module 0 to 3)</td></tr></table>Up to 16 TCP and UDP connections per module can be used for communication if the module used is other than OPTET.
If the module is an OPTET module and a total of four OPTET modules are used in the user system, then up to 32 TCP and UDP connections can be used for communication via the OPTET modules.</td></tr>
<tr><td>Flag configuration</td><td>When an error occurs in the number of parameters or data type, the E will change. The others will become turned off.</td></tr>
<tr><td>Remark</td><td>When a TCP is successfully sent, the system will set the details result code to 0 and set the details result code to 0. If it fails, the system will set the system register to 1 and set the details result code to an error number (not 0).
Determine with the system register whether the operation succeeded or failed.
To identify the cause of the error in the case of a failure, see the Details Result Codes.</td></tr>
<tr><td>Typical use</td><td>Opened when management number = 1, execution flag = R001
 
M000=0, B1 ------ Send issue ON
USND 1 ------ Send issue
OFF M000 ------ Send issue OFF
R001=0 ------ Wait until the operation is complete
S9C0=0, ON R100;ON R101 ------ Normal, R100; abnormal, R101
B1:</td></tr>
<tr><td>Effective parameter

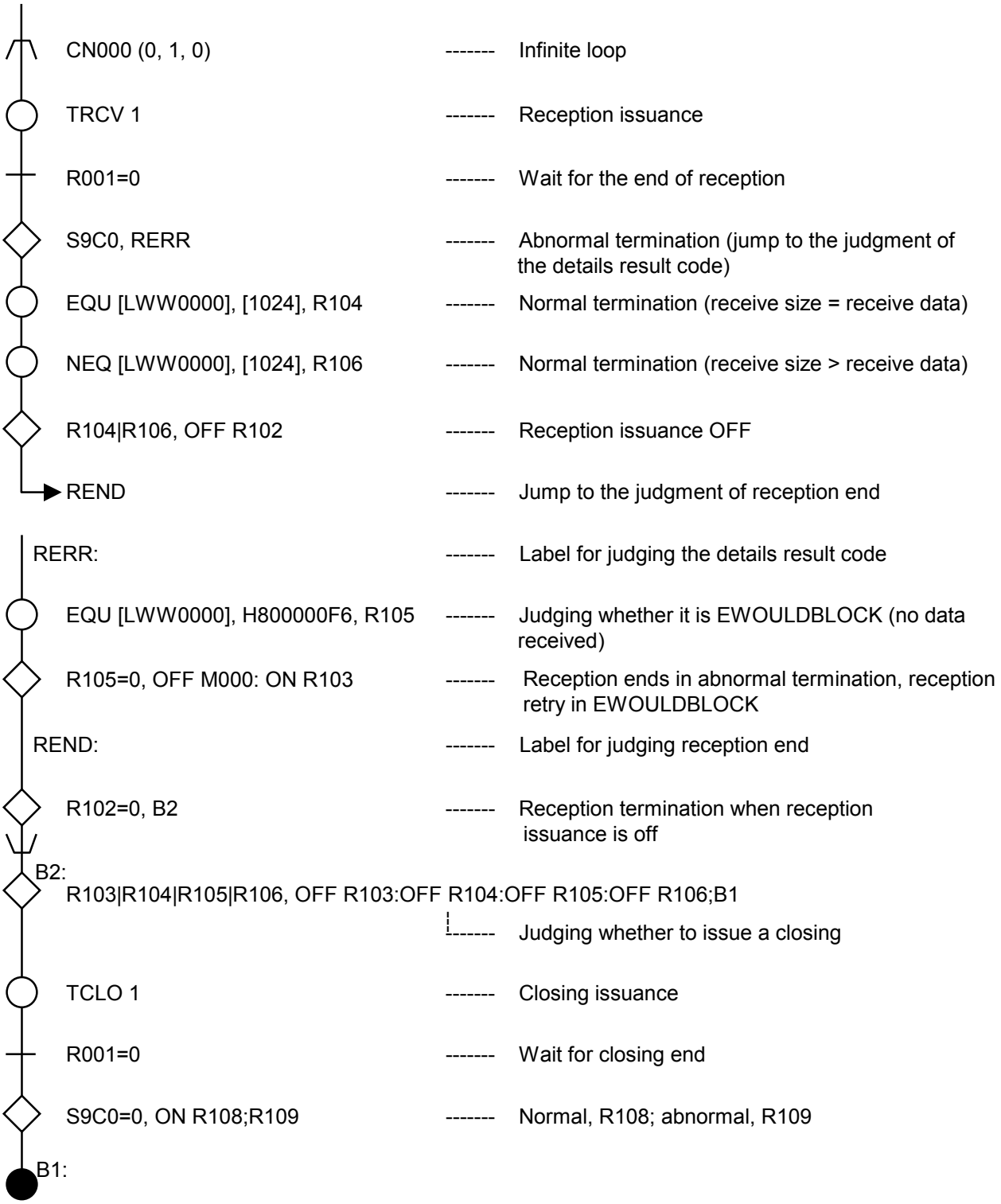
△ is an address. Parameter error if the number is odd.</td><td><table><tr><th>S</th><th>Bit-type PI/O</th><th>Word-type PI/O</th><th>Constant</th></tr><tr><td>Direct word length</td><td>–</td><td>√</td><td>√</td></tr><tr><td>Direct long length</td><td>–</td><td>–</td><td>–</td></tr><tr><td>Indirect word length</td><td>–</td><td>△</td><td>△</td></tr><tr><td>Indirect long length</td><td>–</td><td>–</td><td>–</td></tr></table></td></tr>
</table>

5.7.4 Sample program

This section describes a sample program for operations from the opening of a socket to communication until the closing of the socket by means of applied instructions for Ethernet communication.
This sample program consists of parameter settings with management number = 1, execution flag = R001, and details result code = LWL0000.

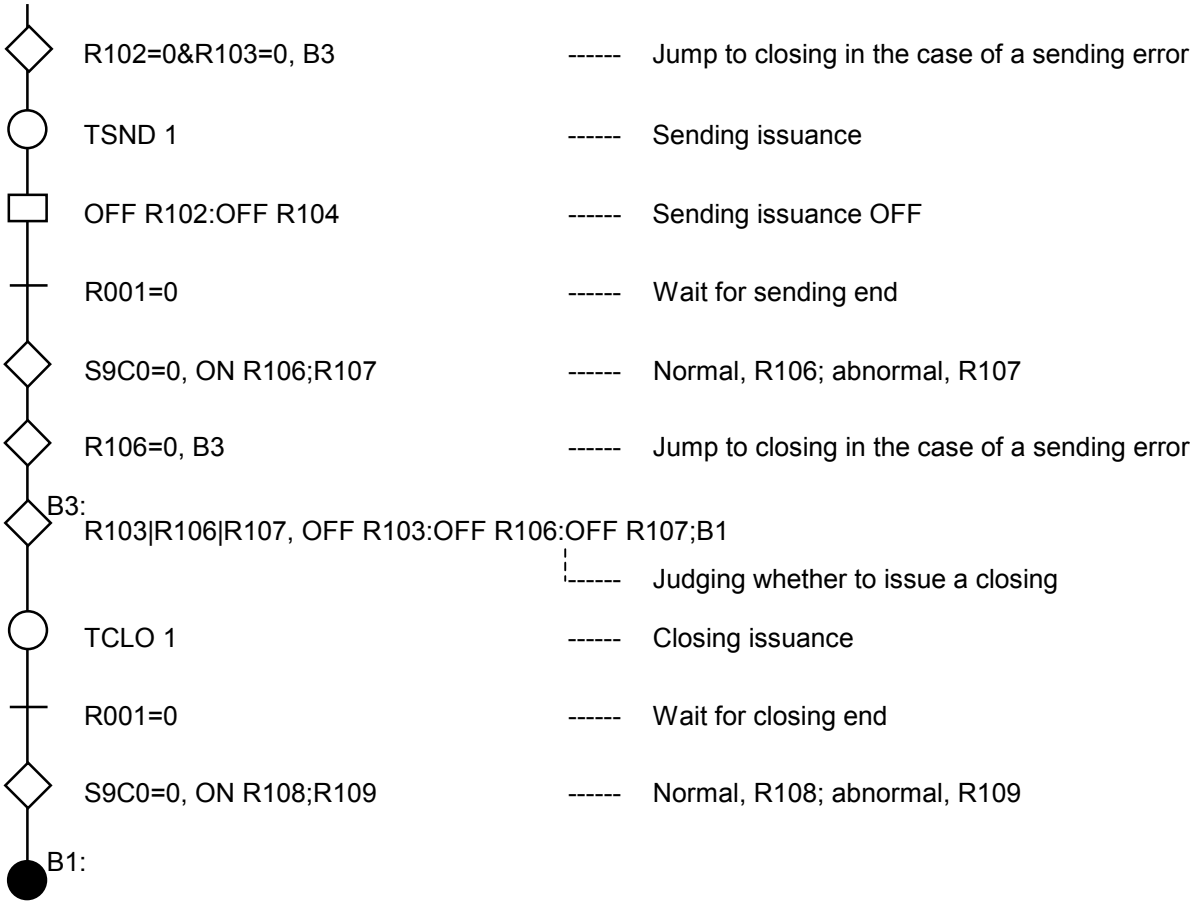
(1) TCP client




CN000 (0, 1, 0) ------- Infinite loop
TRCV 1 ------- Reception issuance
R001=0 ------- Wait for the end of reception
S9C0, RERR ------- Abnormal termination (jump to the judgment of the details result code)
EQU [LWW0000], [1024], R104 ------- Normal termination (receive size = receive data)
NEQ [LWW0000], [1024], R106 ------- Normal termination (receive size > receive data)
R104|R106, OFF R102 ------- Reception issuance OFF
REND ------- Jump to the judgment of reception end
RERR: ------- Label for judging the details result code
EQU [LWW0000], H800000F6, R105 ------- Judging whether it is EWOULDBLOCK (no data received)
R105=0, OFF M000: ON R103 ------- Reception ends in abnormal termination, reception retry in EWOULDBLOCK
REND: ------- Label for judging reception end
R102=0, B2 ------- Reception termination when reception issuance is off
B2:
R103|R104|R105|R106, OFF R103:OFF R104:OFF R105:OFF R106;B1
------- Judging whether to issue a closing
TCLO 1 ------- Closing issuance
R001=0 ------- Wait for closing end
S9C0=0, ON R108;R109 ------- Normal, R108; abnormal, R109
B1:

(2) TCP server

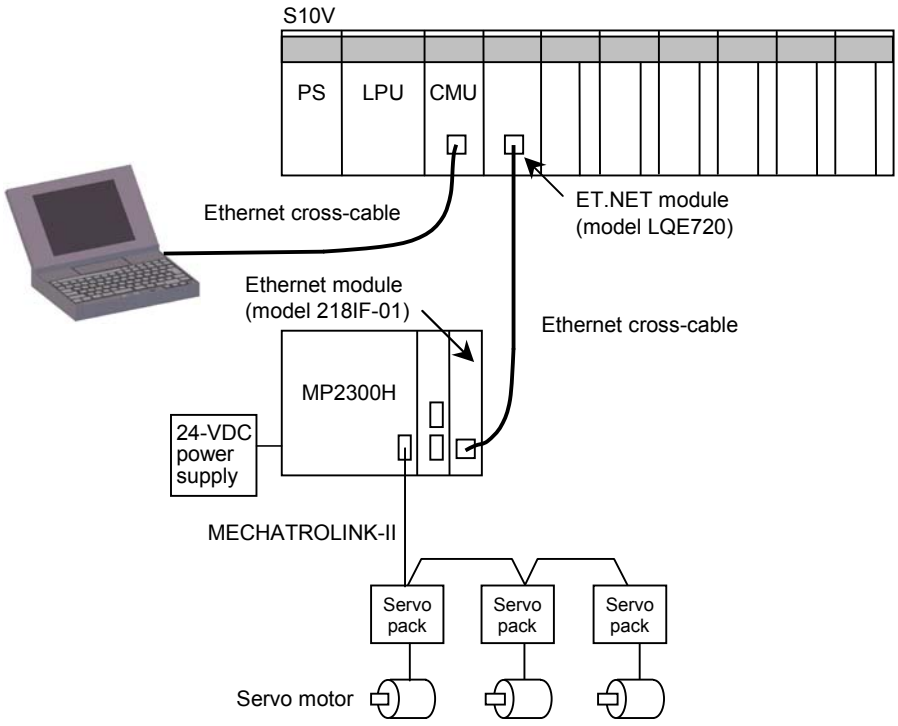
M000=0, B1	------	Instruction to start communication
TOP 1	------	Opening issuance
OFF M000	------	Instruction to start communication OFF
R001=0	------	Wait for opening end
S9C0=0, ON R100;ON R101	-------	Normal, R100; abnormal, R101
R100=0, B1	------	Jump to the program end in an opening error
CN000 (0, 1, 0)	------	Infinite loop
TRCV 1	------	Reception issuance
R001=0	------	Wait for reception end
S9C0, RERR	------	Abnormal termination (jump to the judgment of the details results code)
EQU [LWW0000], [1024], R102	------	Normal termination (receive size = receive data)
NEQ [LWW0000], [1024], R104	------	Normal termination (receive size > receive data)
R102|R104, OFF R100	------	Reception issuance OFF
REND	------	Jump to the judgment of reception end
RERR:	------	Label for judging the details result code
EQU [LWW0000], H800000F6, R103	------	Judging whether it is EWOULDBLOCK (no data received)
R103=0, OFF R100:ON R105	------	Reception ends in abnormal termination, reception retry in EWOULDBLOCK
REND:	------	Label for judging reception end
R100=0, B2	------	Reception termination when reception issuance is off


R102=0&R103=0, B3 ------ Jump to closing in the case of a sending error
TSND 1 ------ Sending issuance
OFF R102:OFF R104 ------ Sending issuance OFF
R001=0 ------ Wait for sending end
S9C0=0, ON R106;R107 ------ Normal, R106; abnormal, R107
R106=0, B3 ------ Jump to closing in the case of a sending error
B3:
R103|R106|R107, OFF R103:OFF R106:OFF R107;B1
------ Judging whether to issue a closing
TCLO 1 ------ Closing issuance
R001=0 ------ Wait for closing end
S9C0=0, ON R108;R109 ------ Normal, R108; abnormal, R109
B1:

6 MOTION CONTROL INSTRUCTIONS

6.1 Purpose

The purpose of motion control instructions is to control given servo motors directly under HI-FLOW by means of a model MP2300H motion controller (hereinafter simply called an MP2300H controller) of Yaskawa Electric Corp. connected with the S10V controller.



HI-FLOW can control up to 32 axes at a time.

6.2 Specifications

6.2.1 System configuration

The use of motion control instructions requires the prior installation of the following hardware modules of the versions listed below.

Module name	Model	Ver-Rev
LPU	LQP510	02-02 or later
CMU	LQP520	04-00 or later
ET.NET	LQE720	01-00 or later

The above Ver-Rev numbers are those of the microprograms for the individual modules that can be found in the “Module List” presented on-screen by the S10V base system.

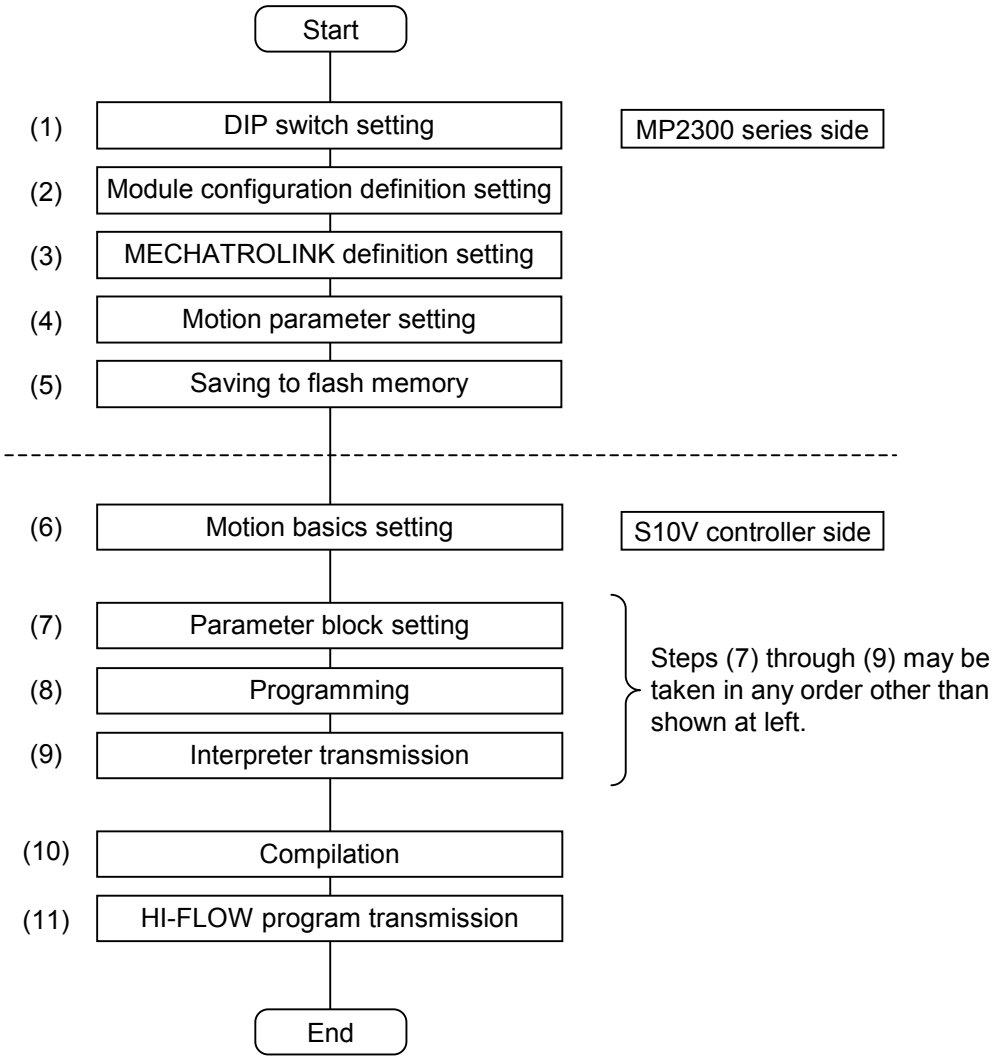
6.2.2 Communication interface between S10V and MP2300H controllers

Item	Specification
Mode of communication	One-to-one (between S10V and MP2300H controllers)
Type of Yaskawa Electric Corp.’s motion controller used	Motion controller (model MP2300H) with an Ethernet® interface unit installed in it
Ports used	34200, 34201, 34205 (in decimal) (Common to both the S10V and the MP2300H controller)
IP addresses used	Arbitrarily selected, except for the network address, which must be identical between the S10V and the MP2300H controller.
Communication protocols used	TCP, UDP

The physical communication line between the S10V and the MP2300H controller must be dedicated only to motion control and must not be connected to anything but the S10V controller and MP2300H controller.

6.3 Usage

Motion control instructions can be used only in programming work that is done in the following way:



■ Setup done on MP2300H controller side

To use the motion module under control of HI-FLOW, you must complete the setup described below.

The setup of the motion module can be done by using the MP-series engineering tool, called MPE720, that comes with the motion module. For information on how to use this tool, refer to the MP2300H series manual or see Supplement 6, "MP2300H System Reconfiguration Procedure."

(1) DIP switch setting

Sets necessary operating conditions for the motion module by DIP switch setting.

(2) Module configuration definition setting

Sets necessary module configuration definitions for the motion module.

(3) MECHATROLINK definition setting

Sets necessary parameters required for using the MECHATROLINK transmission system.

(4) Motion parameter setting

Sets necessary parameters required for performing motion control for each axis.

(5) Saving to flash memory

Saves all the setup information you have supplied to the MP2300H series' flash memory.

■ Setup done on S10V controller side

On S10V controller side, you can complete necessary setup by operating the HI-FLOW system. For information on how to use this tool, refer to the Software Manual, Operation, HI-FLOW For Windows® (Manual number SVE-3-132).

(6) Motion basics setting

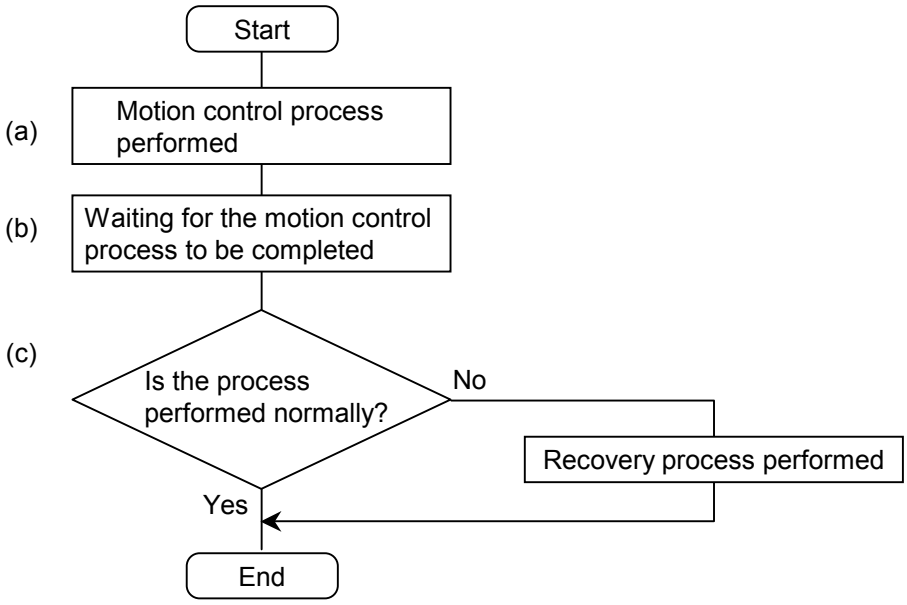
Makes basic settings required for the execution of motion control instructions, such as data communication settings, axis definitions, motion status flag storage setting, and axis-status LPU memory transfer destination setting.

(7) Parameter block setting

Sets parameter values in a parameter block, if necessary. In most cases, this setting is done initially.

(8) Programming

To use motion control instructions successfully, you must check the status of the motion module each time the requested process has been performed. This is shown in the following flowchart.



(a) Motion control processes

There are 19 types of motion control instructions available for motion control processing. These 19 types are listed below.

No.	Command	Function overview
1	Servo ON	Turns on a desired servo(s).
2	Servo OFF	Turns off a desired servo(s).
3	Positioning	Positions a specified axis (or axes) at a desired target position(s) at a desired speed(s).
4	External positioning	Moves a specified axis (or axes) a desired distance(s) if the external positioning signal is turned on during a move(s), and then positions them there
5	Home position return	Causes the system to return to the home position
6	Constant-speed feed	Moves a specified axis (or axes) in a desired direction(s) at a desired speed(s).
7	Command stop	Aborts the command currently under execution
8	Command holding	Temporarily holds the command currently under execution
9	Command reset hold	Releases the command currently under execution from its temporary hold state.
10	Speed control	Changes the current speed(s) of movement in progress for a specified axis (or axes).
11	Speed-position control	Changes the current target position(s) and positioning speed(s) for a specified axis (or axes).
12	Torque control	Sets a torque value(s) or changes the set torque value(s).
13	Speed override	Changes the current percentage value(s) for the set speed(s).
14	Change torque limit	Changes the set torque limits
15	Change speed loop gain	Changes the set speed loop gain(s).
16	Change position loop gain	Changes the set position loop gain(s).
17	Set unit	Changes a desired unit(s) among those currently set for parameter values, such as speed and positioning units and filter type
18	Alarm clear	Clears the alarm(s) for a specified axis (or axes).
19	NOP	Clears all existing motion control instruction information

You can specify any of these motion control instructions or processes through interaction with the “Edit of motion control command” window while the HI-FLOW sheet is displayed on-screen along with the motion control symbol pasted.

(b) Waiting for motion control processes to be completed

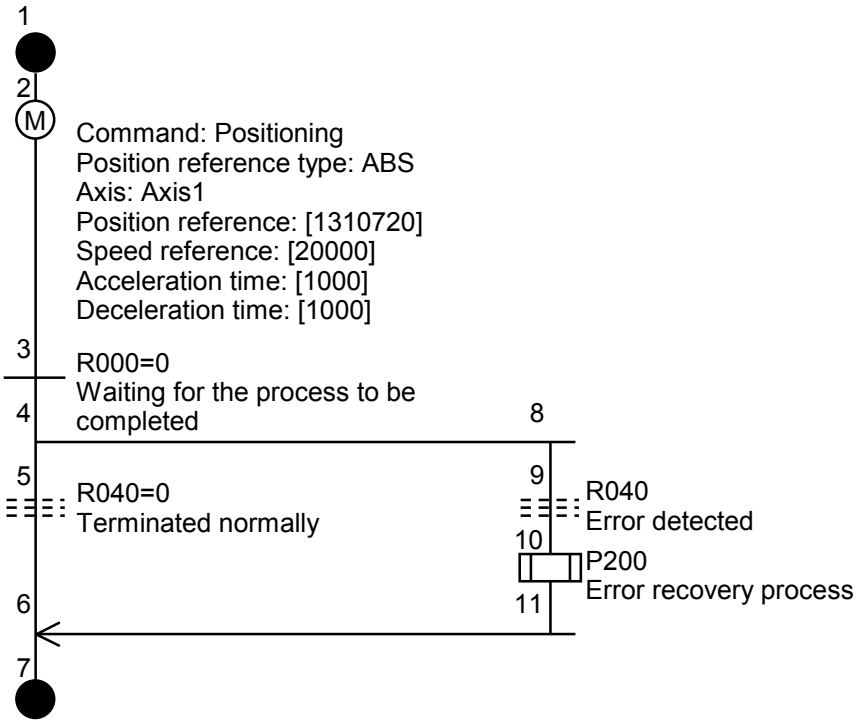
Motion control instructions end their execution before the performance of the corresponding processes is completed. Thus, it is advised that the user who has issued a motion control command wait for the corresponding process to be completed before issuing a next motion control command. The only exception to this is those commands that try to change values of operating parameters during the execution of motion control instructions to change the speed, position, etc. (For details, see functional descriptions of the motion control instructions.)

Whether the performance of a requested process is completed or not can be determined from the "action flag(s)" that have been set in the "Motion basic setup" screen (window) for the axis (or axes) currently subjected to control. If all of the action flags are set to 0 (OFF), they indicate that the performance of a requested process is completed.

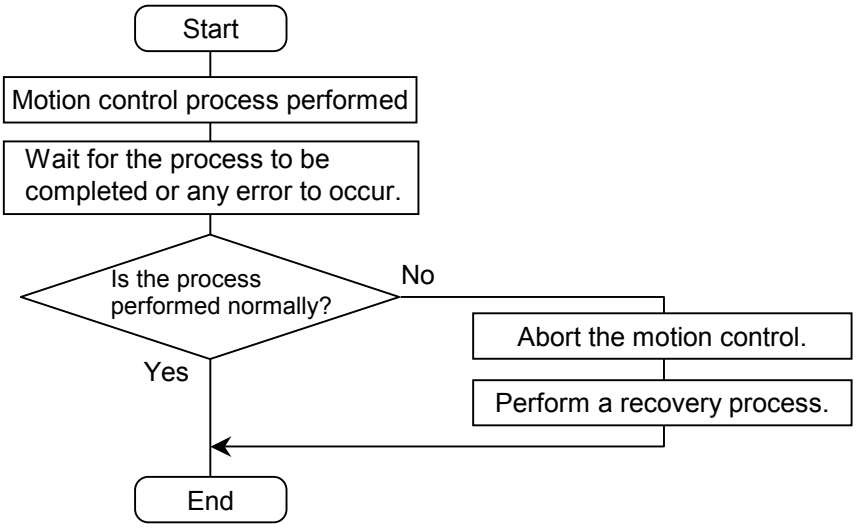
(c) Testing on the result of performing the process

The action flag(s) are set to 0 also when the requested process is terminated abnormally. Thus, to determine if the process is performed successfully or not, check the corresponding command's error flag that indicates whether or not the execution of that command is ended without errors. Detailed information on any detected error is stored in place as command error information. However, the command error information supplied may not detail the detected error if it is an event-like error detected on motion side. In this case, you can identify the error from an alarm issued as axis status information, a warning, and/or motion command status bit 3 (command abend status).

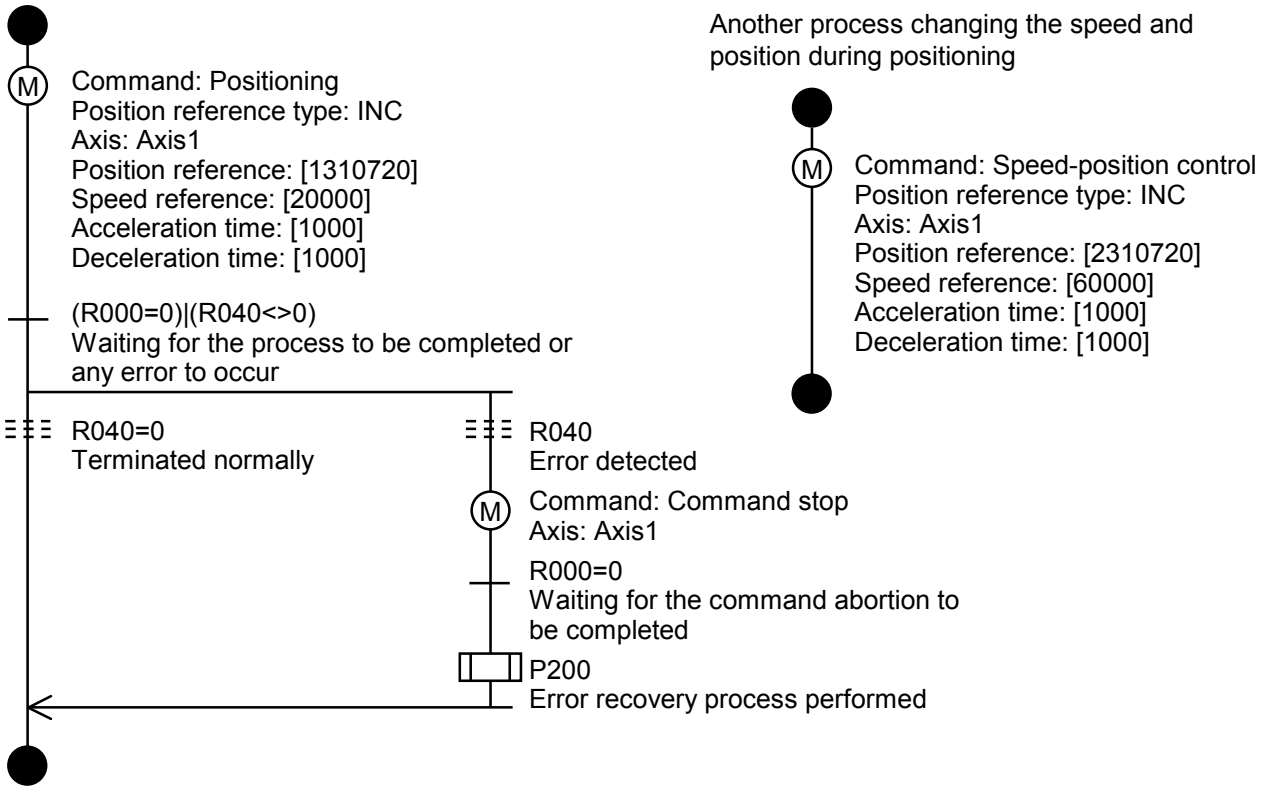
Shown below is an example of a program performing such testing.
(R000: Action flag; R040: Error flag)



If a user-created program is one that changes the value of an operating parameter, such as speed or position, during positioning by a motion control instruction under execution, an error may occur due to the change in the parameter value before the completion of the process. To handle this situation successfully, the program must, as shown below, have a design that waits for not only the motion control process (e.g., positioning) to be completed but also any error to occur. If an error occurs, the process may have not been completed. If so, the design must abort the process.



Shown below is an example of a program having the design described above.
(R000: Action flag; R040: Error flag)



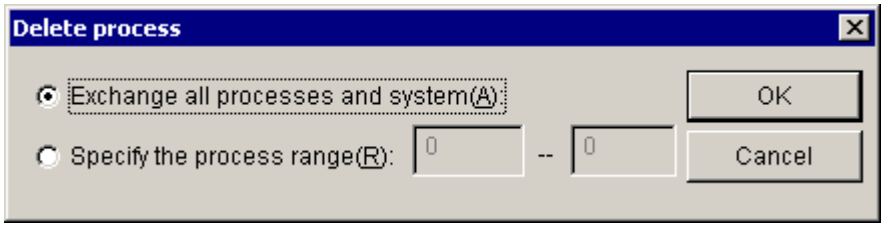
(9) Interpreter transmission

The purpose of interpreter transmission is to replace an existing HI-FLOW system with a new one or incorporate a new HI-FLOW system into the S10V controller system. Interpreter transmission must be initially done to one single real machine (S10V controller).

<Procedure>

Choose [Utility] - [PCs] - [Delete process of PCs] in the HI-FLOW process sheet displayed on-screen.

The "Delete process" window as shown below then appears.



In this window, select the Exchange all processes and system radio button and click the OK button. Then, the existing HI-FLOW system will be replaced.

(10) Compilation

To compile a HI-FLOW program you have produced, choose [Build] from the [Build] menu in the HI-FLOW process sheet. If an error is detected in the HI-FLOW program during the compilation, correct the error according to the error message displayed in the Output bar and try again.

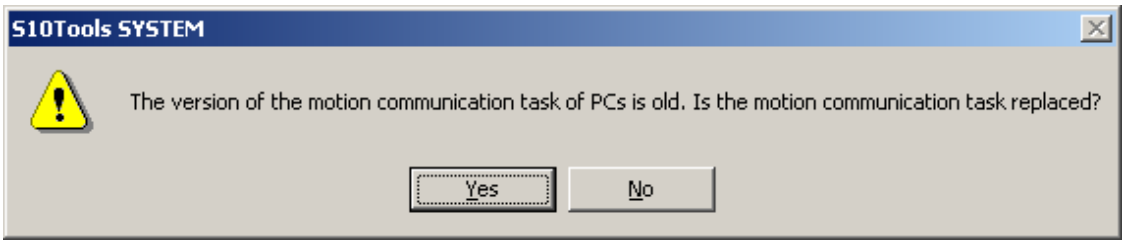
(11) Transmitting a compiled HI-FLOW program to the PCs

To transmit a HI-FLOW program you have compiled to the PCs, choose [Online] - [Send] - [All processes] from the [Mode] menu in the HI-FLOW process sheet.

Prior to this, if an execution environment for the communication task is not set up in the PCs yet, it will be automatically set up by initialization upon completion of an automatic reset operation of the PCs.

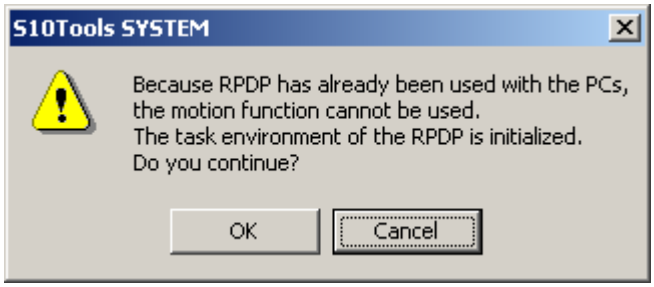
If the communication task is not transmitted to the PCs yet, it will be automatically transmitted together with the compiled HI-FLOW program. Upon completion of the transmission, the PCs will be automatically reset.

The message shown below may be displayed at the time of transmitting the communication task, indicating that some of the new features are not usable. In this case, the user is advised to click the OK button. Clicking OK will replace the old communication task in the PCs with a new one.



CAUTION

- HI-FLOW programs must be transmitted to the PCs together with the communication task if you have sent NX/HOST system files to the same PCs. The reason for this is that, if NX/HOST system files are sent to the PCs from the NX/Tools, any existing tasks that have been registered by a tool other than the NX/Tools will be automatically deleted in the PCs.
- HI-FLOW programs must be transmitted to the PCs together with the communication task if you have executed the CPMS debugger's command to initialize the execution environment of tasks. The reason for this is that, if such a command is executed, all the existing tasks in the PCs will be automatically deleted.
- The communication task is transmitted to the PCs according to the following parameters whose values are specified in the ranges shown below, so do not use any parameter values in those ranges within the CPMS debugger.
  Task number: 206 to 208
  Task storage area: /300E0000 to /300FFFFF ($TASK)
  /50800000 to /509FFFFF ($GLBRW)
- Motion functionality is not usable if the RPDP is used. In this case, if you want to use the motion functionality, click the OK button in the message shown below, which is displayed when transmitting a HI-FLOW program to the PCs. Clicking OK will initialize the execution environment for tasks. You should note that such initialization will cause all of the existing RPDP tasks to be deleted in the PCs.

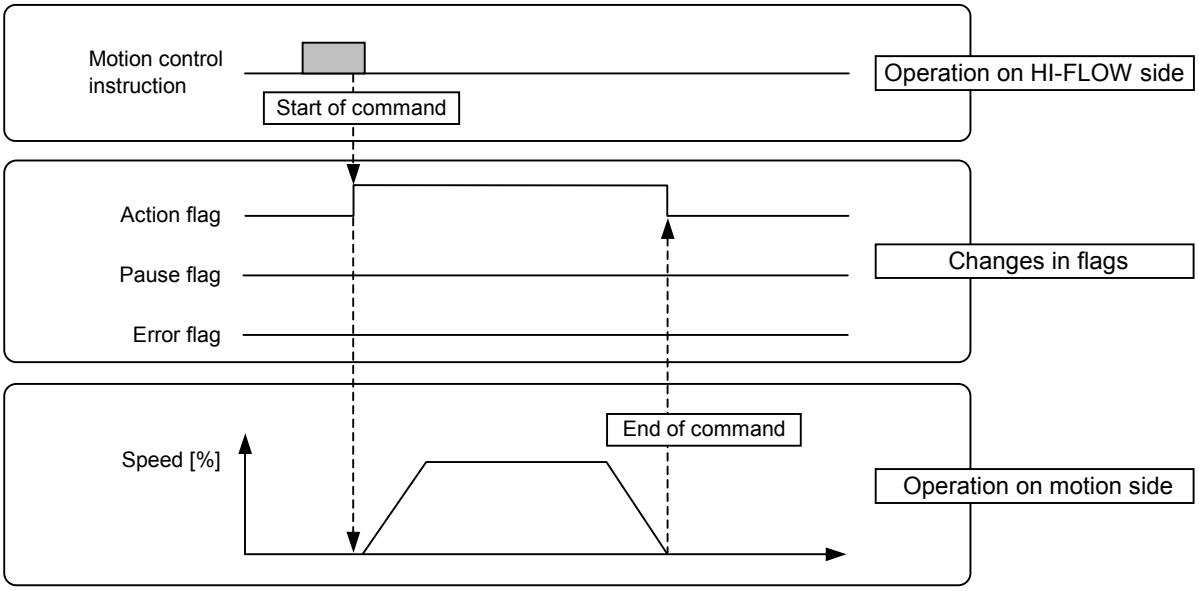


6.4 Motion Status Flags

The motion status flags -- Action flag, Pause flag, and Error flag -- indicate the result of execution of a motion control instruction or the status of motion for a selected axis (or axes). They are used in checking and managing the execution status of motion control instructions, waiting for motion control processes to be completed, or checking and managing the result of processes performed. Each of these flags is detailed below.

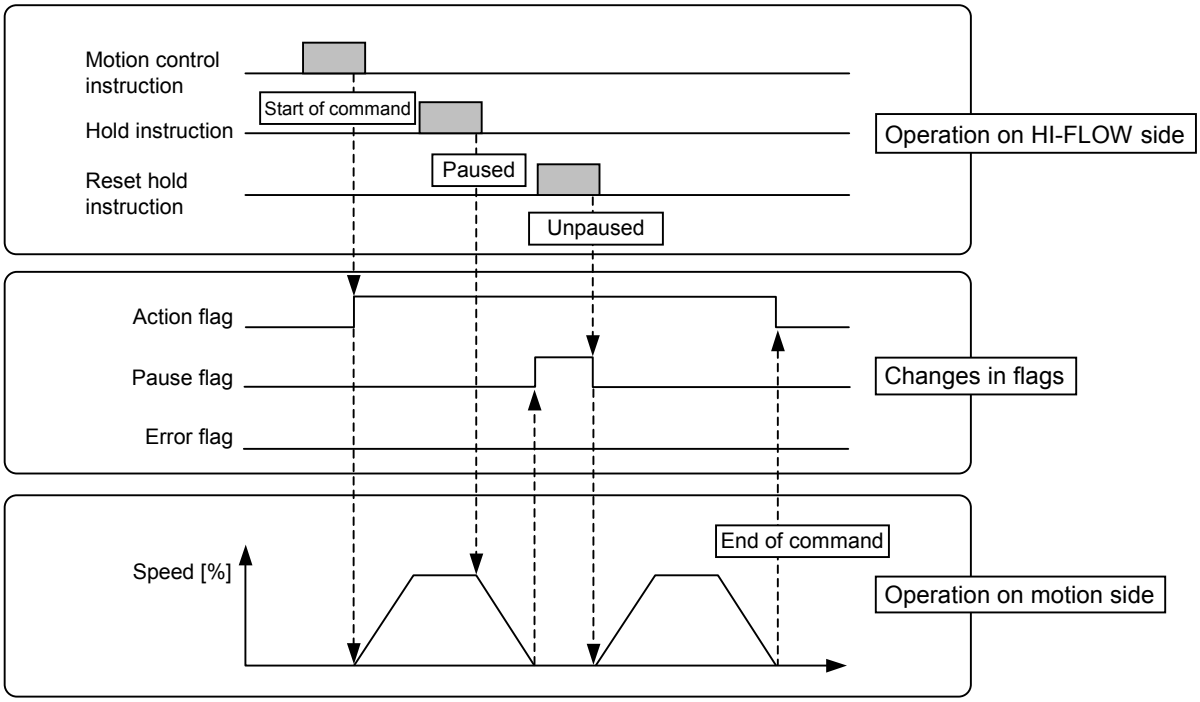
(1) Action flag

The Action flag is provided for the purpose of checking and managing the execution status of motion commands. It is set (= 1) when the execution of a command starts, and is reset (= 0) when it ends. By monitoring the set/reset status of this flag, you can learn the time when the execution of a motion control instruction is completed.



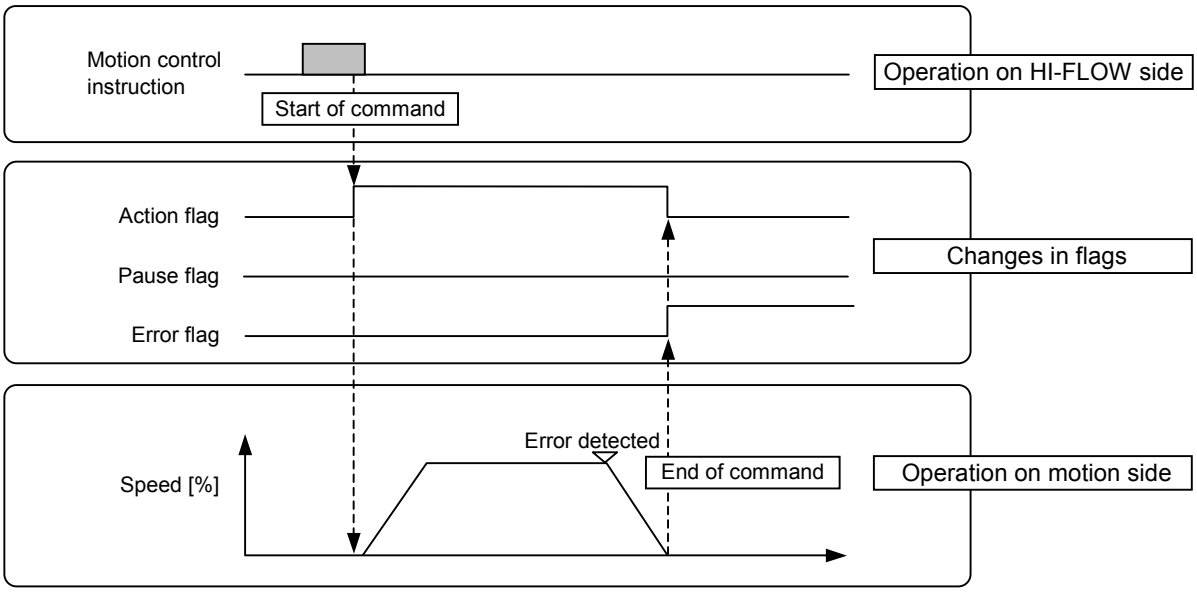
(2) Pause flag

The Pause flag is provided for the purpose of checking and managing the pause (or hold) status of motion commands. It is set when a command is placed in pause state, and is reset when it is taken out of the pause state. By monitoring the set/reset status of this flag, you can see if the execution of a motion control instruction is currently in pause state.



(3) Error flag

The Error flag is provided for the purpose of checking and managing the success or failure of commands. It is reset (= 0) when the execution of a command is terminated normally, and is set (= 1) when the execution of a command is terminated abnormally. By monitoring the set/reset status of this flag, you can see if the execution of a motion control instruction is done error-free.



(4) Command errors

Command errors are recorded in a user-specified storage area in which error codes generated on HI-FLOW side are stored. When the Error flag is set (= 1), you can use that storage area in order to identify the error detected on HI-FLOW side. (For information on any error detected on motion side, refer to the alarm or warning provided as part of the axis status.)

To report on detected command errors, the corresponding error codes as listed below are stored in the storage area.

(1/2)

Error code	Error type	Description	Required user action
0x2002	Parameter error	An error was detected in a specified command parameter.	Check the specified parameters to the relevant command.
0x2003	Axis alarm being issued	An alarm is being generated concerning the axis for which a command has been issued.	Issue an alarm clear command to clear the alarm condition.
0x2004	Inappropriate motion command being executed	When a servo-ON command was issued, the MP2300H controller's motion command being executed was other than the NOP command.	Issue an alarm clear command and change the motion command to NOP.
0x2005	Alarm-clear in progress	The alarm-clear process is currently being executed, so no command can be issued for execution.	Wait for the alarm-clear process to be completed. When it is completed, issue a command.
0x2006	Axis already in use	A next command was issued for an axis that was already executing a command not allowing overwriting by another command.	Wait for the command to be completed, than issue a next command.
0x2007	A command attempted in servo-OFF state	Although a command was issued for an axis, the servo of that axis was in OFF state.	Turn on the servo, then issue a command.
0x2008	Operation preparation not finished yet	Although a servo-ON command was issued, the motion module was not ready for operation.	Check if the MECHATROLINK cable is not disconnected.
0x2009	Non-connected module specified	A command was issued for an axis of a module not connected yet.	Check the axis definition to see if an attempt is made to issue a command to a module not connected yet. If there is no problem with the axis definition, check if the MECHATROLINK cable is not disconnected.
0x200A	Incorrect addition attempted	A speed-position control command is used incorrectly for either a positioning command or an external positioning command.	If the MP2300H controller's encoder type setting is incremental encoder, and a "Positioning" or "External positioning" command specifying absolute values is issued, then the "Speed-position control" command cannot be issued. Review and correct the program.
0x2100	Non-supported command received	A received command is found not supported in the HI-FLOW system.	The communication task may have been corrupted. Load it in again.

(2/2)

Error code	Error type	Description	Required user action
0x2101	Communication timeout detected	A timeout condition is detected in the MP2300H controller.	If this problem persists even after the communication has been retried, power down the MP2300H controller and then power it up again.
0x2102	Communication preparation not finished yet	The MP2300H controller is not ready for communication.	Wait for the MP2300H controller to become ready for communication. When it becomes ready, issue a command.
0x2103	Command size too small	The number of words actually transmitted as a communication header was found smaller than expected.	The communication task may have been corrupted. Load it in again.
0x2104	Command size too large	The number of words actually transmitted as a communication header was found larger than expected.	The communication task may have been corrupted. Load it in again.
0x2105	Non-matching command entry size detected	The command entry actually transmitted as communication data was found not matching the expected one.	
0x4001	Attempt made to do more communication retries than permitted	No response from the motion module was detected during command transmission.	Check if the MP2300H controller is up and running, and if its connecting cable is not disconnected.
0x8001	ET.NET abnormality	An error was detected by the ET.NET handler.	The ET.NET may have stopped due to an error. Replace the ET.NET.
0x8100	Timeout period setting error on motion side	The set value of the timeout period on motion side was found destroyed.	By using your tool, correct the set value of “Timeout of motion module” in the “Set communication” tab pane of the “Motion basic setup” window.

(5) Allocation of axial status

User-specified storage areas other than the above are provided for storing motion status information. You can use any of those storage areas when obtaining error information or process status information for the corresponding axis used.

One axis-status storage area is allocated for each of the axes used, as shown below.

Relative address (word)

00	Management information
01	(Reserved for future expansion)
02	Operation status
03	Motion command response code
04	Motion command status
05	Position management status
06 07	Warning
08 09	Alarm
0A	Axis status
0B 0C 0D 0E 0F	(Reserved for future expansion)

■ Management information

Management information indicates whether the reported operation status of motion for the axis is significant or not, and is provided in the following bit:

Bit	Operation status significance indication
0	Insignificance flag (set to 1 when the reported operation status is insignificant)
1 to 15	Unused

■ Operation status

The operation status of motion for the axis is reported by using the following bits:

Bit	Operation status reported
0	Motion control operation ready
1	Running (servo ON)
2	System busy
3	Servo ready
4 to 15	Unused

■ Motion command response code

The motion command response code generated is the command number of the command currently under execution and is one of the following:

Bit	Command type
0	NOP
1	Positioning
2	External positioning
3	Home position return
4	Linear interpolation and constant-speed control
7	Constant-speed feed
23	Speed control and speed-position control
24	Torque control

■ Motion command status

The execution status of the motion command is reported by using the following bits:

Bit	Motion command status reported
0	Command-executing flag (BUSY)
1	Command hold completed (HOLD)
2	Unused
3	Command error occurrence (FAIL)
4 to 6	Unused
7	“Reset absolute encoder” completed
8	Command executing completed (COMPLETE)
9 to 15	Unused

■ Position management status

Bit	Command information
0	Distribution completed
1	Positioning completed
2	Latch completed
3	Position proximity
4	Zero point position
5	Zero point return completed
6	Machine lock
7	Unused
8	ABS system infinite length position control info LOAD completed
9	POSMAX turn number presetting completed
10 to 15	Unused

■ Warning

The warning issued for the axis is reported by using the following bits:

Bit	Warning issued
0	Excessively following error
1	Setting parameter error
2	Fixed parameter error
3	Servo driver error
4	Motion command setting error
5	Unused
6	Positive overtravel
7	Negative overtravel
8	Servo-ON not completed yet
9	Servo driver communication warning
10 to 31	Unused

■ Alarm

The alarm issued for the axis is reported by using the following bits:

Bit	Alarm issued
0	Servo driver error
1	Positive overtravel
2	Negative overtravel
3	Positive software limit
4	Negative software limit
5	Servo OFF
6	Positioning time over
7	Excessive positioning moving amount
8	Excessive speed
9	Excessively following error
10	Filter type change error
11	Filter time constant change error
12	Servo driver command timeout error
13	Zero point not set
14 to 15	(Reserved for system use)
16	Servo driver synchronization communication error
17	Servo driver communication error
18	Servo driver command timeout error
19	ABS encoder count exceeded
20	PG disconnected error
21 to 29	Unused
30	SERVOPACK motor type mismatch
31	SERVOPACK encoder type mismatch

■ Axis status

The operation status of motion for the axis is reported by using the following bits:

Bit	Operation status reported
0	(Reserved for system use)
1	Alarm clear currently in progress
2	Axis specification error
3 to 15	Unused

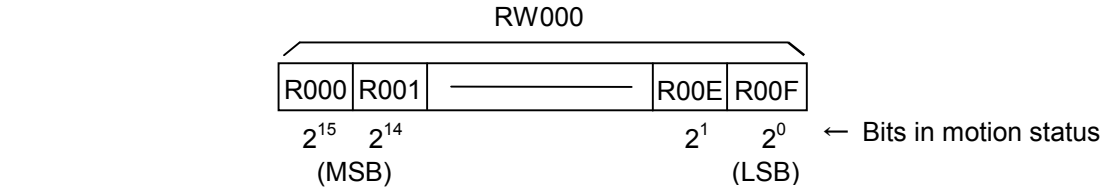
Bit 2 (axis specification error) in the above table is set to 0 when the axis is found installed, and is set to 1 when it is found not installed (not connected).

■ Referencing bits of the motion statuses in the S10V controller

The information below shows how to reference bits of the various motion statuses that are mapped to the S10V controller's register(s) by motion basics setting.

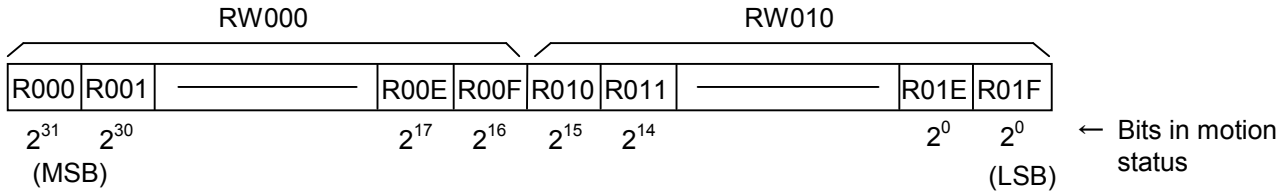
• Word-type statuses

The word-type statuses (i.e., the operation statuses, motion command status, and position management status) are in the correspondence shown below with the S10V controller's register. The figure below shows an example where a word-type status is mapped to the register RW000. In this case, each bit of the motion status has the following one-to-one correspondence with each bit of the register: Bit0 = R00F, Bit1 = R00E, ..., Bit14 = R001, Bit15 = R000.



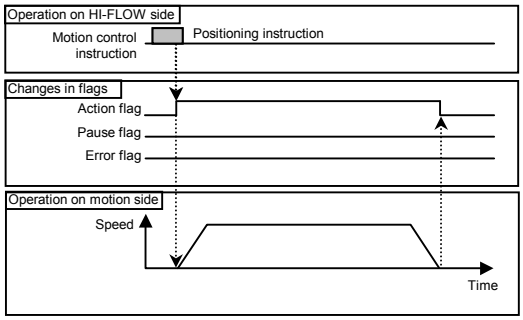
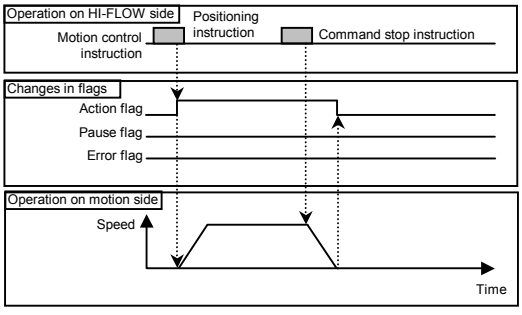
• Long word-type statuses

The long word-type statuses (i.e., the warning and alarm) are in the correspondence shown below with the S10V controller's registers. The figure below shows an example where a long word-type status is mapped to the registers RW000 and RW010. In this case, each bit of the motion status has the following one-to-one correspondence with each XXX of the registers: Bit0 = R01F, Bit1 = R01E, ..., Bit14 = R011, Bit15 = R010, Bit16 = R00F, Bit17 = R00E, ..., Bit30 = R001, Bit31 = R000.



6.5 Functional Descriptions

This section provides detailed descriptions of all available motion control instructions. The description of each instruction is given in the format shown below.

POS	Positioning
Function description	Positions a specified axis (or axes) at a desired target position(s) at a desired speed(s).
Parameter and operation	Ⓜ Command: Positioning Position reference type: See ① below. Axis: Is used to specify the axis number(s) of an axis (or axes) to be positioned. Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when any one or ones of the parameters ② through ⑤ below are omitted. Position reference: See ② below. Speed reference: See ③ below. Acceleration time: See ④ below. Deceleration time: See ⑤ below. ① Position reference type Is used to switch between the following two position reference methods available for positioning operations: Option: INC — Sets every target position as a relative position to the current position. Option: ABS — Sets every target position as an absolute distance from the home position. ② Position reference Is a target position to be used in positioning. The meaning of a numeric value specified as this parameter differs depending on the ABS/INC specification ① given. ③ Speed reference Is the velocity at which to perform a requested positioning operation. ④ Acceleration time Is an acceleration time constant (ms) or acceleration rate (reference unit/sec**2) required for acceleration before positioning. ⑤ Deceleration time Is a deceleration time constant (ms) or deceleration rate (reference unit/sec**2) required for deceleration after positioning.
Flag configuration	Action flag: Is set to 1 when positioning starts, and is set to 0 when it ends. This flag is also set to 0 when positioning fails. Pause flag: Is set to 1 when the execution of a hold instruction is completed, and is set to 0 when the execution of a reset hold instruction is completed. Error flag: Is set to 1 when a requested positioning operation fails. Command error: When the Error flag is set to 1, is reported by the corresponding error code stored in the user-specified storage area. (For a description of this error code, see "6.4 Motion Status Flags.") <Operation when normal>  <Operation when aborted or overwriting attempted> 

Name of motion control instruction

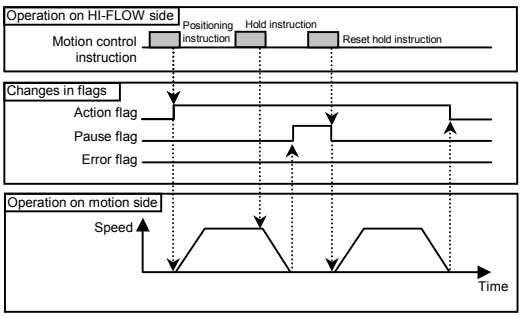
Description of what the motion control instruction does

Details of the parameters specifiable

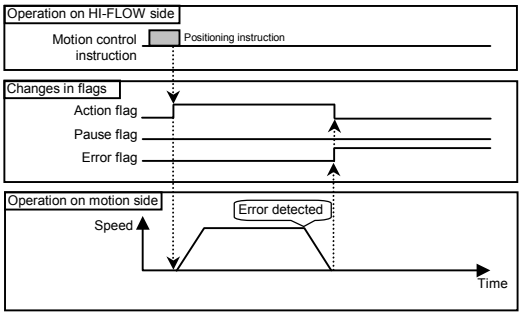
Description of changes in flags due to the execution of the instruction, besides timing charts

Flag configuration (continued from preceding page)

<Operation when paused and unpaused>



<Operation when abended>



(Continued from preceding page)

Remark

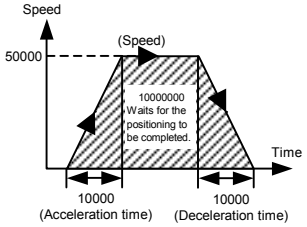
The set speed may be overridden with a new value in the range 0 to 327.67%.

Information that deserves notice

Typical use

```
M  Command: Positioning      -------  Starts positioning.
   Position reference type: INC
   Axis: Axis1
   Parameter block: PB1
   Position reference: [10000000]
   Speed reference: [500000]
   Acceleration time: [10000]
   Deceleration time: [10000]
   R000=0                    -------  Waits for the positioning to be completed
```

Usage example



Execution result of usage example

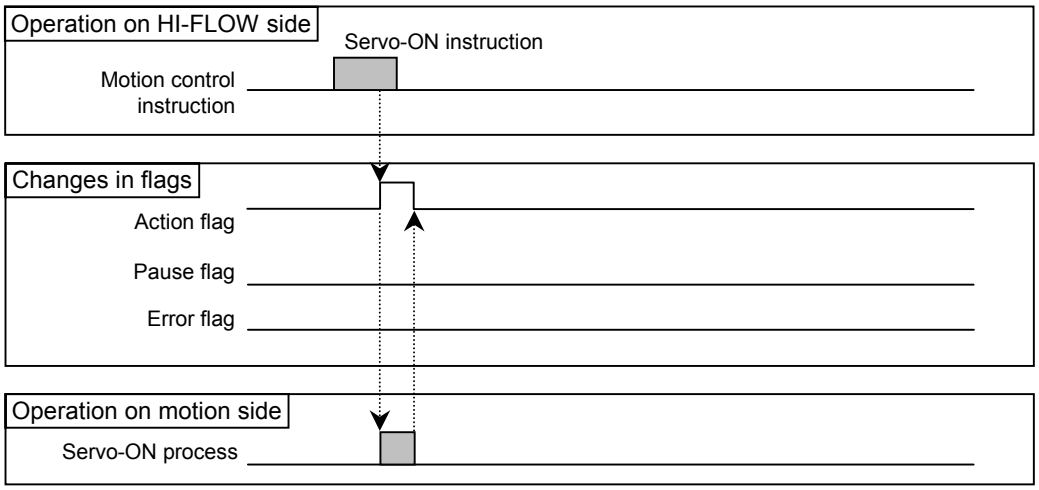
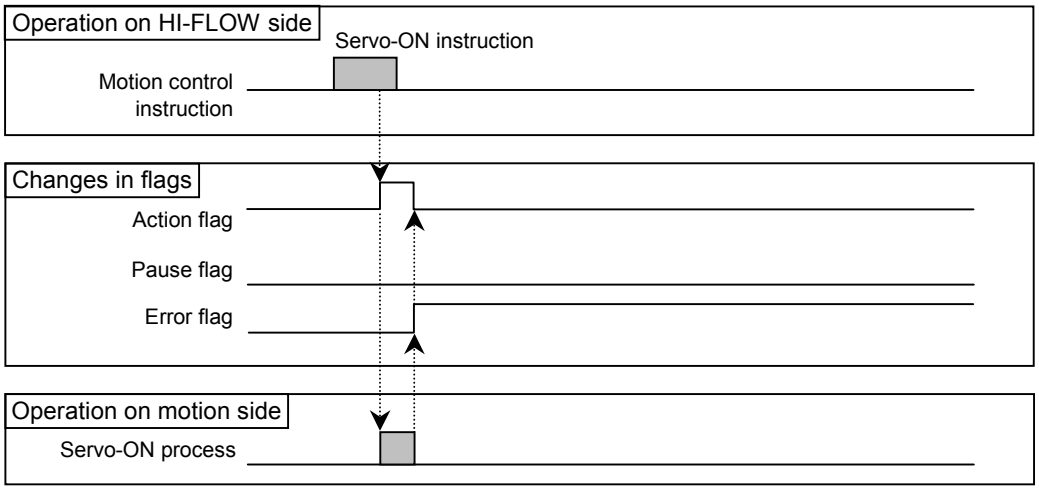
Effective parameter

No.	Parameter name	PI/O data type	Allowable range of settings
①	Position reference type	Word	0: INC 1: ABS
②	Position reference	Long word	-2147483648 to 2147483647
③	Speed reference	Long word	-2147483648 to 2147483647
④	Acceleration time	Long word	0 to 2147483647
⑤	Deceleration time	Long word	0 to 2147483647

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

List of all specifiable parameters. In motion control instructions, these parameters may be either word-type PI/O data, long word-type PI/O data, or constants. They may also be specified in a proper addressing mode.

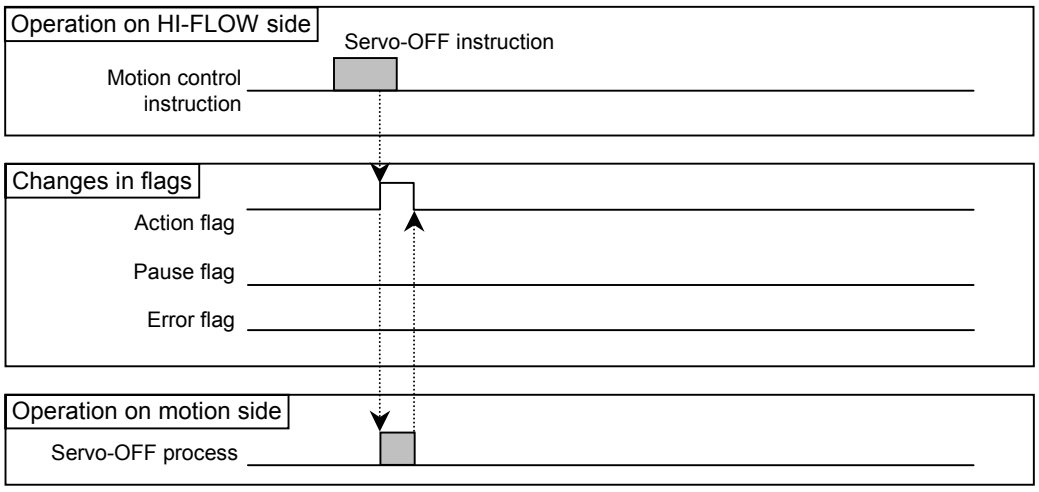
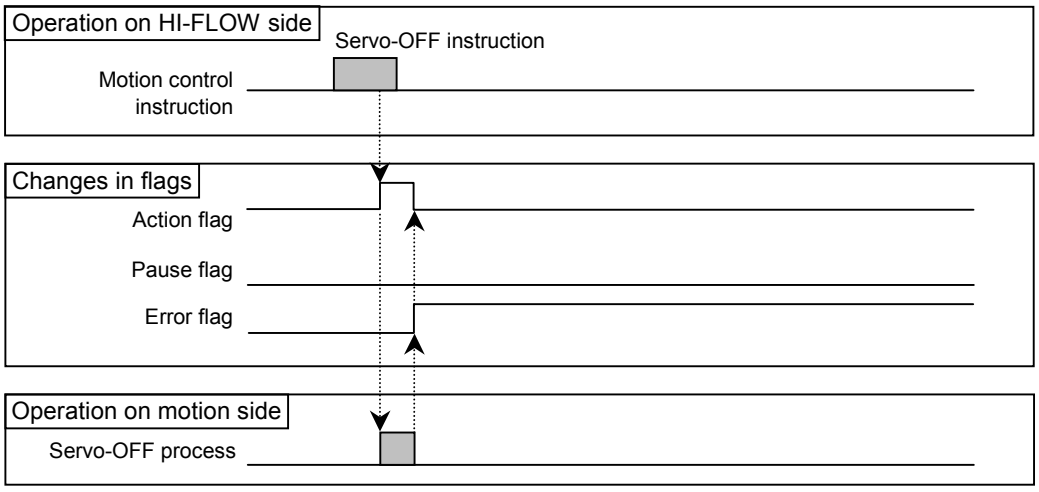
(1/2)

SVON	Servo ON
Function description	Puts a selected servo(s) into an energized state.
Parameter and operation	Ⓜ Command: Servo ON Axis: Is used to specify the axis number(s) of an axis (or axes) whose servo(s) are to be turned on. (The axis number[s] specified must be within the range 1 to 32.)
Flag configuration	Action flag: Is set to 1 when Servo-ON starts, and is set to 0 when it ends. It is also set to 0 when Servo-ON fails. Pause flag: Is always set to 0. Error flag: Is set to 1 when Servo-ON fails. <Operation when normal>  <Operation when abended> 
Remark	Servo-ON fails if used during an alarm condition. Be sure to execute an alarm clear command before the Servo-ON instruction.

(2/2)

Typical use	 M Command: Servo ON --------- Servo-ON starts. Axis: Axis1 R000=0 --------- Waits for the Servo-ON to be completed.
Effective parameter	

(1/2)

SVOFF	Servo OFF
Function description	Puts a selected servo(s) into a de-energized state.
Parameter and operation	Ⓜ Command: Servo OFF Axis: Is used to specify the axis number(s) of an axis (or axes) whose servo(s) are to be turned off. (The axis number[s] specified must be within the range 1 to 32.) <Notice> Be sure to execute the NOP instruction before this instruction so that all existing instruction information may be cleared.
Flag configuration	Action flag: Is set to 1 when Servo-OFF starts, and is set to 0 when it ends. It is also set to 0 when Servo-OFF fails. Pause flag: Is always set to 0. Error flag: Is set to 1 when Servo-OFF fails. <Operation when normal>  <Operation when abended> 
Remark	

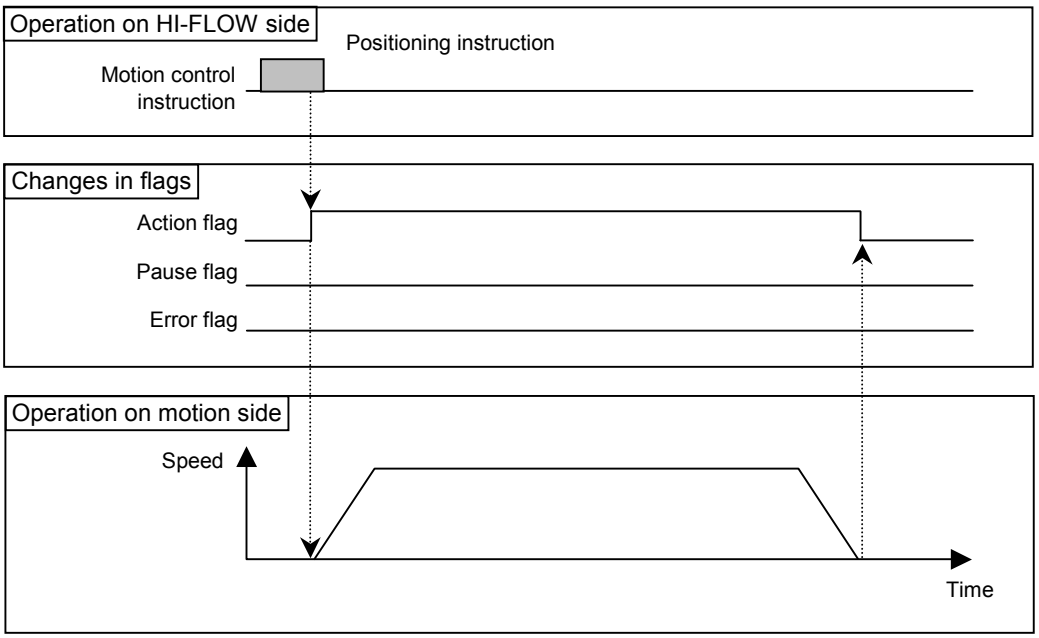
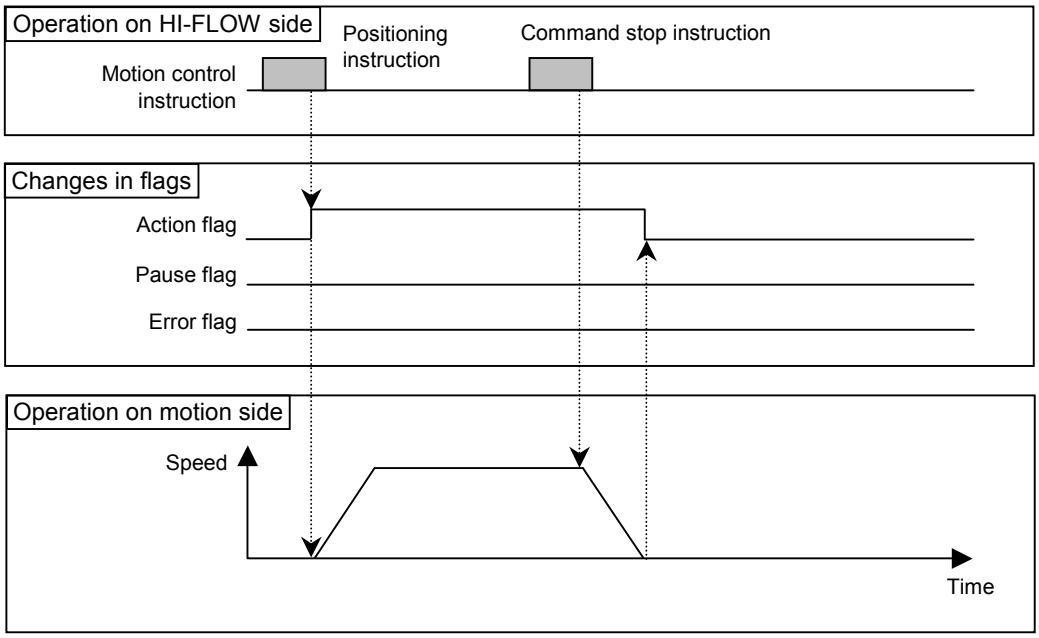
(2/2)

Typical use	 Command: NOP --------- NOP instruction executed Axis: Axis1 Command: Servo-OFF --------- Servo-OFF starts. Axis: Axis1 R000=0 --------- Waits for the Servo-OFF to be completed.
Effective parameter	

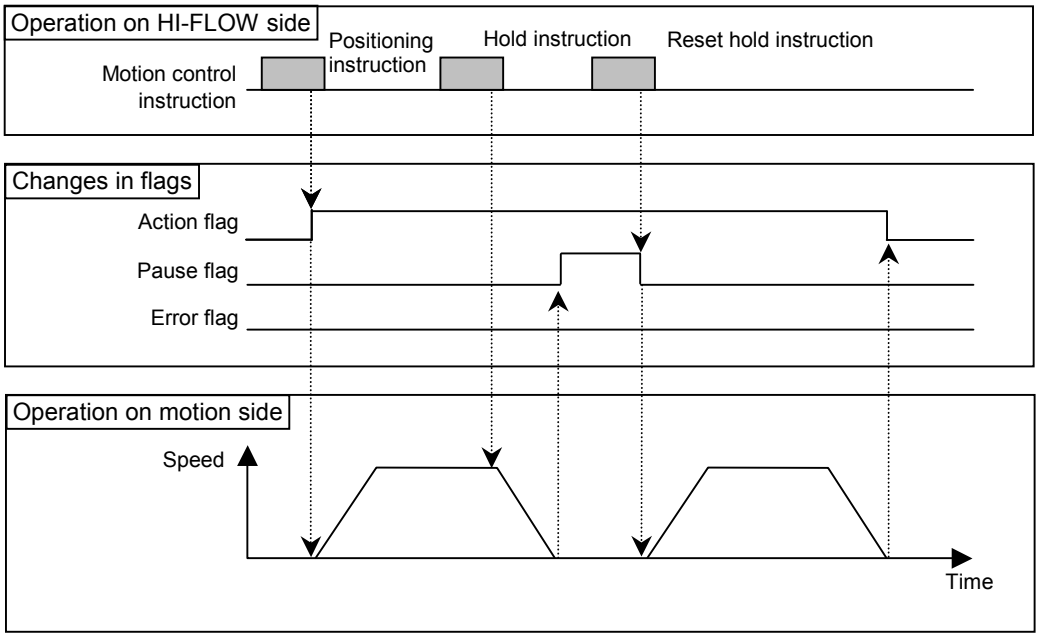
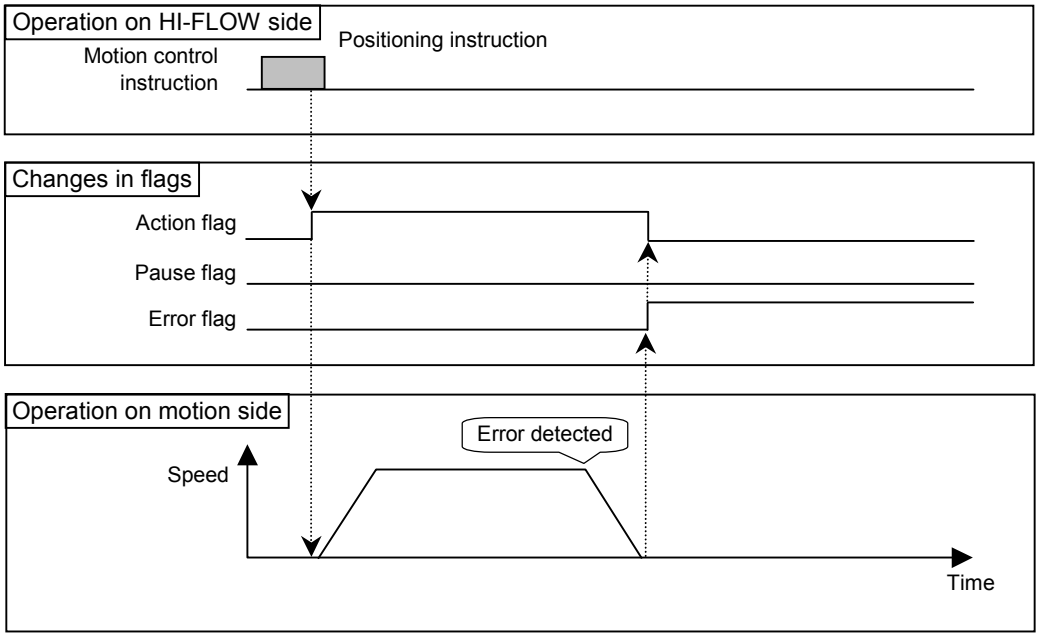
(1/4)

<table>
<tr><td>POS</td><td>Positioning</td></tr>
<tr><td>Function description</td><td>Positions a specified axis (or axes) at a desired target position(s) at a desired speed(s).</td></tr>
<tr><td>Parameter and operation</td><td>Ⓜ Command: Positioning
Position reference type: See ① below.
Axis: Is used to specify the axis number(s) of an axis (or axes) to be positioned.
Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when any one or ones of the parameters ② through ⑤ below are omitted.
Position reference: See ② below.
Speed reference: See ③ below.
Acceleration time: See ④ below.
Deceleration time: See ⑤ below.
① Position reference type
Is used to switch between the following two position reference methods available for positioning operations:<table><tr><th>Option</th><th>Description</th></tr><tr><td>INC</td><td>Sets every target position as a relative position to the current position.</td></tr><tr><td>ABS</td><td>Sets every target position as an absolute distance from the home position.</td></tr></table>② Position reference
Is a target position to be used in positioning. The meaning of a numeric value specified as this parameter differs depending on the ABS/INC specification ① given.
③ Speed reference
Is the velocity at which to perform a requested positioning operation.
④ Acceleration time
Is an acceleration time constant (ms) or acceleration rate (reference unit/sec**2) required for acceleration before positioning.
⑤ Deceleration time
Is a deceleration time constant (ms) or deceleration rate (reference unit/sec**2) required for deceleration after positioning.</td></tr>
<tr><td>Flag configuration</td><td>Action flag: Is set to 1 when positioning starts, and is set to 0 when it ends. This flag is also set to 0 when positioning fails.
Pause flag: Is set to 1 when the execution of a hold instruction is completed, and is set to 0 when the execution of a reset hold instruction is completed.
Error flag: Is set to 1 when a requested positioning operation fails.
Command error: When the Error flag is set to 1, is reported by the corresponding error code stored in the user-specified storage area. (For a description of this error code, see “6.4 Motion Status Flags.”)</td></tr>
</table>

(2/4)

Flag configuration (continued from preceding page)	<Operation when normal>  <Operation when aborted or overwriting attempted> 

(3/4)

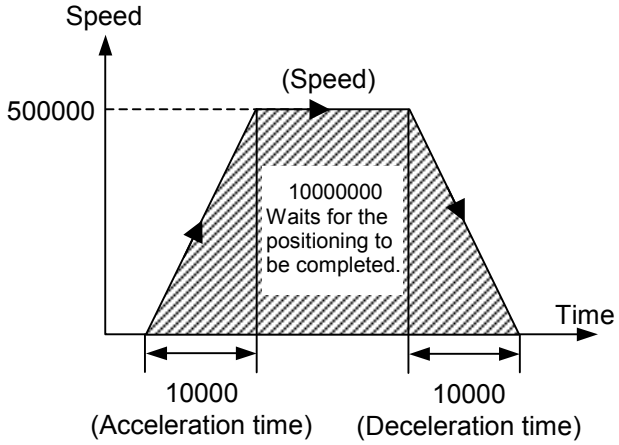
Flag configuration (continued from preceding page)	<Operation when paused and unpaused>  <Operation when abended> 
Remark	The set speed may be overridden with a new value in the range 0 to 327.67%.

(4/4)

Typical use

Command: Positioning --------- Starts positioning.
Position reference type: INC
Axis: Axis1
Parameter block: PB1
Position reference: [10000000]
Speed reference: [500000]
Acceleration time: [10000]
Deceleration time: [10000]

R000=0 --------- Waits for the positioning to be completed



Effective parameter

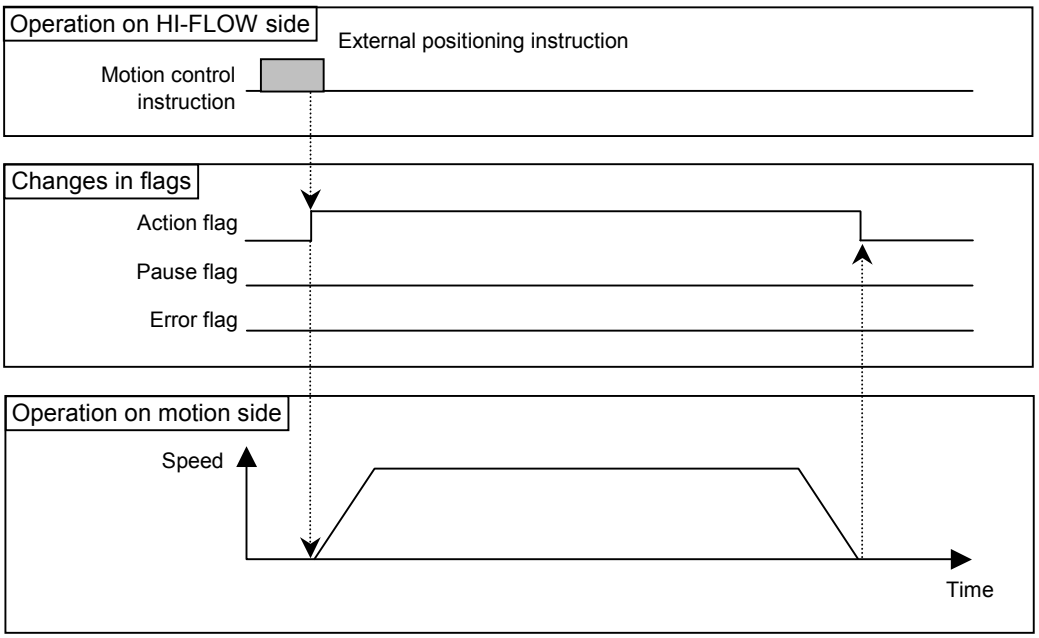
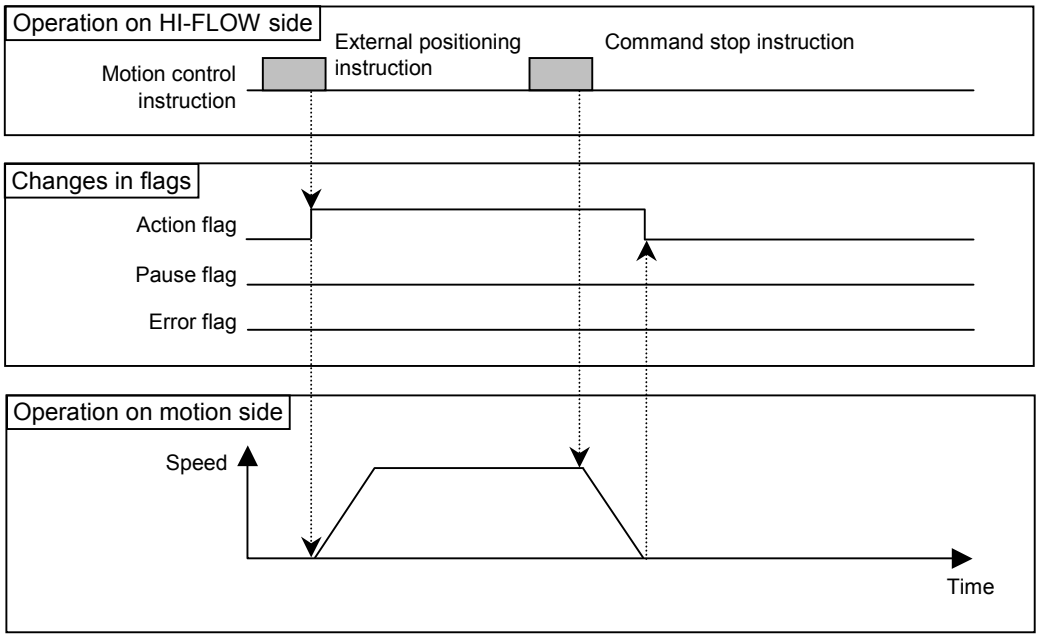
No.	Parameter name	PI/O data type	Allowable range of settings
①	Position reference type	Word	0: INC 1: ABS
②	Position reference	Long word	-2147483648 to 2147483647
③	Speed reference	Long word	-2147483648 to 2147483647
④	Acceleration time	Long word	0 to 2147483647
⑤	Deceleration time	Long word	0 to 2147483647

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

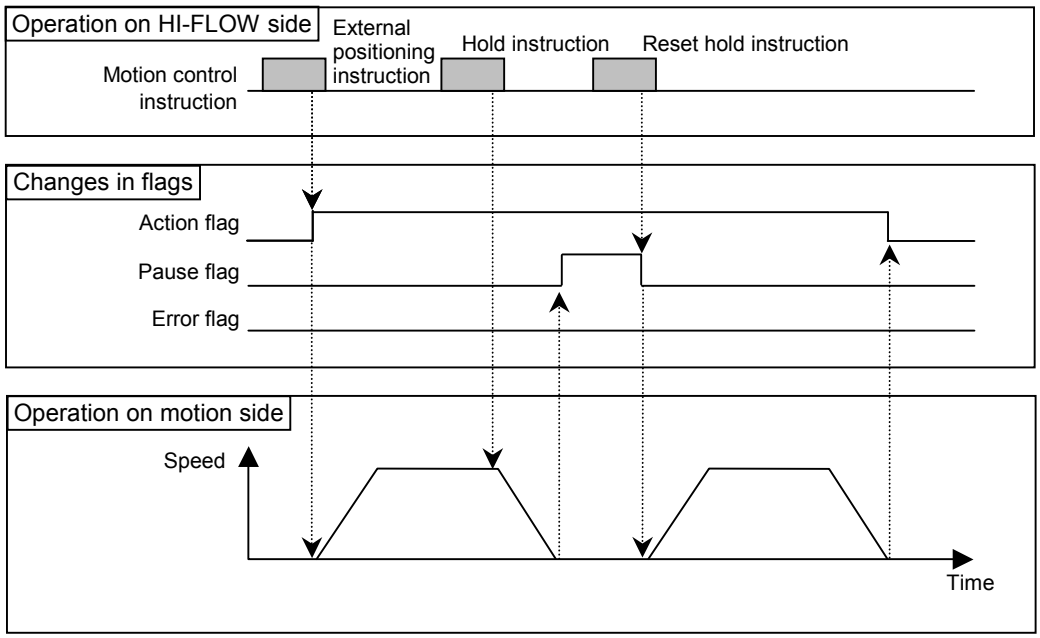
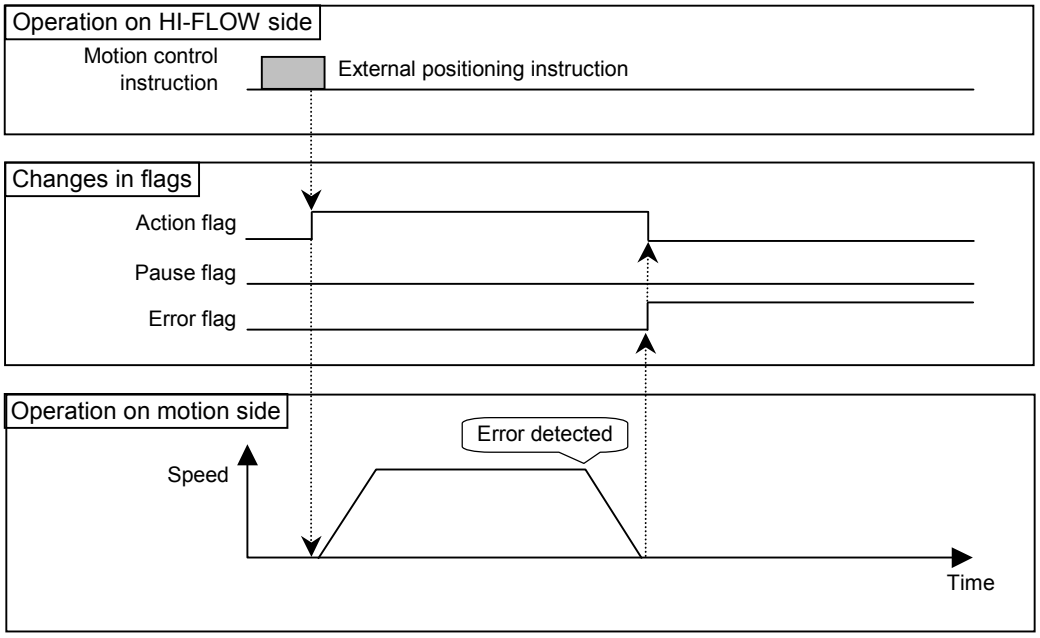
(1/4)

<table>
<tr><td>EXPOS</td><td>External positioning</td></tr>
<tr><td>Function description</td><td>Positions a specified axis (or axes) at a desired target position(s) at a desired speed(s) if the external positioning signal is not turned on during the move(s). If, however, it is turned on during the move(s), this instruction moves the axis (or axes) a specified distance(s) and positions them there.</td></tr>
<tr><td>Parameter and operation</td><td>Ⓜ Command: External positioning
Position reference type: See ① below.
Axis: Is used to specify the axis number(s) of an axis (or axes) to be positioned externally.
Parameter block: Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when any one or ones of the parameters ② through ⑥ are omitted.
Position reference: See ② below.
Speed reference: See ③ below.
Acceleration time: See ④ below.
Deceleration time: See ⑤ below.
External positioning: See ⑥ below.
① Position reference type
Is used to switch between the following two position reference methods available for positioning operations:
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>INC</td><td>Sets every target position as a relative position to the current position.</td></tr>
<tr><td>ABS</td><td>Sets every target position as an absolute distance from the home position.</td></tr>
</table>
② Position reference
Is a target position to be used in positioning. The meaning of a numeric value specified as this parameter differs depending on the ABS/INC specification ① given.
③ Speed reference
Is the velocity at which to perform a requested positioning operation.
④ Acceleration time
Is an acceleration time constant (ms) or acceleration rate (reference unit/sec**2) required for acceleration before positioning.
⑤ Deceleration time
Is a deceleration time constant (ms) or deceleration rate (reference unit/sec**2) required for deceleration after positioning.
⑥ External positioning move distance
Is an amount of movement of the axis that is to be made upon the input of the external positioning signal.
<Notice>
Be sure to execute the NOP instruction following the completion of the external positioning.</td></tr>
<tr><td>Flag configuration</td><td>Action flag: Is set to 1 when external positioning starts, and is set to 0 when it ends. This flag is also set to 0 when external positioning fails.
Pause flag: Is set to 1 when the execution of a hold instruction is completed, and is set to 0 when the execution of a reset hold instruction is completed.
Error flag: Is set to 1 when a requested external positioning operation fails.</td></tr>
</table>

(2/4)

Flag configuration (continued from preceding page)	<Operation when normal>  <Operation when aborted or overwriting attempted> 

(3/4)

Flag configuration (continued from preceding page)	<Operation when paused and unpaused>  <Operation when abended> 
Remark	The set speed may be overridden with a new value in the range 0 to 327.67%.

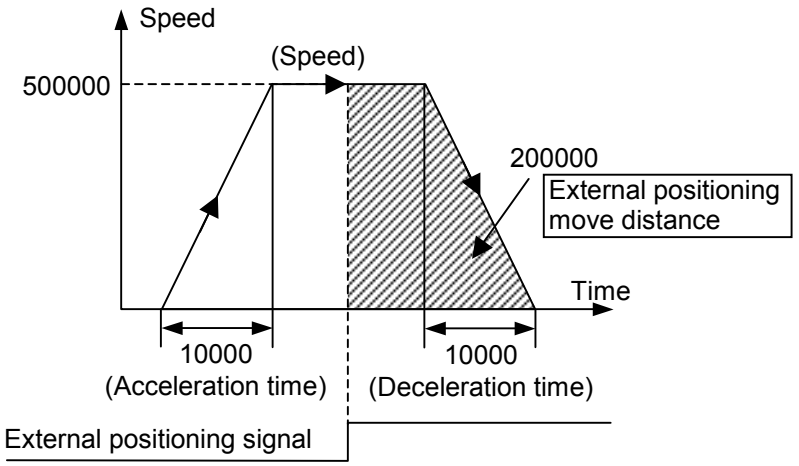
(4/4)

Typical use

Command: External positioning --------- Starts positioning.
Position reference type: INC
Axis: Axis1
Parameter block: PB1
Position reference: [10000000]
Speed reference: [500000]
Acceleration time: [10000]
Deceleration time: [10000]
External positioning move distance: [200000]

R000=0 --------- Waits for the positioning to be completed

Command: NOP --------- NOP instruction executed
Axis: Axis1



Effective parameter

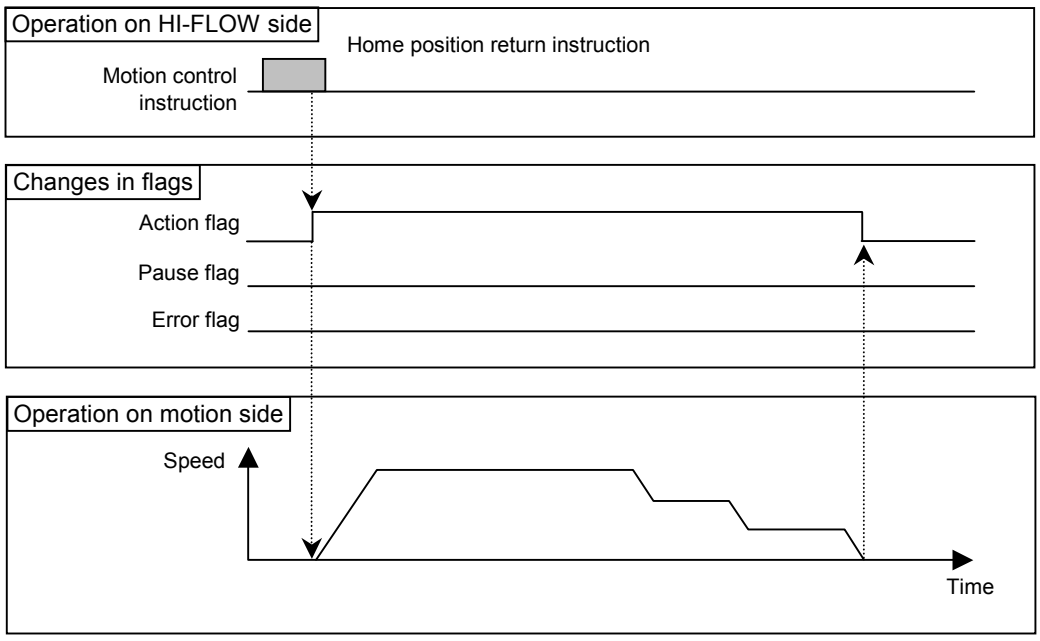
No.	Parameter name	PI/O data type	Allowable range of settings
①	Position reference type	Word	0: INC 1: ABS
②	Position reference	Long word	-2147483648 to 2147483647
③	Speed reference	Long word	-2147483648 to 2147483647
④	Acceleration time	Long word	0 to 2147483647
⑤	Deceleration time	Long word	0 to 2147483647
⑥	External positioning move distance	Long word	-2147483648 to 2147483647

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

(1/5)

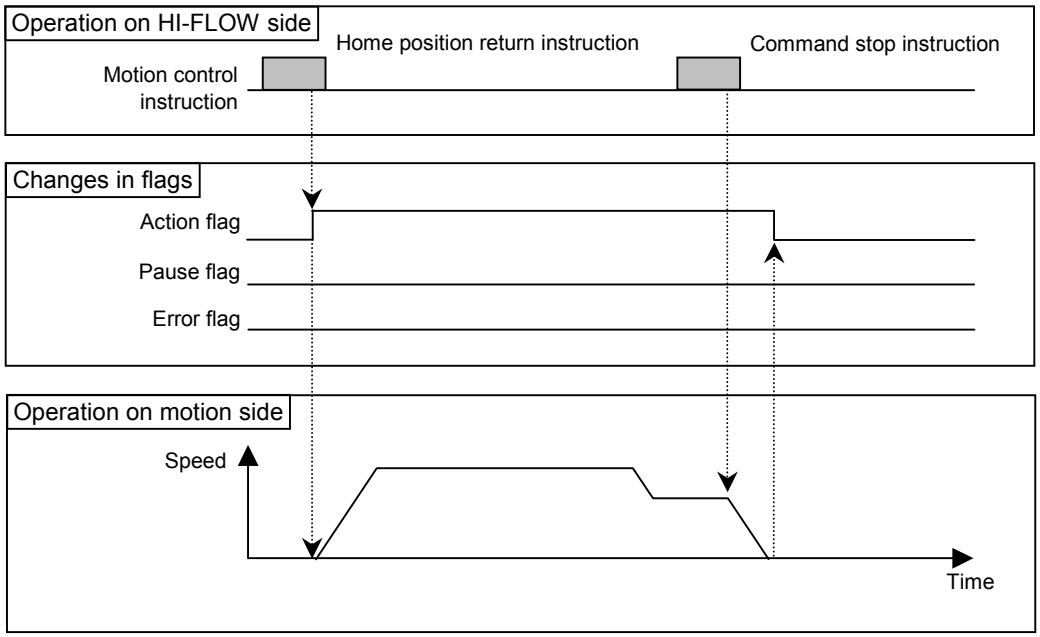
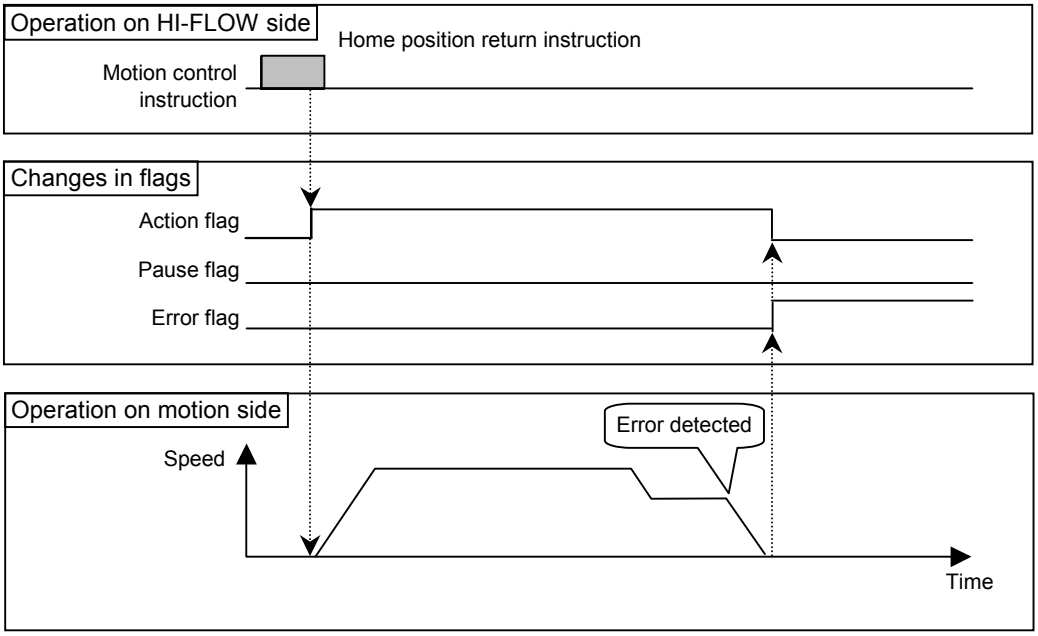
<table>
<tr><td>ZRET</td><td>Home position return</td></tr>
<tr><td>Function description</td><td>Causes the system to return to the origin(s) of a machine coordinate system(s).</td></tr>
<tr><td>Parameter and operation</td><td>
Ⓜ Command: Home position return
Home return type: See ① below.
Axis: Is used to specify the axis number(s) of an axis (or axes) to be homed.
Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when any one or ones of the parameters ② through ⑧ are omitted.
Speed reference: See ② below.
Acceleration time: See ③ below.
Deceleration time: See ④ below.
Home direction: See ⑤ below.
Approach speed: See ⑥ below.
Creep speed: See ⑦ below.
Home offset: See ⑧ below.
① Home return type
Switches between the following 17 home (zero-point) return types available for home return operations:
<table>
<tr><th>Type</th><th>Description</th></tr>
<tr><td>DEC1 + C-Phase</td><td>A 3-step deceleration method using deceleration LS and Phase-C pulse signal</td></tr>
<tr><td>Zero Signal</td><td>A home return method using ZERO signal</td></tr>
<tr><td>DEC1 + Zero Signal</td><td>A 3-step deceleration method using deceleration LS and ZERO signal</td></tr>
<tr><td>C-Phase</td><td>A home return method using the C-phase pulse signal</td></tr>
<tr><td>DEC2 + Zero Signal Method</td><td>A home return method using deceleration LS as the area signal and ZERO signal as the zero-point signal</td></tr>
<tr><td>DEC1 + LMT + Zero Signal Method</td><td>A home return method using deceleration LS and two limit (LMT) signals for zero-point return as the area signal and ZERO signal as the zero-point signal</td></tr>
<tr><td>DEC2 + Phase-C Signal Method</td><td>A home return method using deceleration LS as the area signal and Phase-C signal as the zero-point signal</td></tr>
<tr><td>DEC1 + LMT + Phase-C Signal Method</td><td>A home return method using deceleration LS and two limit (LMT) signals for zero-point return as the area signal and Phase-C signal as the zero-point signal</td></tr>
<tr><td>C pulse Only</td><td>A home return method using only Phase-C pulse signal</td></tr>
<tr><td>POT & C pulse</td><td>A home return method using the positive OT signal and Phase-C pulse signal</td></tr>
<tr><td>POT Only</td><td>A home return method using only the positive OT signal</td></tr>
<tr><td>HOME LS & C pulse</td><td>A home return method using the HOME signal and Phase-C pulse signal</td></tr>
<tr><td>HOME Only</td><td>A home return method using only the HOME signal</td></tr>
<tr><td>NOT & C pulse</td><td>A home return method using the negative OT signal and Phase-C pulse signal</td></tr>
<tr><td>NOT Only</td><td>A home return method using only the negative OT signal</td></tr>
<tr><td>INPUT & C pulse</td><td>A home return method using input signals and Phase-C pulse signal</td></tr>
<tr><td>INPUT Only</td><td>A home return method using only input signals</td></tr>
</table>
</td></tr>
</table>

(2/5)

<table>
<tr><td>Parameter and operation (continued from preceding page)</td><td>
② Speed reference
Is the velocity at which to perform a requested home return operation.
③ Acceleration time
Is an acceleration time constant (ms) or acceleration rate (reference unit/sec**2) required for acceleration before homing.
④ Deceleration time
Is a deceleration time constant (ms) or deceleration rate (reference unit/sec**2) required for deceleration during homing.
⑤ Home direction
Is a direction of movement of the axis during homing. One of the following two directions of axis movement is selectable for homing.
<table>
<tr><th>Option</th><th>Remarks</th></tr>
<tr><td>Reverse rotation</td><td>Default</td></tr>
<tr><td>Forward rotation</td><td></td></tr>
</table>
⑥ Approach speed
Is a velocity at which the axis will move following the closing of the deceleration LS.
⑦ Creep speed
Is a velocity at which the axis will move to home following the detection of the zero-point signal.
⑧ Home offset
Is an amount of axis movement to be made starting at the rising edge of the zero-point signal and ending at home position.
<Notice>
Be sure to execute the NOP instruction following the completion of the homing.
</td></tr>
<tr><td>Flag configuration</td><td>
Action flag: Is set to 1 when homing starts, and is set to 0 when it ends. This flag is also set to 0 when homing fails.
Pause flag: Is always set to 0.
Error flag: Is set to 1 when a requested homing operation fails.
<Operation when normal>


</td></tr>
</table>

(3/5)

Flag configuration (continued from preceding page)	<Operation when aborted or overwriting attempted>  <Operation when abended> 
Remarks	

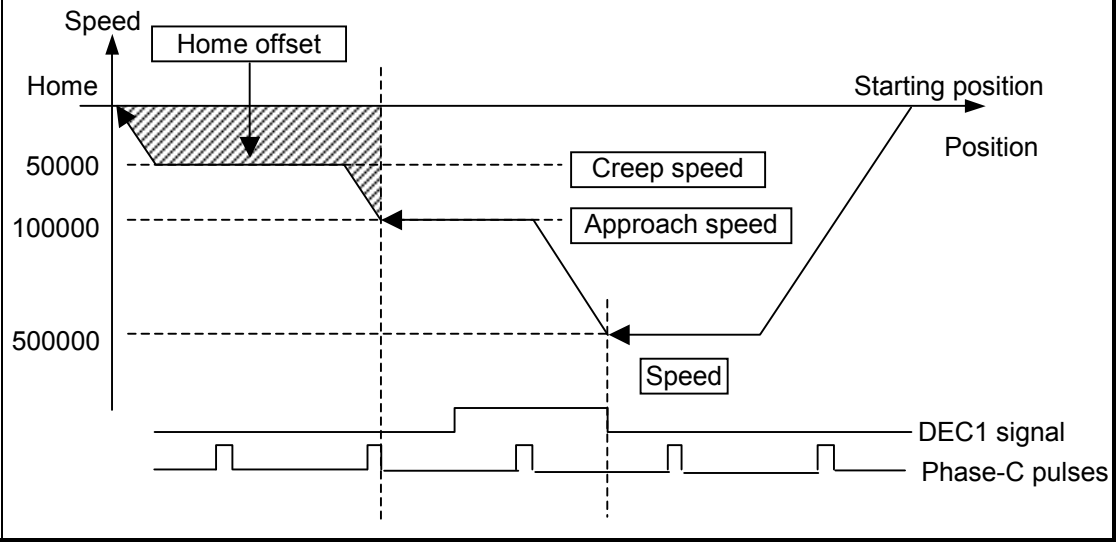
(4/5)

Typical use

M Command: Home position return ---------- Starts homing.
Home return type: DEC1 + C-Phase
Axis: Axis1
Parameter block: PB1
Speed reference: [500000]
Acceleration time: [10000]
Deceleration time: [10000]
Home direction: Reverse rotation
Approach speed: [100000]
Creep speed: [50000]
Home offset: [20000]

R000=0 ---------- Waits for the homing to be completed.

M Command: NOP ---------- NOP instruction executed
Axis: Axis1



(5/5)

Effective parameter

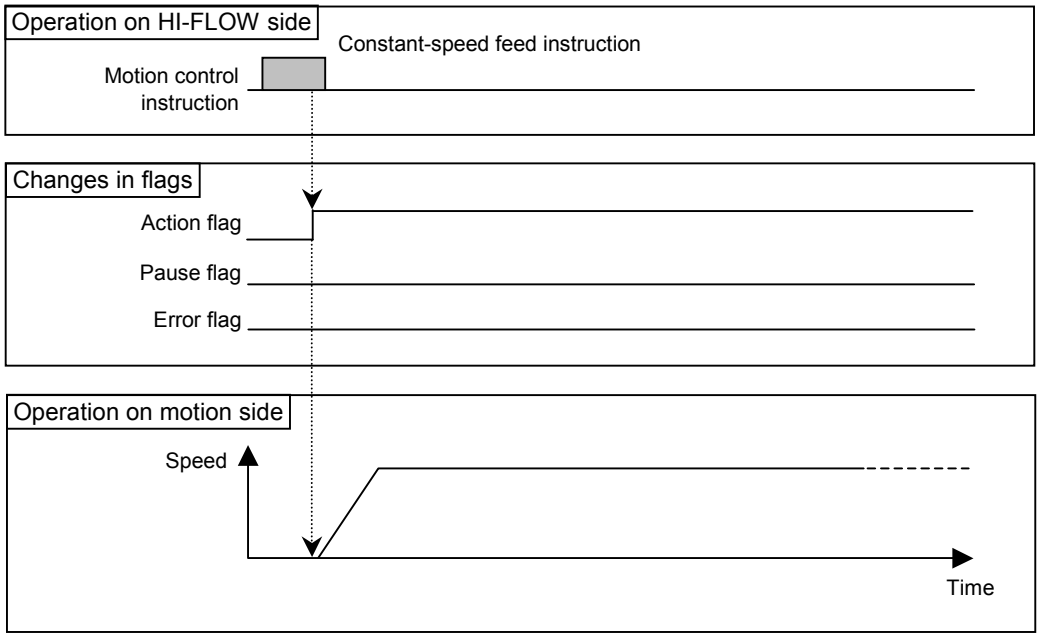
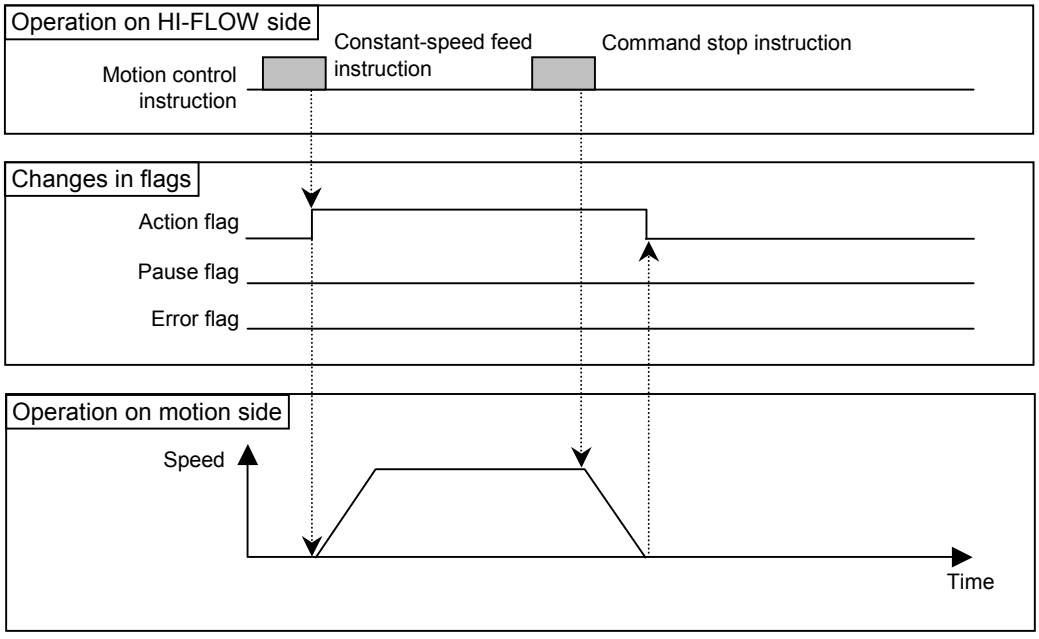
No.	Parameter name	PI/O data type	Allowable range of settings
①	Home return type	Word	0: DEC1 + C-Phase 1: Zero Signal 2: DEC1 + Zero Signal 3: C-Phase 4: DEC2 + Zero Signal Method 5: DEC1 + LMT + Zero Signal Method 6: DEC2 + Phase-C Signal Method 7: DEC1 + LMT + Phase-C Signal Method 11: C pulse Only 12: POT & C pulse 13: POT Only 14: HOME LS & C pulse 15: HOME LS Only 16: NOT & C pulse 17: NOT Only 18: INPUT & C pulse 19: INPUT Only
②	Speed reference	Long word	-2147483648 to 2147483647
③	Acceleration time	Long word	0 to 2147483647
④	Deceleration time	Long word	0 to 2147483647
⑤	Home direction	Word	0: Reverse rotation 1: Forward rotation
⑥	Approach speed	Long word	-2147483648 to 2147483647
⑦	Creep speed	Long word	-2147483648 to 2147483647
⑧	Home offset	Long word	0 to 2147483647

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

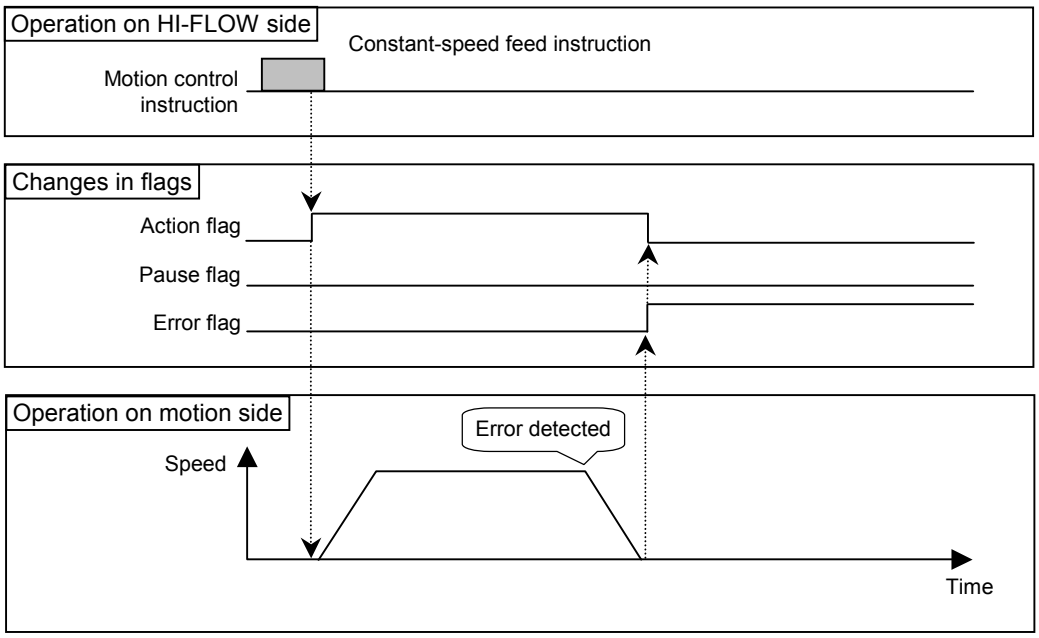
(1/4)

<table>
<tr><td>FEED</td><td>Constant-speed feed</td></tr>
<tr><td>Function description</td><td>Moves a specified axis (or axes) in a desired direction(s) at a desired speed(s). The requested movement will continue until the command stop instruction is executed.</td></tr>
<tr><td>Parameter and operation</td><td>Command: Constant-speed feed
Axis: Is used to specify the axis number(s) of an axis (or axes) to be fed at a specified constant speed(s).
Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when any one or ones of the parameters ① through ④ below are omitted.
Speed reference: See ① below.
Acceleration time: See ② below.
Deceleration time: See ③ below.
Direction: See ④ below.
① Speed reference
Is the velocity at which to perform a requested constant-speed feed operation.
② Acceleration time
Is an acceleration time constant (ms) or acceleration rate (reference unit/sec**2) required for acceleration before constant-speed feed.
③ Deceleration time
Is a deceleration time constant (ms) or deceleration rate (reference unit/sec**2) required for deceleration after constant-speed feed.
④ Direction
Is a direction of movement of the axis during constant-speed feed. One of the following two directions of axis movement is selectable for constant-speed feeds.
<table><tr><th>Option</th><th>Remarks</th></tr><tr><td>Forward rotation</td><td>Default</td></tr><tr><td>Reverse rotation</td><td></td></tr></table></td></tr>
<tr><td>Flag configuration</td><td>Action flag: Is set to 1 when constant-speed feed starts, and is set to 0 when it ends. It is also set to 0 when constant-speed feed fails.
Pause flag: Is always set to 0.
Error flag: Is set to 1 when a requested positioning operation fails.</td></tr>
</table>

(2/4)

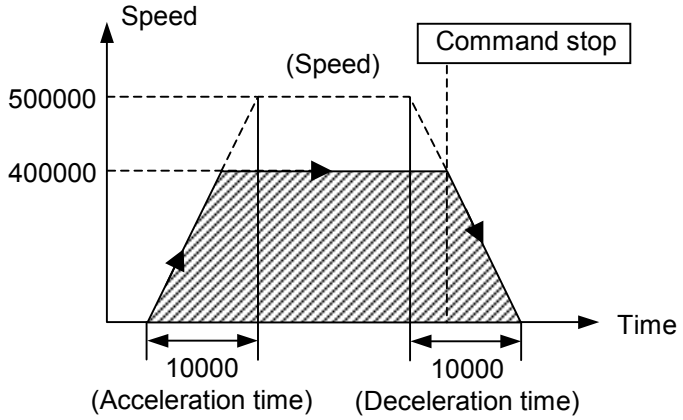
Flag configuration (continued from preceding page)	Action flag: Is set to 1 when constant-speed feed starts, and is set to 0 when it ends. It is also set to 0 when constant-speed feed fails. Pause flag: Is always set to 0. Error flag: Is set to 1 when a requested positioning operation fails. <Operation when normal>  <Operation when aborted or overwriting attempted> 

(3/4)

Flag configuration (continued from preceding page)	<Operation when abended> 
Remarks	
Typical use	Ⓜ Command: Constant-speed feed --------- Starts constant-speed feed. Axis: Axis1 Parameter block: PB1 Speed reference: [500000] Acceleration time: [10000] Deceleration time: [10000] Direction: Forward rotation WT000 (100) --------- Waits 10 seconds. Ⓜ Command: Command stop --------- Aborts the constant-speed feed. Axis: Axis1 R000=0 --------- Waits for the constant-speed feed to be completed.

(4/4)

Typical use (continued from preceding page)



Effective parameter

No.	Parameter name	PI/O data type	Allowable range of settings
①	Speed reference	Long word	-2147483648 to 2147483647
②	Acceleration time	Long word	0 to 2147483647
③	Deceleration time	Long word	0 to 2147483647
④	Direction	Word	0: Forward rotation 1: Reverse rotation

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

(1/2)

<table>
<tr><td>ABORT
HOLDS
HOLDE</td><td>Command stop
Command holding
Command reset hold</td></tr>
<tr><td>Function description</td><td>Aborts, pauses, or unpauses the command currently under execution.</td></tr>
<tr><td>Parameter and operation</td><td>(M) Command: Command stop, Command holding, or Command reset hold
Axis: Is used to specify the axis number(s) of an axis (or axes) for which the operation status of the command currently under execution is to be changed. (The axis number[s] specified must be within the range 1 to 32.)</td></tr>
<tr><td>Flag configuration</td><td>Action flag: Is set to 0 when a requested command stop operation is completed.
Pause flag: Is set to 1 when a requested command holding operation starts, and is set to 0 when a requested command reset hold operation ends.
Error flag: Is set to 1 when a requested command stop, command holding, or command reset hold operation fails.</td></tr>
<tr><td>Remarks</td><td>The Command stop, Command holding, and the Command reset hold instruction can be applied to any of the following commands:
<table>
<tr><th>Command name</th><th>Command stop</th><th>Command holding/Command reset hold</th></tr>
<tr><td>Positioning</td><td>√</td><td>√</td></tr>
<tr><td>External positioning</td><td>√</td><td>√</td></tr>
<tr><td>Home position return</td><td>√</td><td>ne</td></tr>
<tr><td>Constant-speed feed</td><td>√</td><td>ne</td></tr>
<tr><td>Torque control</td><td>√</td><td>ne</td></tr>
</table>
√: Can be applied to the command.
ne: Cannot be applied to the command.</td></tr>
</table>

(2/2)

Typical use	

Command: Positioning ----- Starts positioning (A).
Position reference type: INC
Axis: Axis1
Parameter block: PB1
Position reference: [10000000]
Speed reference: [500000]
Acceleration time: [10000]
Deceleration time: [10000]

WT000 (30) ----- Waits three seconds.

Command: Command holding ----- Starts command-holding (B).
Axis: Axis1

R020 ----- Waits for the command-holding to be completed.

WT001 (30) ----- Waits three seconds.

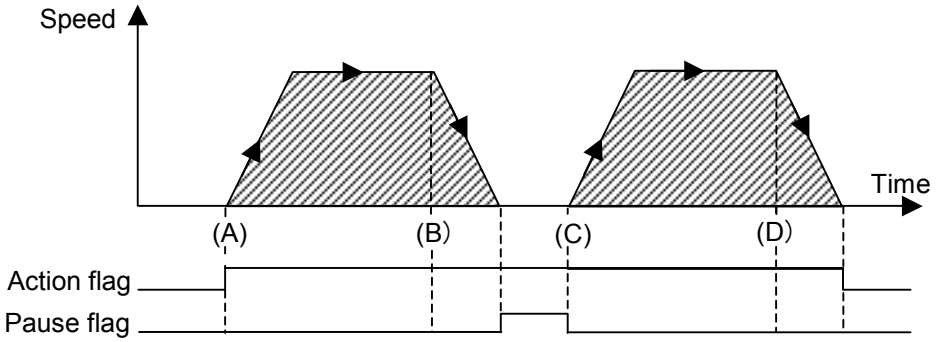
Command: Command reset hold ----- Starts command-reset-holding (C).
Axis: Axis1

R020=0 ----- Waits for the command-reset-holding to be completed.

WT002 (30) ----- Waits three seconds.

Command: Command stop ----- Starts aborting the positioning command (D).
Axis: Axis1

R000=0 ----- Waits for the positioning-command abortion to be completed.

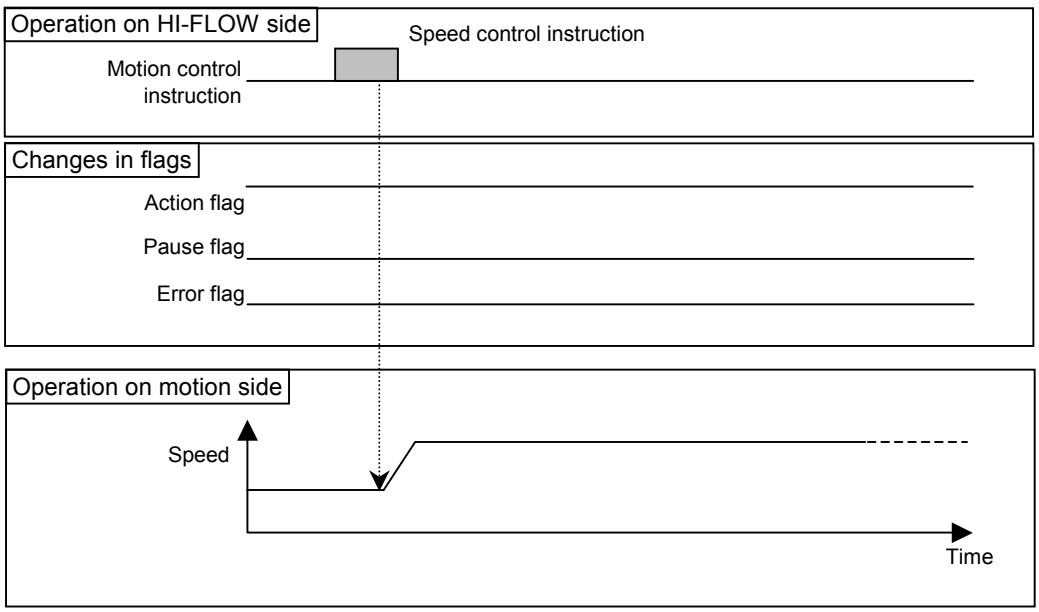


Effective parameter	

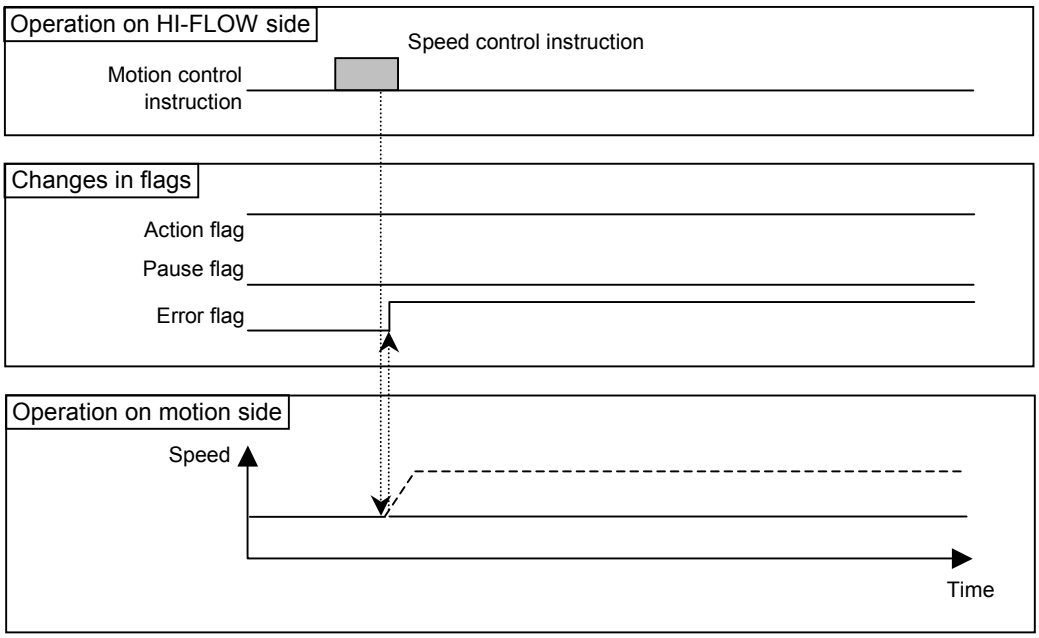
(1/3)

CHGV	Speed control
Function description	Changes the current speed(s) of movement in progress for a specified axis (or axes).
Parameter and operation	Ⓜ Command: Speed control Axis: Is used to specify the axis number(s) of an axis (or axes) for which to change the current speed(s) of movement in progress. Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when any one or ones of the parameters ① through ④ below are omitted. Speed reference: See ① below. Acceleration time: See ② below. Deceleration time: See ③ below. Direction: See ④ below. ① Speed reference Is a new velocity value to be used after changing. ② Acceleration time Is a new acceleration time constant (ms) or acceleration rate (reference unit/sec**2) to be used after changing. ③ Deceleration time Is a new deceleration time constant (ms) or deceleration rate (reference unit/sec**2) to be used after changing. ④ Direction Is a new direction setting to be used after changing. One of the following two directions of axis movement is selectable for speed control operations.
Flag configuration	Action flag: Remains unchanged after speed control operations. Pause flag: Is always set to 0. Error flag: Is set to 1 when a requested speed control operation fails. <Operation when normal>

Option	Remarks
Forward rotation	Default
Reverse rotation	



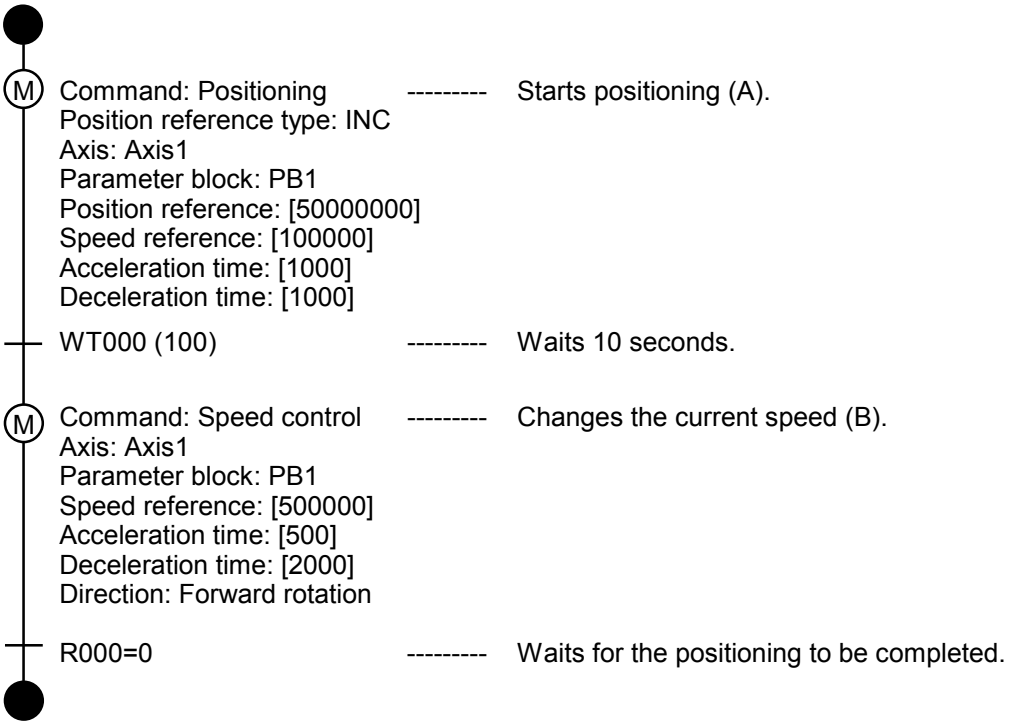
(2/3)

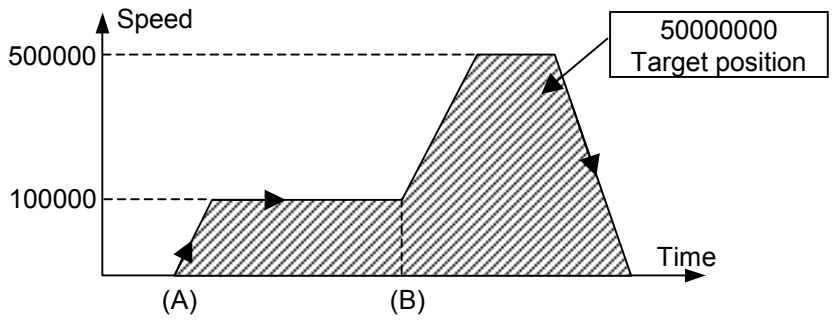
<table>
<tr>
<td>Flag configuration (continued from preceding page)</td>
<td>
<Operation when abended>


</td>
</tr>
<tr>
<td>Remarks</td>
<td>
The table below shows whether or not each speed control parameter has effect on the commands listed below.
<table>
<tr><th>Command name</th><th>Direction</th><th>Speed reference</th><th>Acceleration time</th><th>Deceleration time</th></tr>
<tr><td>Positioning</td><td>in</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>External positioning</td><td>in</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Home position return</td><td>in</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Constant-speed feed</td><td>√</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>Torque control</td><td>in</td><td>√</td><td>in</td><td>in</td></tr>
</table>
√: Has effect on the command.
in: Has no effect on the command.
</td>
</tr>
</table>

(3/3)

Typical use (continued from preceding page)



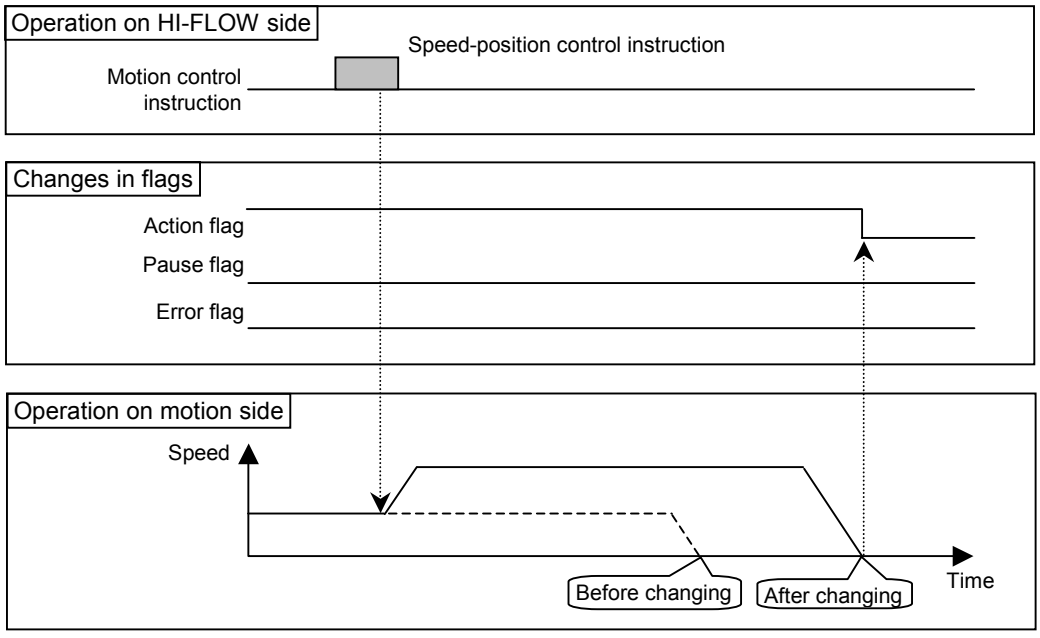


Effective parameter

No.	Parameter name	PI/O data type	Allowable range of settings
①	Speed reference	Long word	-2147483648 to 2147483647
②	Acceleration time	Long word	0 to 2147483647
③	Deceleration time	Long word	0 to 2147483647
④	Direction	Word	0: Forward rotation 1: Reverse rotation

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

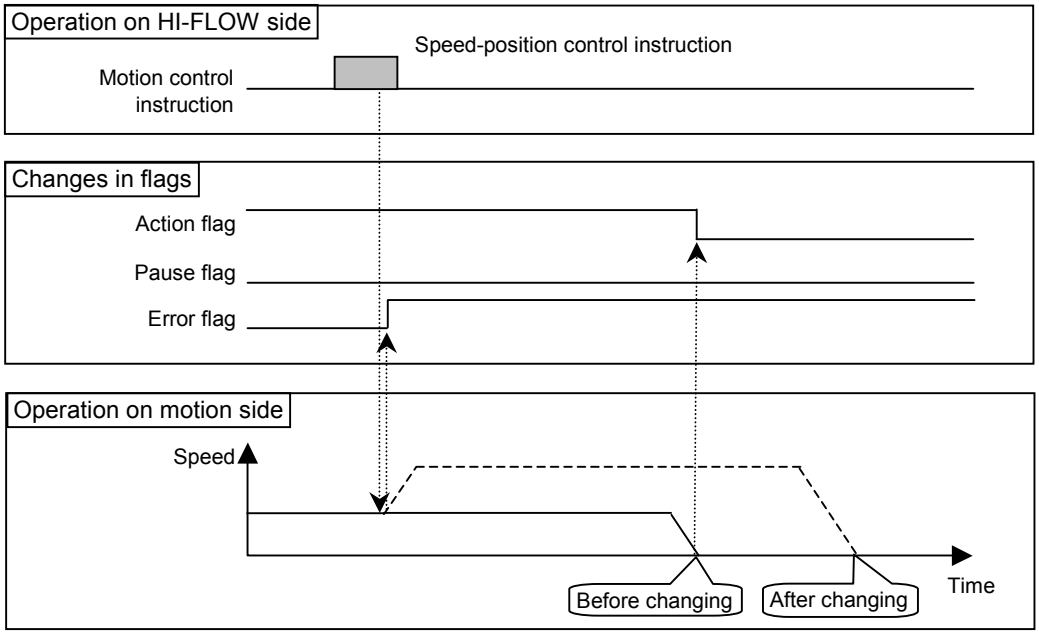
(1/3)

CHGVP	Speed-position control
Function description	Changes the current target position(s) and positioning speed(s) for a specified axis (or axes).
Parameter and operation	Command: Speed-position control Axis: Is used to specify the axis number(s) of an axis (or axes) for which to change the current target position(s) and positioning speed(s). Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when any one or ones of the parameters ① through ④ below are omitted. Position reference: See ① below. Speed reference: See ② below. Acceleration time: See ③ below. Deceleration time: See ④ below. ① Position reference Is a new target position to be used after changing. ② Speed reference Is a new velocity to be used after changing. ③ Acceleration time Is a new acceleration time constant (ms) or acceleration rate (reference unit/sec**2) to be used after changing. ④ Deceleration time Is a new deceleration time constant (ms) or deceleration rate (reference unit/sec**2) to be used after changing.
Flag configuration	Action flag: Remains unchanged after speed-position control operations. Pause flag: Is always set to 0. Error flag: Is set to 1 when a requested speed-position control operation fails. <Operation when normal> 

(2/3)

Flag configuration (continued from preceding page)

<Operation when abended>



Remarks

The table below shows whether or not each speed-position control parameter has effect on the commands listed below.

Command name	Position reference	Speed reference	Acceleration time	Deceleration time
Positioning	√	√	√	√
External positioning	in	√	√	√
Home position return	in	√	√	√
Constant-speed feed	in	√	√	√
Torque control	in	√	in	in

√: Has effect on the command.
in: Has no effect on the command.

If the MP2300H controller's encoder type setting is incremental encoder, and a "Positioning" or "External positioning" command specifying absolute values is issued, then the "Speed-position control" command cannot be issued. If it is issued in such a situation, it will end up with the error code 0x200A generated, indicating that an attempt to use an incorrect encoder type is made.

(3/3)

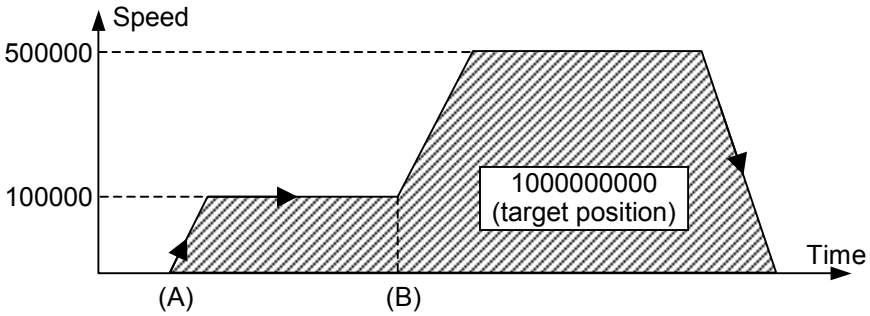
Typical use

Command: Positioning --------- Starts positioning (A).
Position reference type: INC
Axis: Axis1
Parameter block: PB1
Position reference: [50000000]
Speed reference: [100000]
Acceleration time: [10000]
Deceleration time: [10000]

WT000 (100) --------- Waits 10 seconds.

Command: Speed-position control --------- Changes the current target position and speed (B).
Axis: Axis1
Parameter block: PB1
Position reference: [10000000]
Speed reference: [500000]
Acceleration time: [500]
Deceleration time: [2000]

R000=0 --------- Waits for the positioning to be completed.

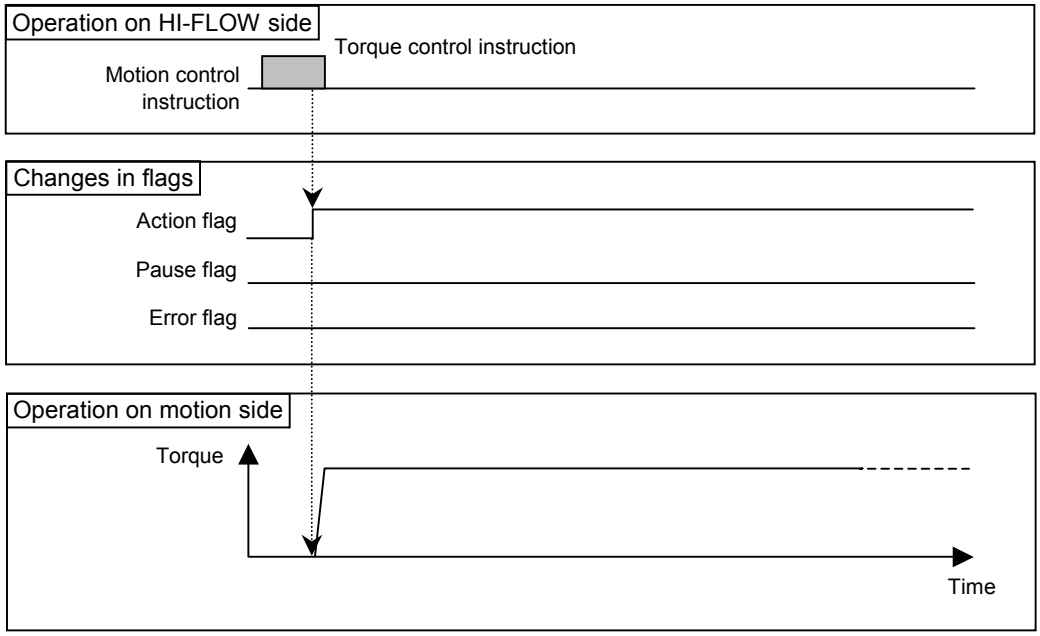


Effective parameter

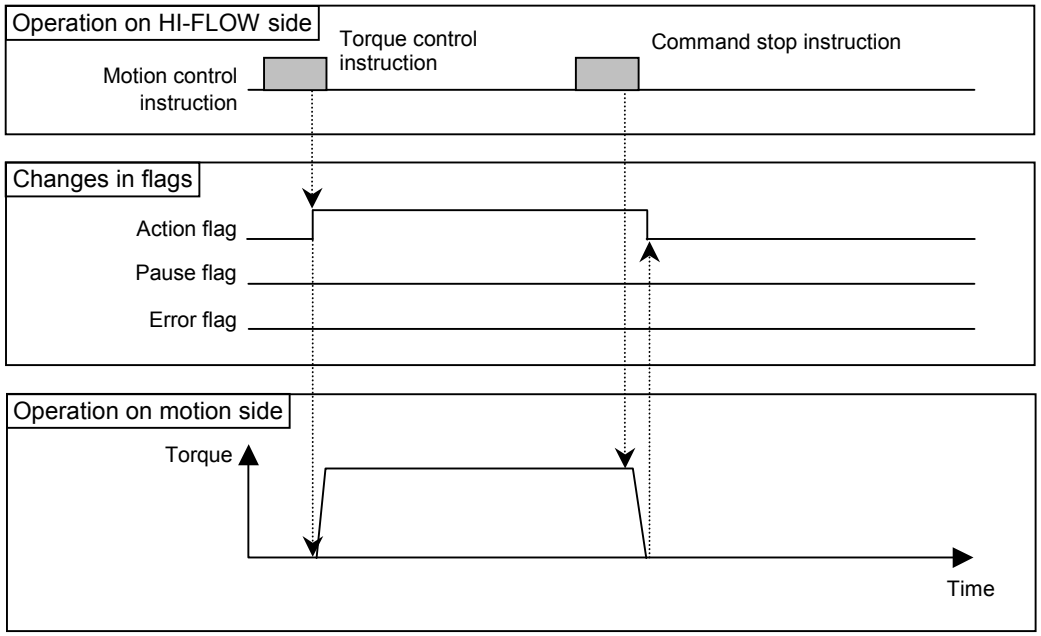
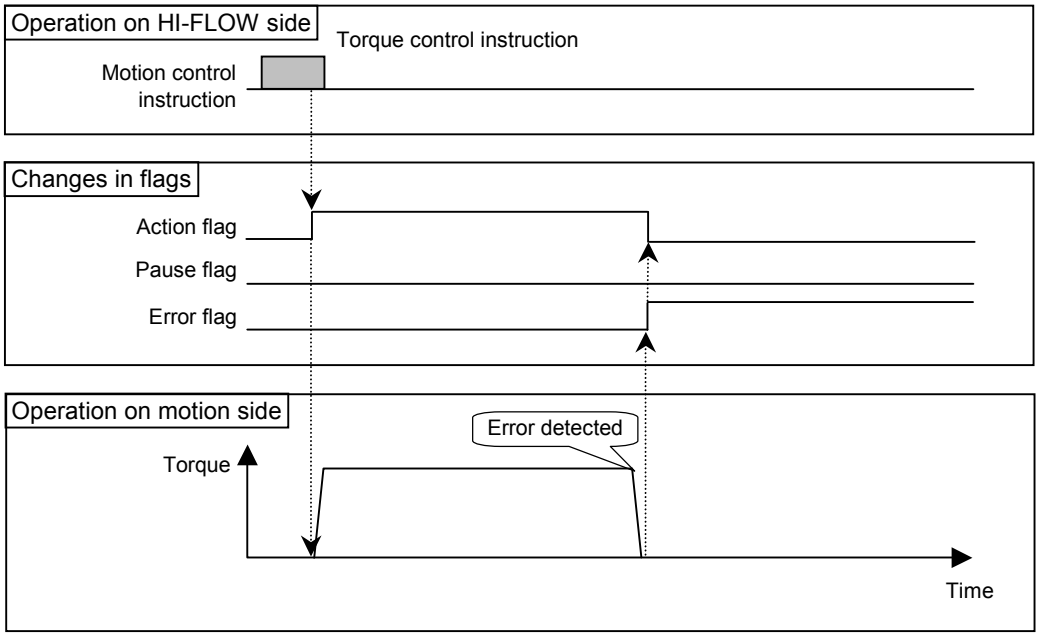
No.	Parameter name	PI/O data type	Allowable range of settings
①	Position reference	Long word	-2147483648 to 2147483647
②	Speed reference	Long word	-2147483648 to 2147483647
③	Acceleration time	Long word	0 to 2147483647
④	Deceleration time	Long word	0 to 2147483647

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

(1/3)

TRQ	Torque control
Function description	Operates a specified axis (or axes) in torque control mode.
Parameter and operation	Command: Torque control Axis: Is used to specify the axis number(s) of an axis (or axes) to be subjected to torque control. Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when either or both of the parameters ① and ② below are omitted. Torque reference: See ① below. Speed limit during torque reference: See ② below. ① Torque reference Is a new torque value to be used after changing. ② Speed limit during torque reference Is a new speed limit value to be used after changing.
Flag configuration	Action flag: Is set to 0 when a requested torque control operation fails. Pause flag: Is always set to 0. Error flag: Is set to 1 when a requested torque control operation fails. <Operation when normal> 

(2/3)

Flag configuration (continued from preceding page)	<Operation when aborted or overwriting attempted>  <Operation when abended> 
Remark	• This command can be executed even when servos of axes are in OFF state. • To end the torque control operation in progress, issue the command stop command.

(3/3)

Typical use

Command: Torque control --------- Starts torque control (A).
Axis: Axis1
Parameter block: PB1
Torque reference: [1000]
Speed limit during torque reference: 1000

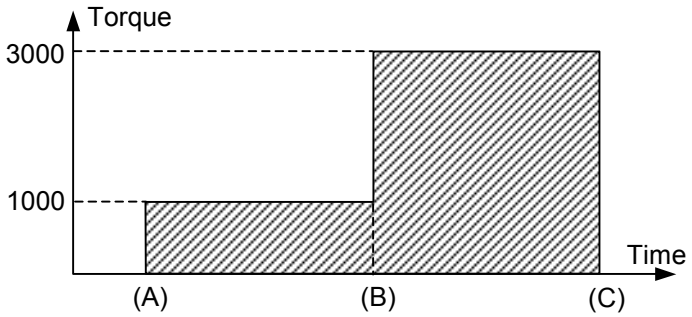
WT000 (100) --------- Waits 10 seconds.

Command: Torque control --------- Changes the parameters to the torque control command.
Axis: Axis1
Parameter block: PB1
Torque reference: [3000]
Speed limit during torque reference: 3000

WT001 (100) --------- Waits 10 seconds.

Command: Command stop --------- Ends the torque control operation in progress (C).
Axis: Axis1

R000=0 --------- Waits for the torque control operation to be ended.

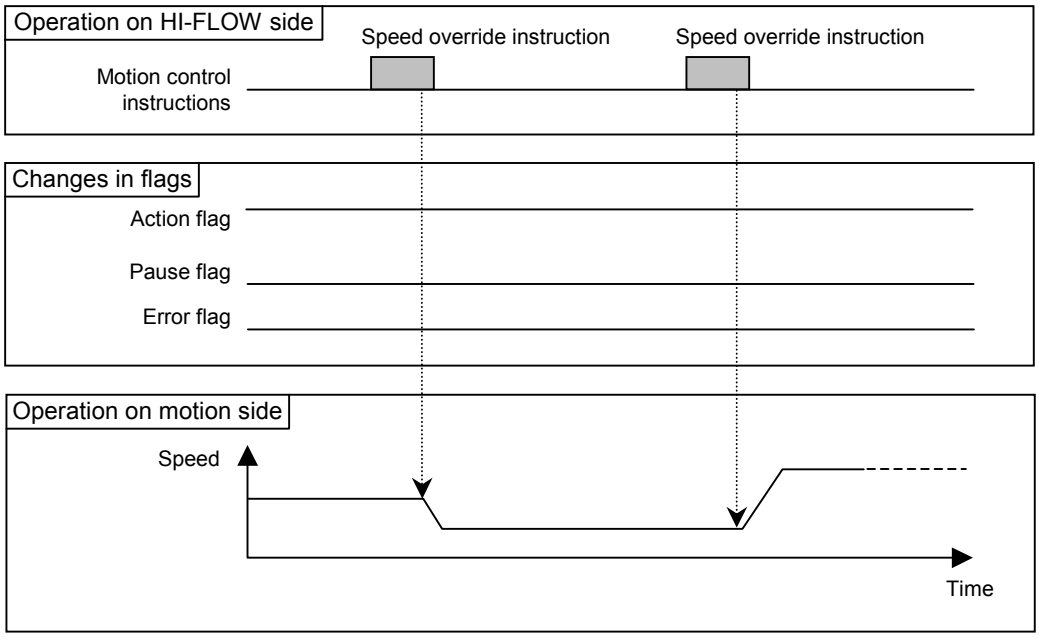


Effective parameter

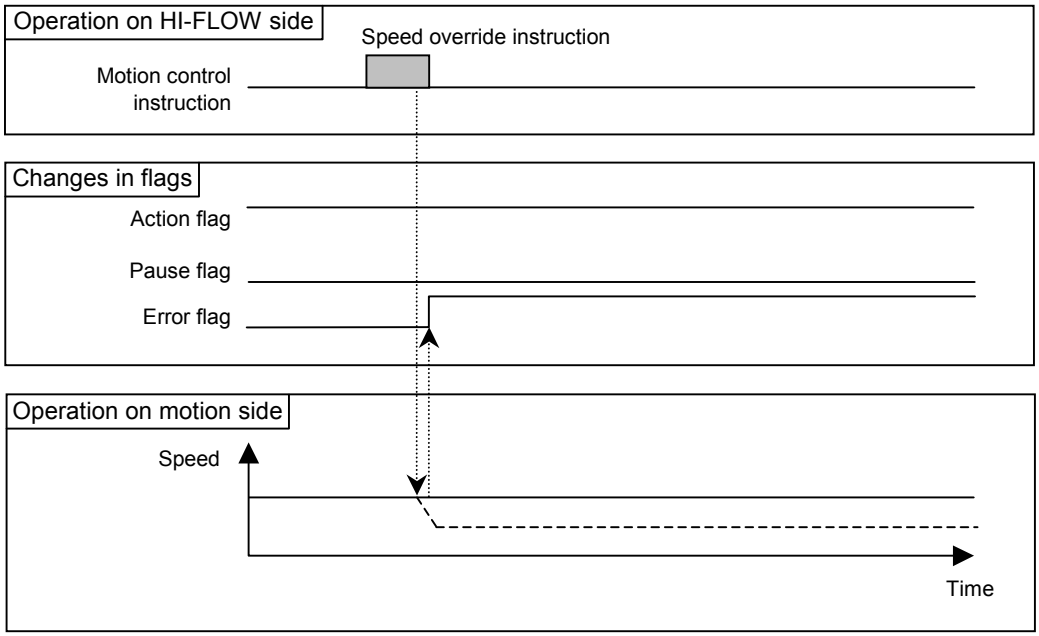
No.	Parameter name	PI/O data type	Allowable range of settings
①	Torque reference	Long word	-2147483648 to 2147483647
④	Speed limit during torque reference	Word	-32768 to 32767

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

(1/3)

CHGO	Speed override
Function description	Changes the output percentage value(s) for the set speed(s) in units of 0.01%. This changing operation can be performed during a positioning operation, in which case the set speed(s) are immediately increased or decreased as requested.
Parameter and operation	(M) Command: Speed override Axis: Is used to specify the axis number(s) of an axis (or axes) for which to change the set speed(s). Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when the parameter ① below is omitted. Speed override: See ① below. ① Speed override Is a new output percentage value to be used after changing. It must be specified in units of 0.01% for the set speed.
Flag configuration	Action flag: Remains unchanged after speed override operations. Pause flag: Is always set to 0. Error flag: Is set to 1 when a requested speed override operation fails. <Operation when normal> 

(2/3)

Flag configuration (continued from preceding page)	<Operation when abended> 
Remarks	• The speed override function remains effective at any time during operation. If you do not want to use this function, fix the current speed override setting at 10000 (= 100.00%). • If the current speed override setting is 0, then the output speed is 0, resulting in no run of the motor.

(3/3)

Typical use

Command: Constant-speed feed --------- Starts constant-speed feeding (A).
Axis: Axis1
Parameter block: PB1
Speed reference: [200000]
Acceleration time: [3000]
Deceleration time: [3000]
Direction: Forward rotation

WT000 (100) --------- Waits 10 seconds.

Command: Speed override --------- Changes the current speed override setting (B).
Axis: Axis1
Parameter block: PB1
Speed override: 5000

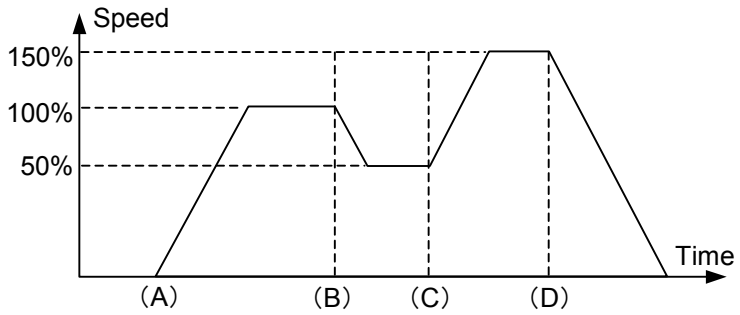
WT001 (50) --------- Waits five seconds.

Command: Speed override --------- Changes the current speed override setting (C).
Axis: Axis1
Parameter block: PB1
Speed override: 15000

WT002 (50) --------- Waits five seconds.

Command: Command stop --------- Aborts the constant-speed feed operation in progress (D).
Axis: Axis1

R000=0 --------- Waits for the constant-speed feed operation to be ended.

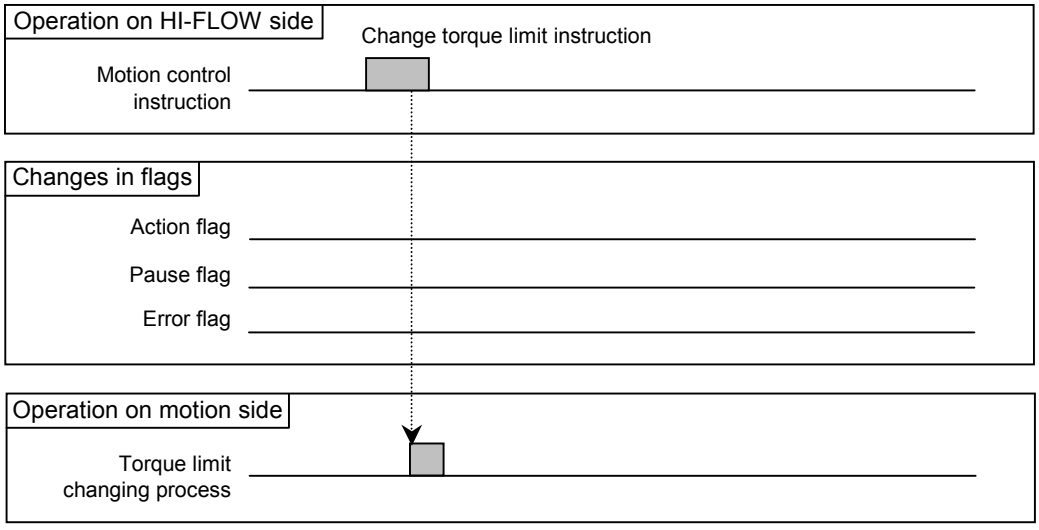


Effective parameter

No.	Parameter name	PI/O data type	Allowable range of settings
①	Speed override	Word	0 to 32767

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

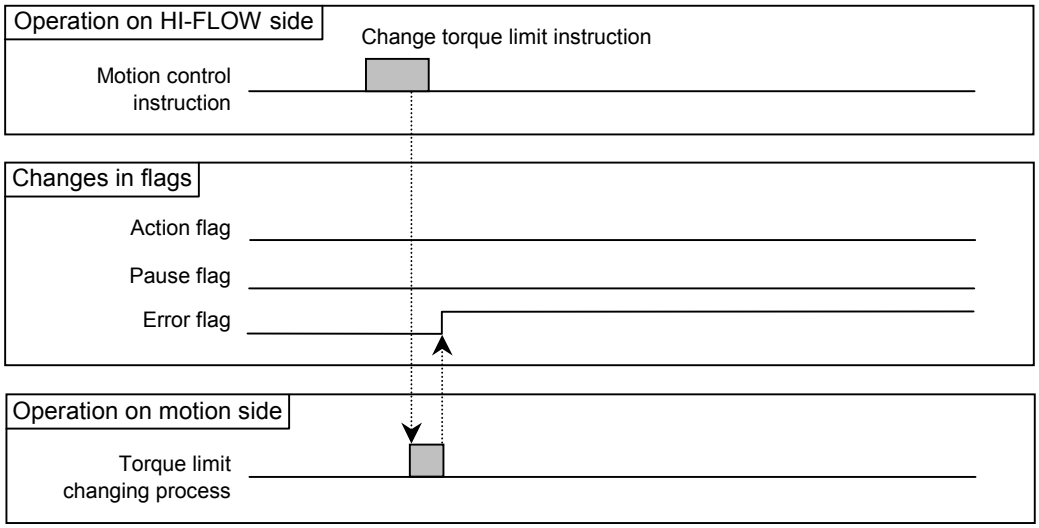
(1/2)

CHGTL	Change torque limit
Function description	Serves as a protection of the machine by limiting the rotational speeds of its motor(s) during a torque control operation.
Parameter and operation	Command: Change torque limit Axis: Is used to specify the axis number(s) of an axis (or axes) for which to change the torque limit(s). Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when either or both of the parameters ① and ② below are omitted. Forward torque limit: See ① below. Reverse torque limit: See ② below. ① Forward torque limit Is a new maximum rotational speed in forward direction that is to be used during a torque control operation. ② Reverse torque limit Is a new maximum rotational speed in reverse direction that is to be used during a torque control operation.
Flag configuration	Action flag: Remains unchanged after change torque limit operations. Pause flag: Is always set to 0. Error flag: Is set to 1 when a requested change torque limit operation fails. <Operation when normal> 

(2/2)

Flag configuration (continued from preceding page)

<Operation when abended>



Remarks

Typical use

M Command: Change torque limit ------ Starts changing the torque limits.
Axis: Axis1
Parameter block: PB1
Forward torque limit: [800]
Reverse torque limit: [800]

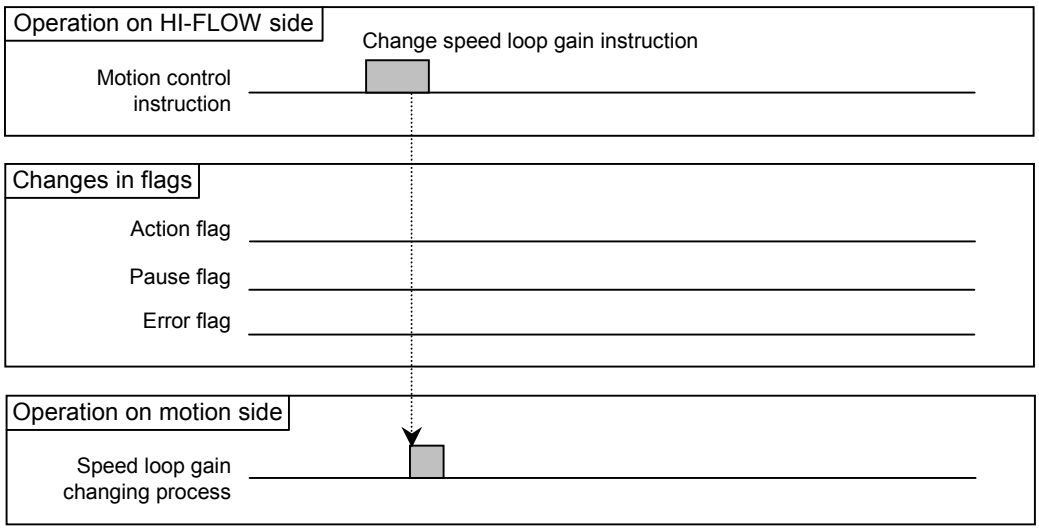
R000=0 ------ Waits for the torque limit changing operation to be completed.

Effective parameter

No.	Parameter name	PI/O data type	Allowable range of settings
①	Forward torque limit	Long word	0 to 800
②	Reverse torque limit	Long word	0 to 800

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

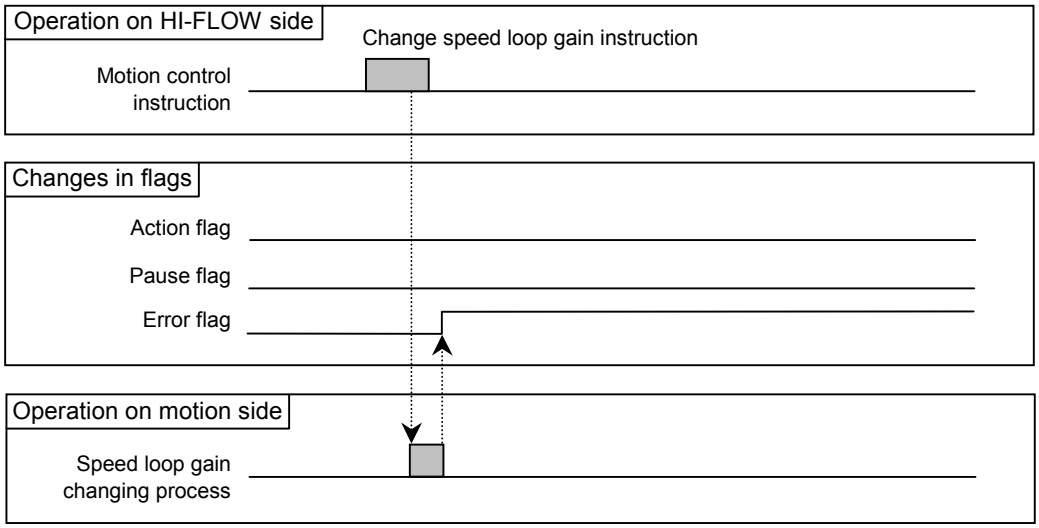
(1/2)

KVS	Change speed loop gain
Function description	Changes the set speed loop gain(s) for the servo motor(s) of a specified axis (or axes).
Parameter and operation	Command: Change speed loop gain Axis: Is used to specify the axis number(s) of an axis (or axes) for which to change the set speed loop gain(s). Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when any one or ones of the parameters ① through ③ below are omitted. Speed loop gain: See ① below. Speed feed forward compensation: See ② below. Speed integration time constant: See ③ below. ① Speed loop gain Is used to set a new responsiveness for the speed loop of the servo pack. To make the servo system stable, the responsiveness of the speed loop should be set as high as possible within the range where the machine system does not start vibrating. The setting unit of the speed loop gain is Hertz (Hz). ② Speed feed forward compensation Is used to shorten the positioning time. The setting unit of the speed feed forward compensation is 0.01%. ③ Speed integration time constant Is used to set a new responsiveness toward very small inputs. If this time constant is made larger, the responsiveness of the servo system will decrease because the time constant tends to cause delays in the servo system. The setting unit of this time constant is 0.01 ms.
Flag configuration	Action flag: Remains unchanged after speed loop gain change operations. Pause flag: Is always set to 0. Error flag: Is set to 1 when a requested speed loop gain change operation fails. <Operation when normal> 

(2/2)

Flag configuration (continued from preceding page)

<Operation when abended>



Remarks

Typical use

M Command: Change speed loop gain --------- Starts changing the speed loop gain(s).

Axis: Axis1
Parameter block: PB1
Speed loop gain: 500
Speed feed forward compensation: 500
Speed integration time constant: 2000

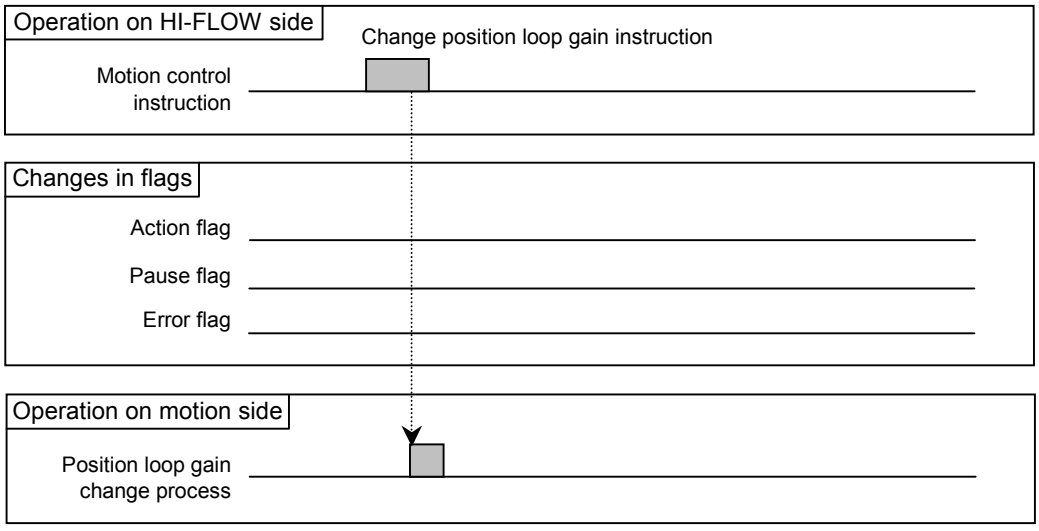
R000=0 --------- Waits for the speed loop gain changing to be completed.

Effective parameter

No.	Parameter name	PI/O data type	Allowable range of settings
①	Speed loop gain	Word	0 to 2000
②	Speed feed forward compensation	Word	0 to 32767
③	Speed integration time constant	Word	15 to 65535

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

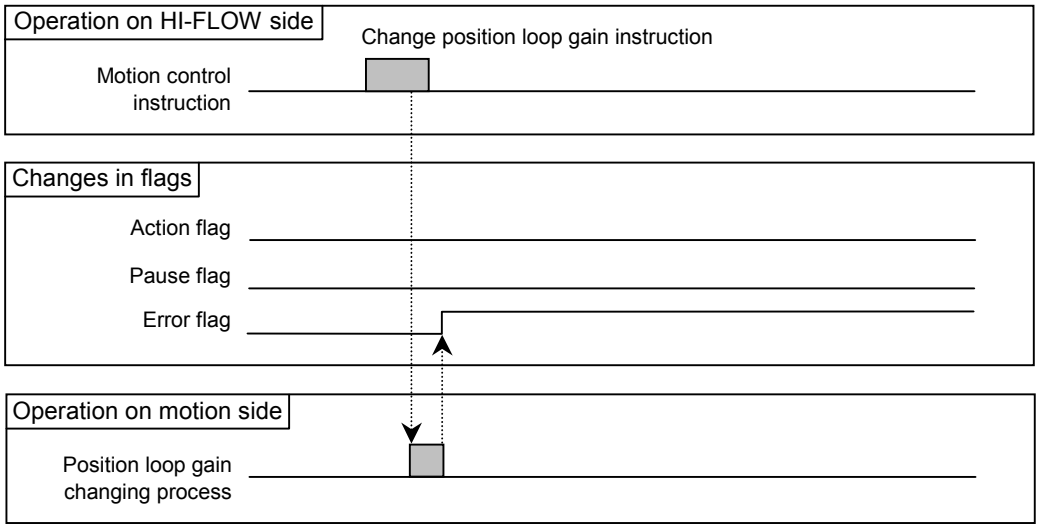
(1/2)

KPS	Change position loop gain
Function description	Changes the set position loop gain(s) for the servo motor(s) of a specified axis (or axes).
Parameter and operation	Command: Change position loop gain Axis: Is used to specify the axis number(s) of an axis (or axes) for which to change the set position loop gain(s). Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when either or both of the parameters ① and ② below are omitted. Position loop gain: See ① below. Position integration time constant: See ② below. ① Position loop gain Is used to set a new responsiveness for the position loop of the servo pack. The higher the set position loop gain, the shorter the positioning time will be. An optimal value must be set for this parameter in accordance with the machine rigidity, inertia, and the type of the servo motor. The setting unit of the position loop gain is 0.1/s. ② Position integration time constant Is used to improve the follow-up precision in electronic cams, shafts, or other applications. The setting unit of this time constant is millisecond (ms).
Flag configuration	Action flag: Remains unchanged after position loop gain change operations. Pause flag: Is always set to 0. Error flag: Is set to 1 when a requested position loop gain change operation fails. <Operation when normal> 

(2/2)

Flag configuration (continued from preceding page)

<Operation when abended>



Remarks

Typical use

M Command: Change position loop gain ---------- Starts changing the position loop gain(s)

Axis: Axis1
Parameter block: PB1
Position loop gain: 1000
Position integration time constant: 0

R000=0 ---------- Waits for the position loop gain changing to be completed.

Effective parameter

No.	Parameter name	PI/O data type	Allowable range of settings
①	Position loop gain	Word	0 to 32767
②	Position integration time constant	Word	0 to 32767

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

(1/4)

CHGU	Set unit
Function description	Changes a desired unit(s) among those currently set for parameter values for control purposes.
Parameter and operation	Ⓜ Command: Set unit Axis: Is used to specify the axis number(s) of an axis (or axes) for which to change a set unit(s). Parameter block: Is used to specify the ID number(s) of a parameter group(s) that will be referenced by default when any one or ones of the parameters ① through ⑩ below are omitted. Speed unit: See ① below. ACC/DCC units: See ② below. Filter type: See ③ below. Torque unit: See ④ below. External positioning signal: See ⑤ below. Positioning completed width: See ⑥ below. Positioning proximity detection width: See ⑦ below. Acceleration time: See ⑧ below. Deceleration time: See ⑨ below. Filter time constant: See ⑩ below. ① Speed unit Is a new unit of velocity to be used after changing. One of the following four speed units is specifiable. Option / Remarks Reference unit/sec 10**n reference unit/min — Default 0.01% 0.0001% ② ACC/DCC units Is a new acceleration/deceleration unit to be used after changing. One of the following two ACC/DCC units is specifiable. Option / Remarks Reference unit/sec**2 ms — Default ③ Filter type Is a new acceleration/deceleration filter type to be used after changing. One of the following three filter types is specifiable. Option / Remarks No filter — Default Exponential ACC/DCC filter Movement averaging filter

(2/4)

Parameter and operation (continued from preceding page)

④ Torque unit
Is a new torque unit to be used after changing. One of the following two torque units is specifiable.

Option	Remarks
0.01%	Default
0.0001%	

⑤ External positioning signal
Is a new external signal to be used in external positioning after changing. One of the following four external positioning signals is specifiable.

Option	Remarks
Phase-C pulse input signal	
/EXT1	Default
/EXT2	
/EXT3	

⑥ Positioning completed width
Is a new width to be used after changing, the width in which, after position reference distribution has been completed, the Positioning Completed signal will be turned on during positioning. Any specified width value must be in conformity with the system's machine specifications. If the specified width value is too small, the positioning will take time to complete.

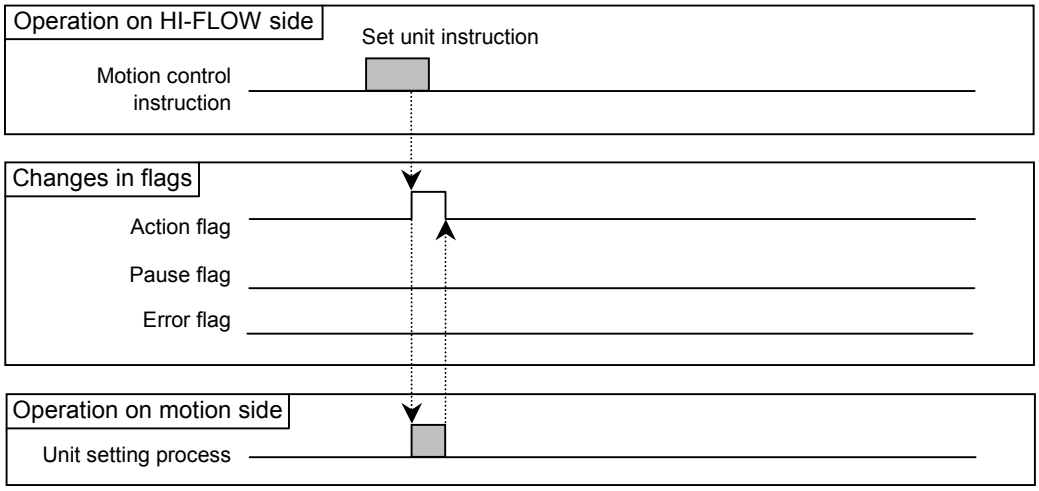
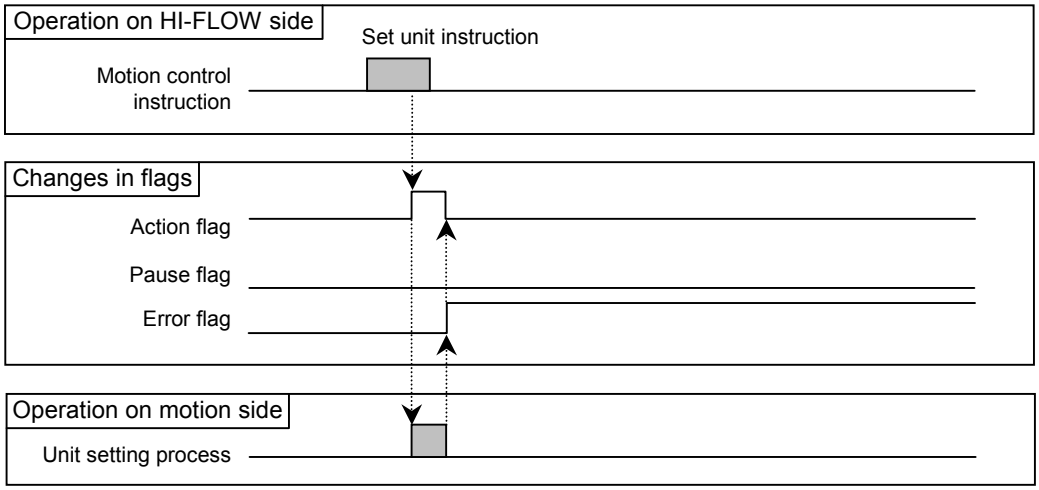
⑦ Positioning proximity detection width
Is a new width to be used in checking after changing to see if the difference in absolute value between a target position and a feedback position is within that new width. If it is within that width, the Position Proximity bit is set to 1.

⑧ Acceleration time
Is a new acceleration time constant or acceleration rate to be used in positioning after changing. The setting unit of the acceleration time depends on the current setting of "② ACC/DCC units."

⑨ Deceleration time
Is a new deceleration time constant or deceleration rate to be used in positioning after changing. The setting unit of the deceleration time depends on the current setting of "② ACC/DCC units."

⑩ Filter time constant
Is a new acceleration/deceleration filter time constant to be used after changing. The setting unit of the filter time constant is 0.1 ms.

(3/4)

Flag configuration	Action flag: Is set to 1 when unit setting starts, and is set to 0 when it ends. It is also set to 0 when unit setting fails. Pause flag: Is always set to 0. Error flag: Is set to 1 when unit setting fails. <Operation when normal>  <Operation when abended> 
Remarks	The acceleration and deceleration times can be specified within the range 0 to 32767 (ms) if the current ACC/DCC unit setting is millisecond (ms) or the default. If an acceleration or deceleration time value larger than 32767 is specified for external positioning, it will be cramped at 32767 ms and then external positioning will be performed, followed by the setting (= 1) of the Setting Parameter Error flag.

(4/4)

Typical use

Command: Set unit ---------- Starts unit setting.
Axis: Axis1
Parameter block: PB1
Speed unit: Reference unit/sec
ACC/DCC units: Reference unit/sec**2

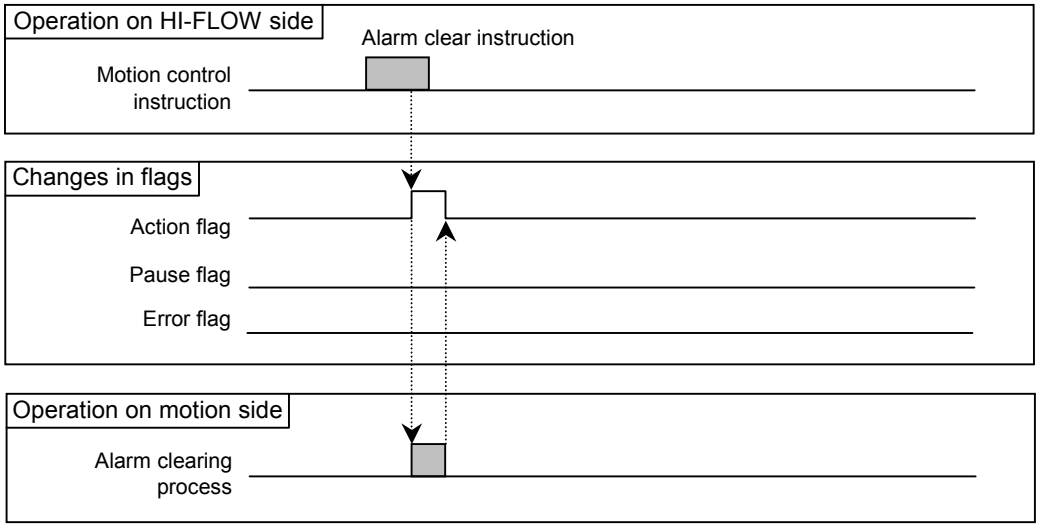
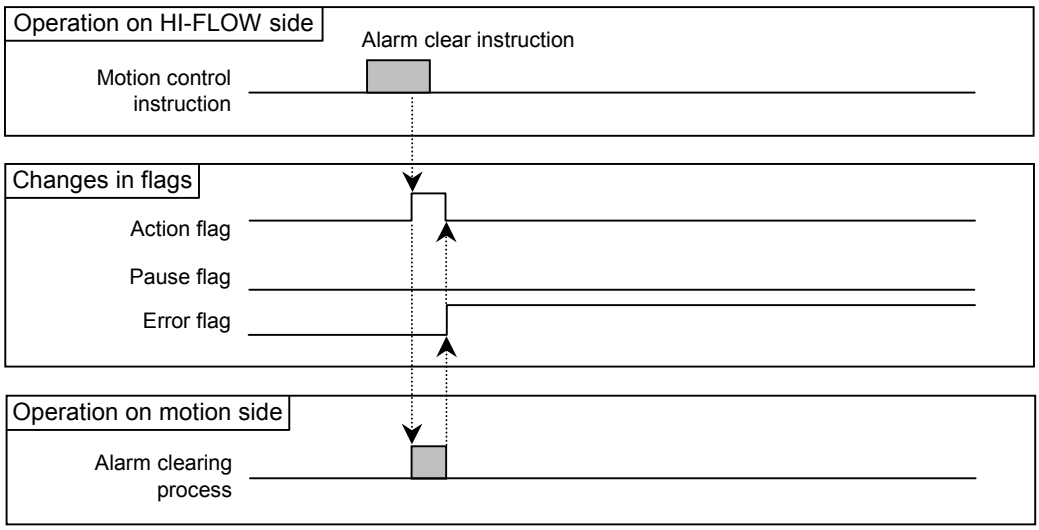
R000=0 ---------- Waits for the unit setting to be completed.

Effective parameter

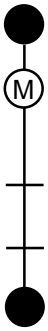
No.	Parameter name	PI/O data type	Allowable range of settings
①	Speed unit	Word	0: Reference unit/sec 1: 10**n reference unit/min 2: 0.01% 3: 0.0001%
②	ACC/DCC units	Word	0: Reference unit/sec**2 1: ms
③	Filter type	Word	0: No filter 1: Exponential ACC/DCC filter 2: Movement averaging filter
④	Torque unit	Word	0: 0.01% 1: 0.0001%
⑤	External positioning signal	Word	2: Phase-C pulse input signal 3: /EXT1 4: /EXT2 5: /EXT3
⑥	Positioning completed width	Long word	0 to 65535
⑦	Positioning proximity detection width	Long word	0 to 65535
⑧	Acceleration time	Long word	0 to 2147483647
⑨	Deceleration time	Long word	0 to 2147483647
⑩	Filter time constant	Word	0 to 65535

(If PI/O data is specified indirectly, any attempt to specify an odd-numbered address together will result in a parameter error.)

(1/2)

ALMCLR	Alarm clear
Function description	Clears the alarm(s) for a specified axis (or axes).
Parameter and operation	Command: Alarm clear Axis: Is used to specify the axis number(s) of an axis (or axes) for which to clear the alarm(s). (The axis number[s] specified must be within the range 1 to 32.) <Notice> After the completion of alarm clearing, be sure to wait 100 ms or more.
Flag configuration	Action flag: Is set to 1 when alarm clearing starts, and is set to 0 when it ends. Pause flag: Is always set to 0. Error flag: Is set to 1 when alarm clearing fails. <Operation when normal>  <Operation when abended> 

(2/2)

Remarks	
Typical use	 M Command: Alarm clear --------- Starts clearing the alarm. Axis: Axis1 R000=0 --------- Waits for the alarm clearing to be completed. WT001 (1) --------- Waits 100 ms.
Effective parameter	

(1/1)

NOP	No operation (or command)
Function description	Clears all motion control instruction information retained on motion controller side. If another motion control instruction is already under execution, the NOP instruction aborts the execution of that other motion control instruction and clears the information - this operation is identical to the command stop (ABORT) operation.
Parameter and operation	Command: NOP Axis: Is used to specify the axis number(s) of an axis (or axes) for which to execute the NOP instruction. (The axis number[s] specified must be within the range 1 to 32.)
Flag configuration	Action flag: If the NOP command is executed in cases where no other motion command is under execution, remains unchanged thereafter. If another motion command is under execution, it is aborted by the NOP command and this flag is set to 0. Error flag: Is set to 1 when instruction information clearing fails.
Remarks	Be sure to issue the NOP command after an external positioning command or home position return command is executed.
Typical use	See the typical example of the external positioning instruction or home position return instruction shown.
Effective parameter	

6.6 Sample Program

This section shows an example of a program that positions two axes by using motion control instructions.

■ Description of what the program does

The program:

① Positions two axes simultaneously in forward direction and then in reverse direction if a value of 1 is set in FW100.

② Positions the axes by rotating them by 90 degrees in forward direction four times (i.e., they make one complete rotation) and then by rotating them by 90 degrees in reverse direction four times if a value of 2 is set in FW100.

③ Repeats Steps ① and ② if a value of 0x1234 is set in FW100.

④ Forcibly stops the motion control in progress if a value of 0 is set in FW100.

■ Required parameter settings

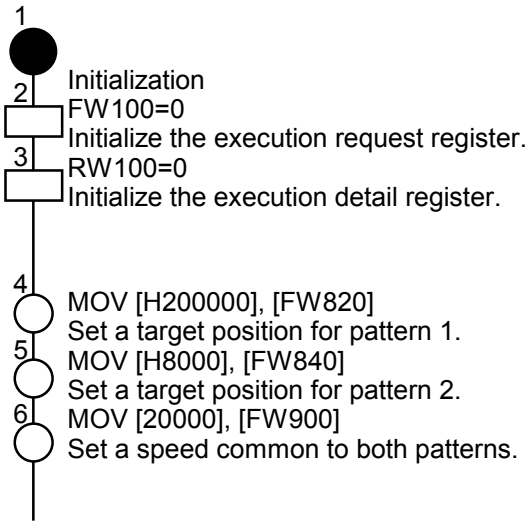
Action flag: R000 to R01F

Pause flag: R020 to R03F

Error flag: R040 to R05F

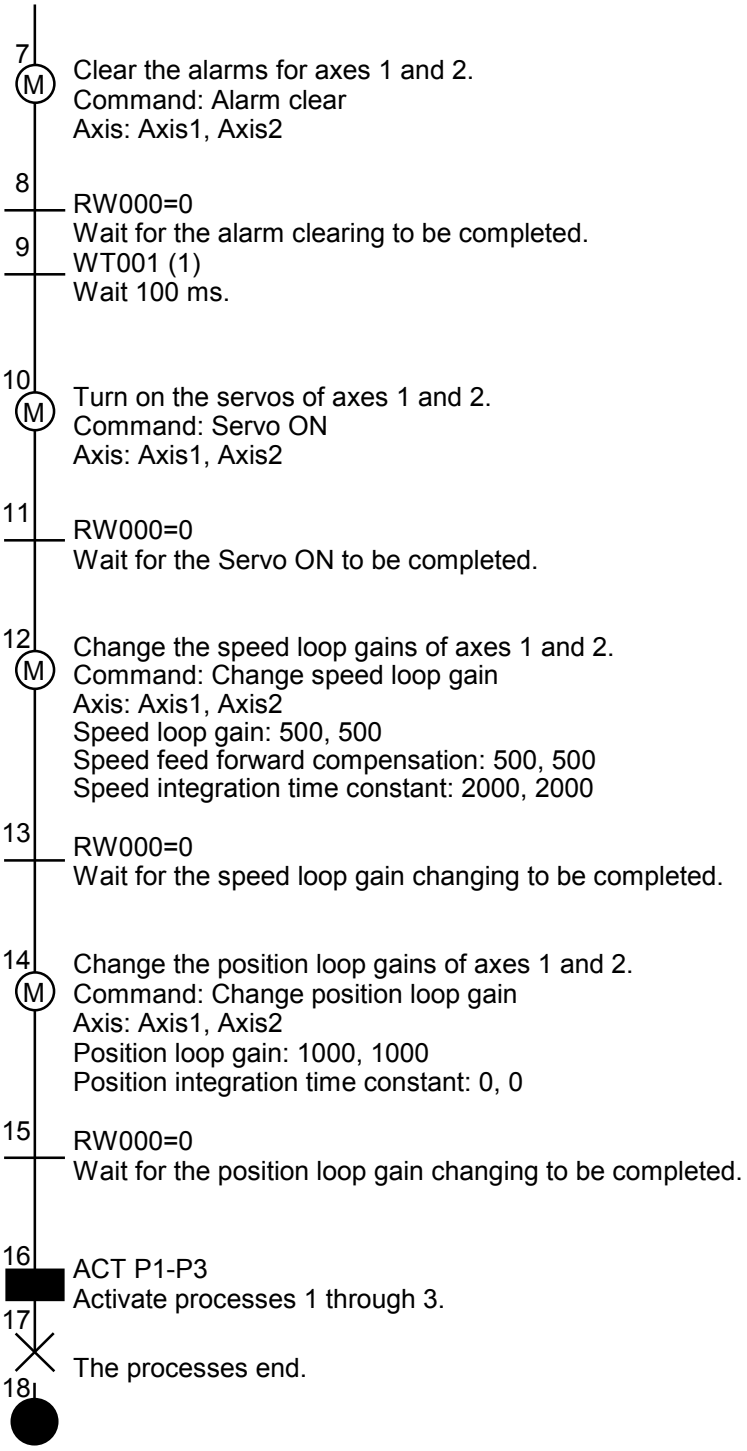
■ Sample program

(1) Process 0: Initialization

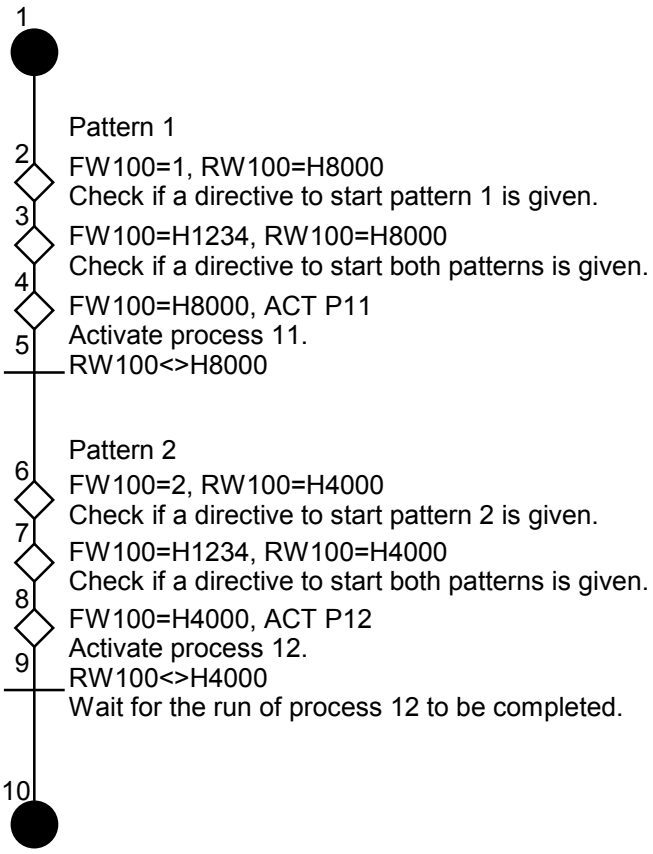


<Continued on next page>

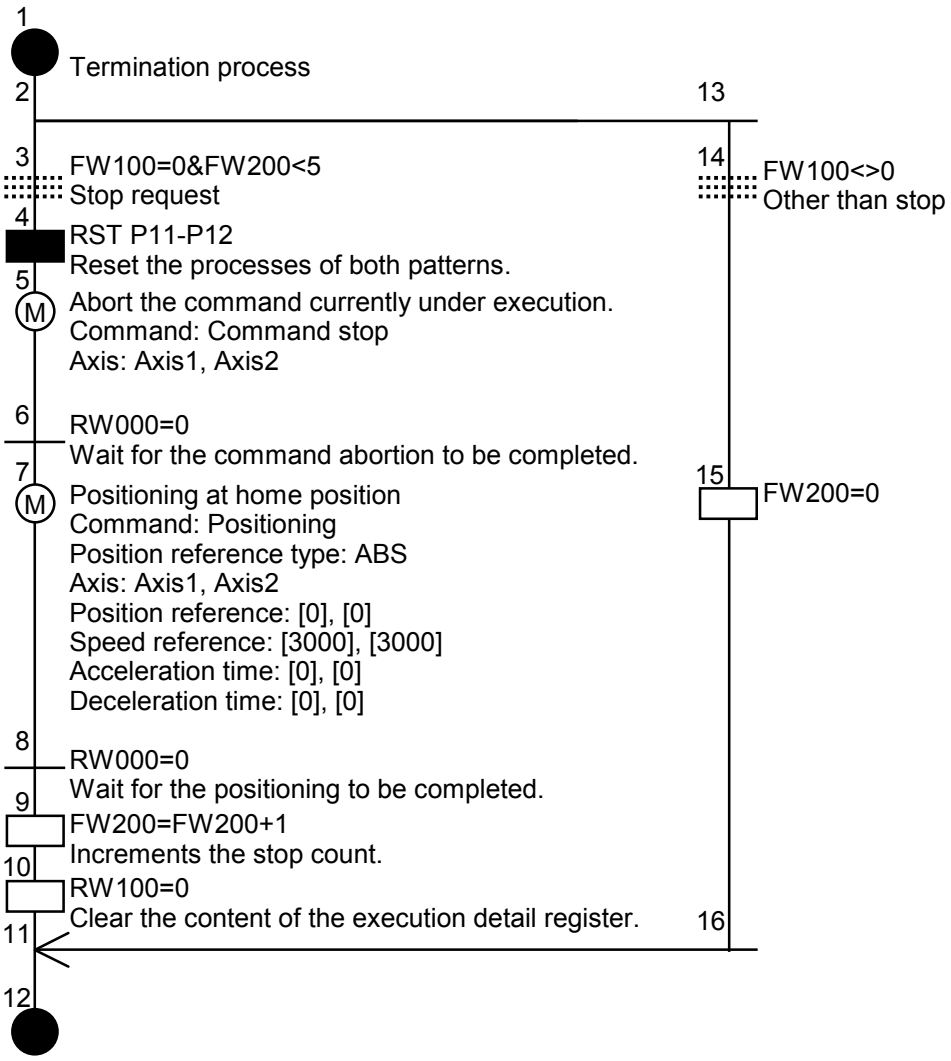
<Continued from preceding page>



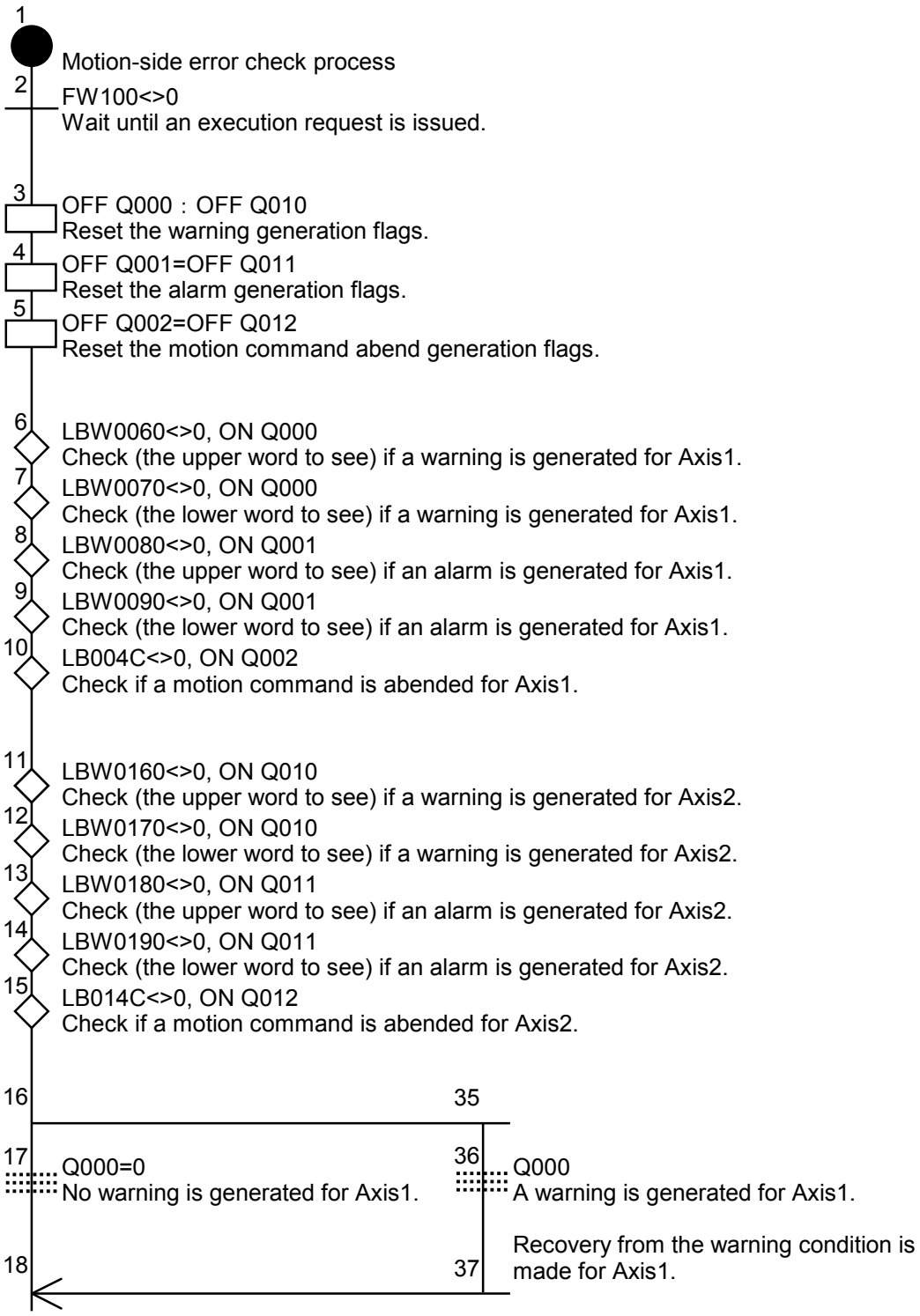
(2) Process 1: Activation of each pattern



(3) Process 2: Termination

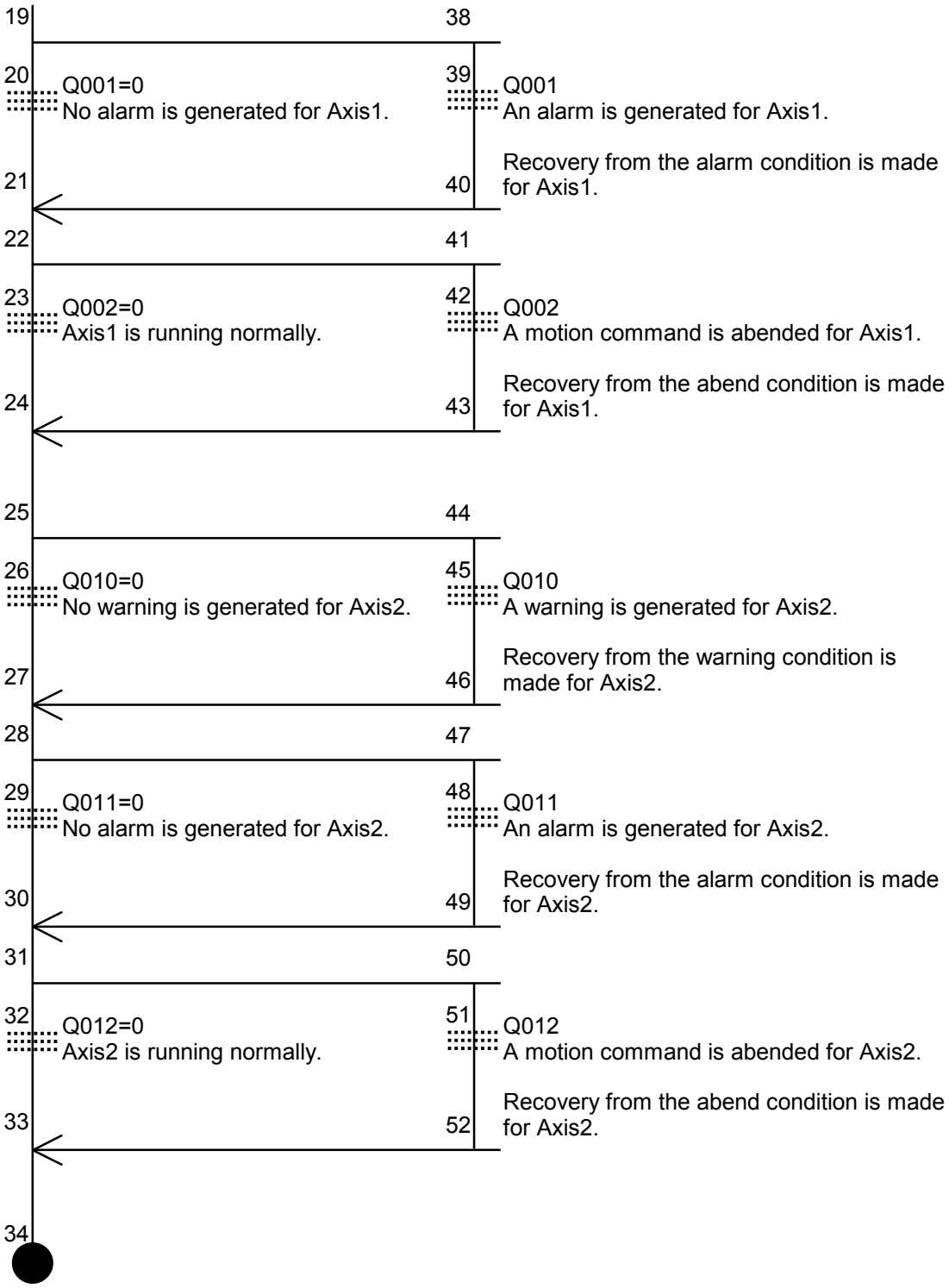


(4) Process 3: Motion-side error checking

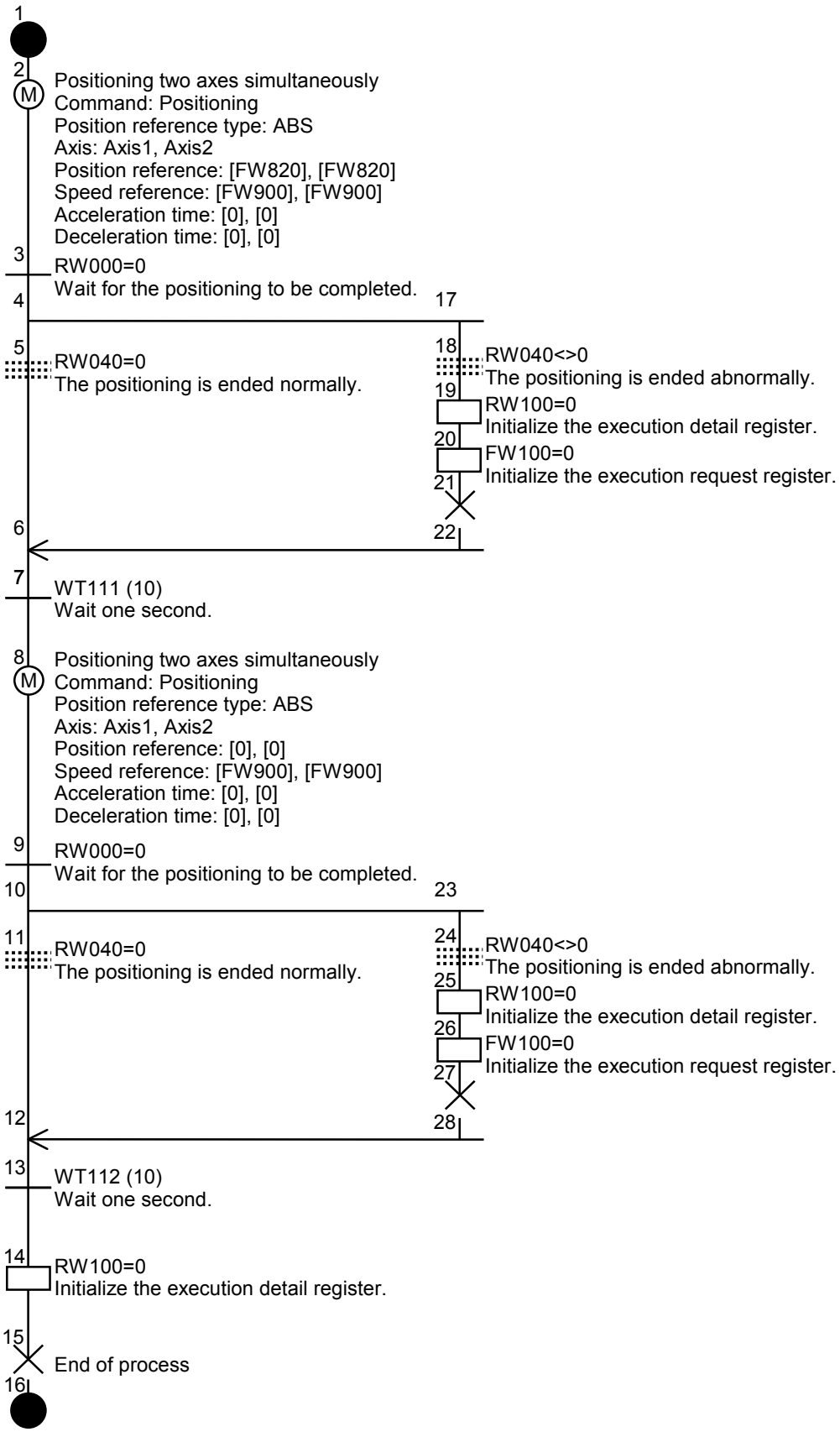


<Continued on next page>

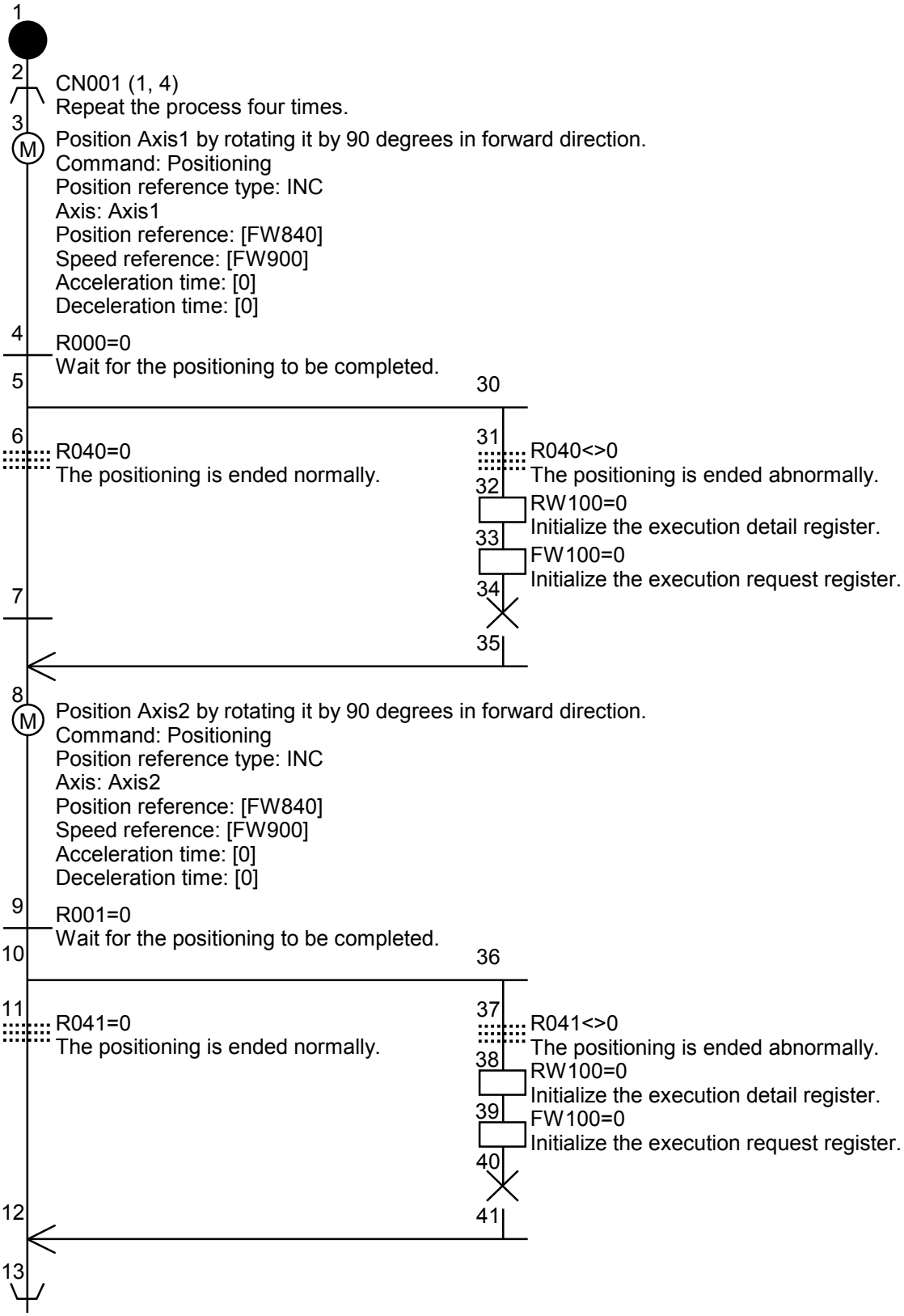
<Continued from preceding page>



(5) Process 11: Simultaneous control of two axes

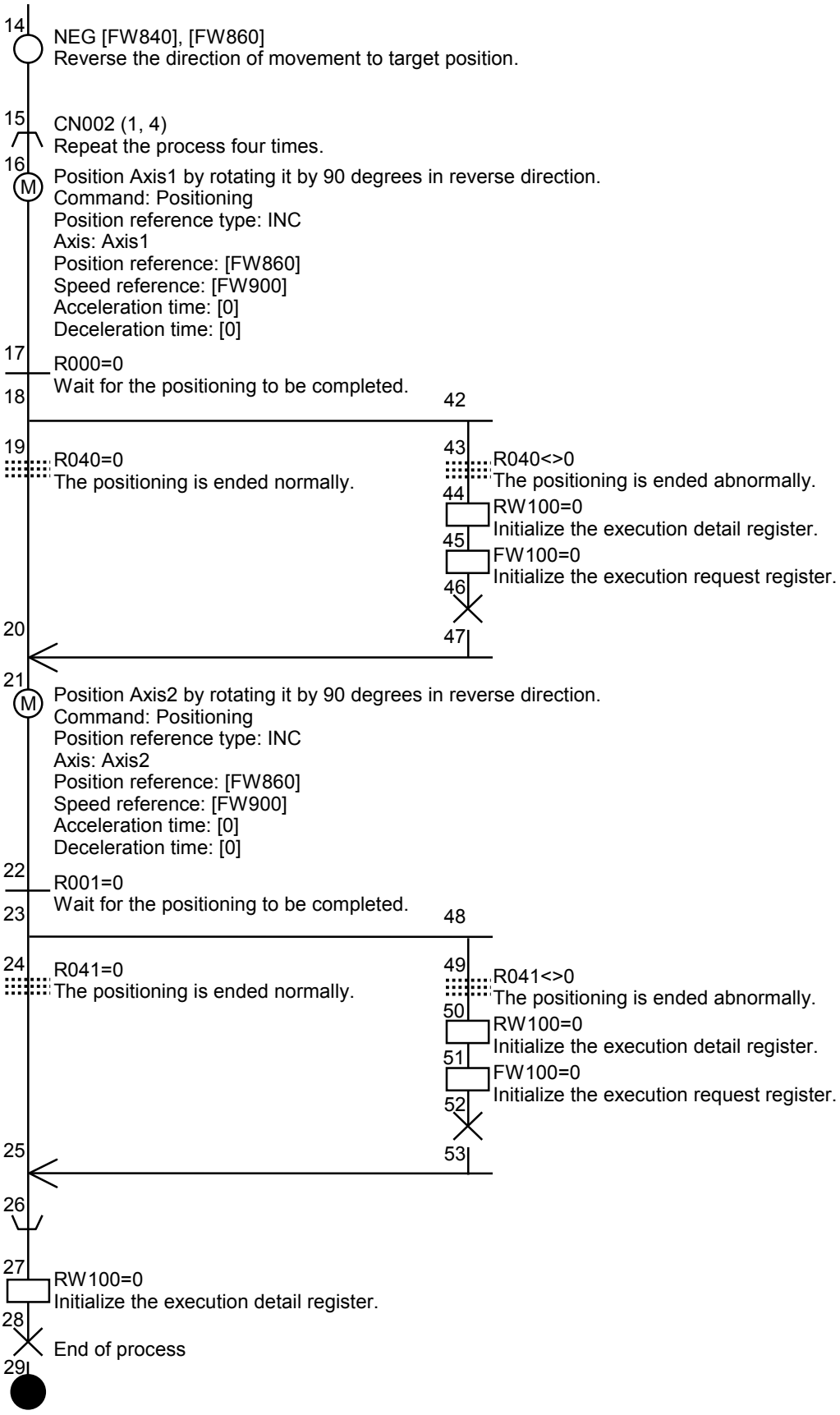


(6) Process 12: Rotating each axis by 90 degrees



<Continued on next page>

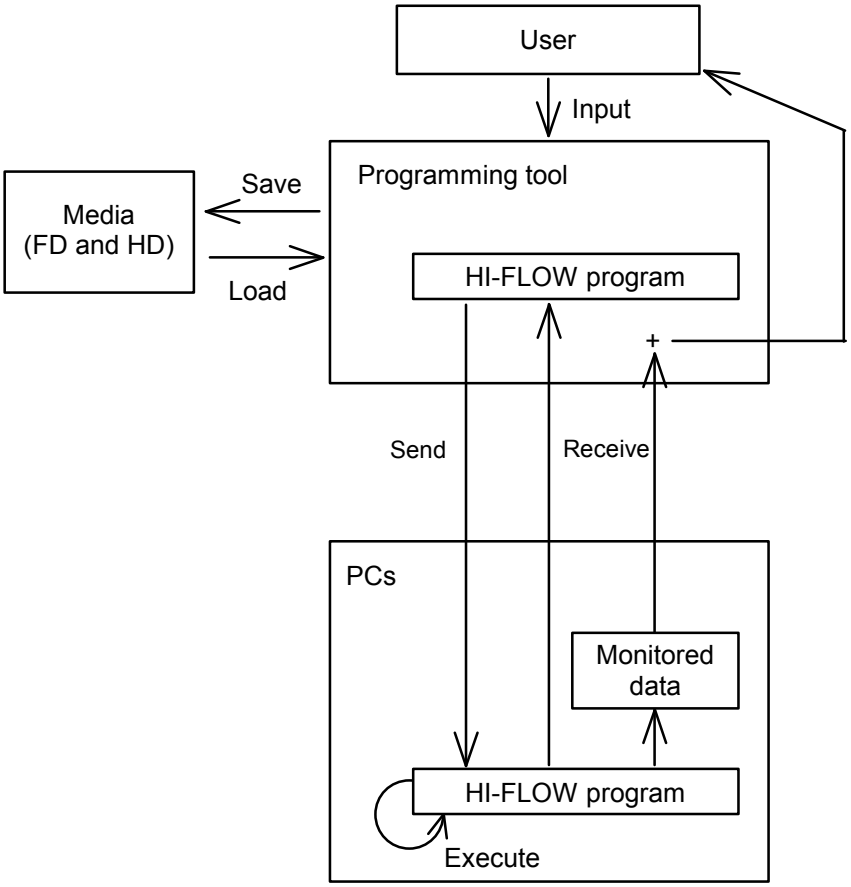
<Continued from preceding page>



SUPPLEMENTS

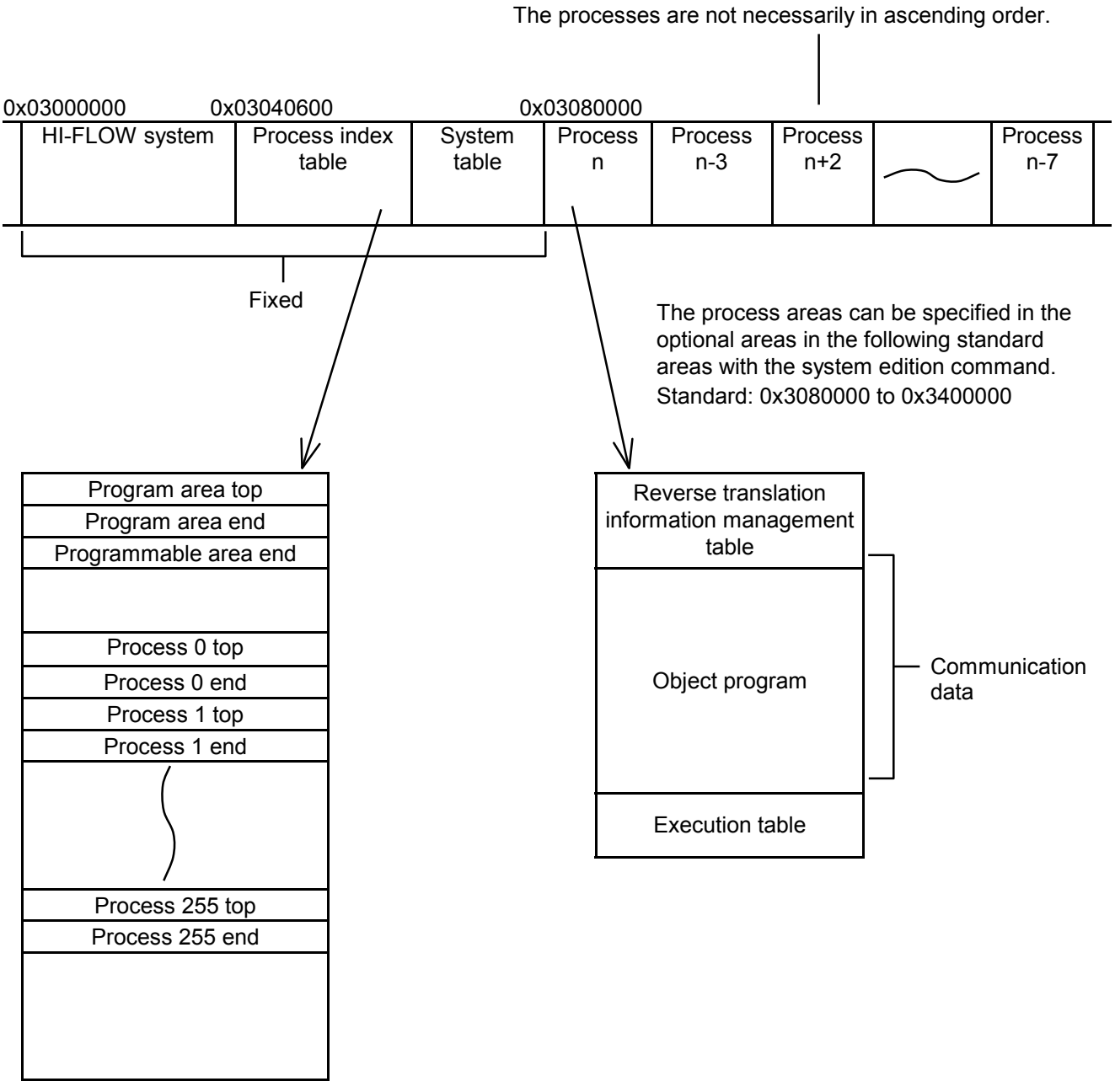
Supplement 1　Flow of the HI-FLOW Program

A HI-FLOW program is created with a programming tool and executed by the PCs.　When monitoring an execution result or similar, the system receives a necessary minimum amount of data from the PCs, synthesizes it with a program included in the tool, and outputs it.　The aim is to minimize traffic in order to increase monitoring speed.
The system also gives and receives data with different media (FD and HD) to save and load created programs.



Supplement 2 PCs Memory

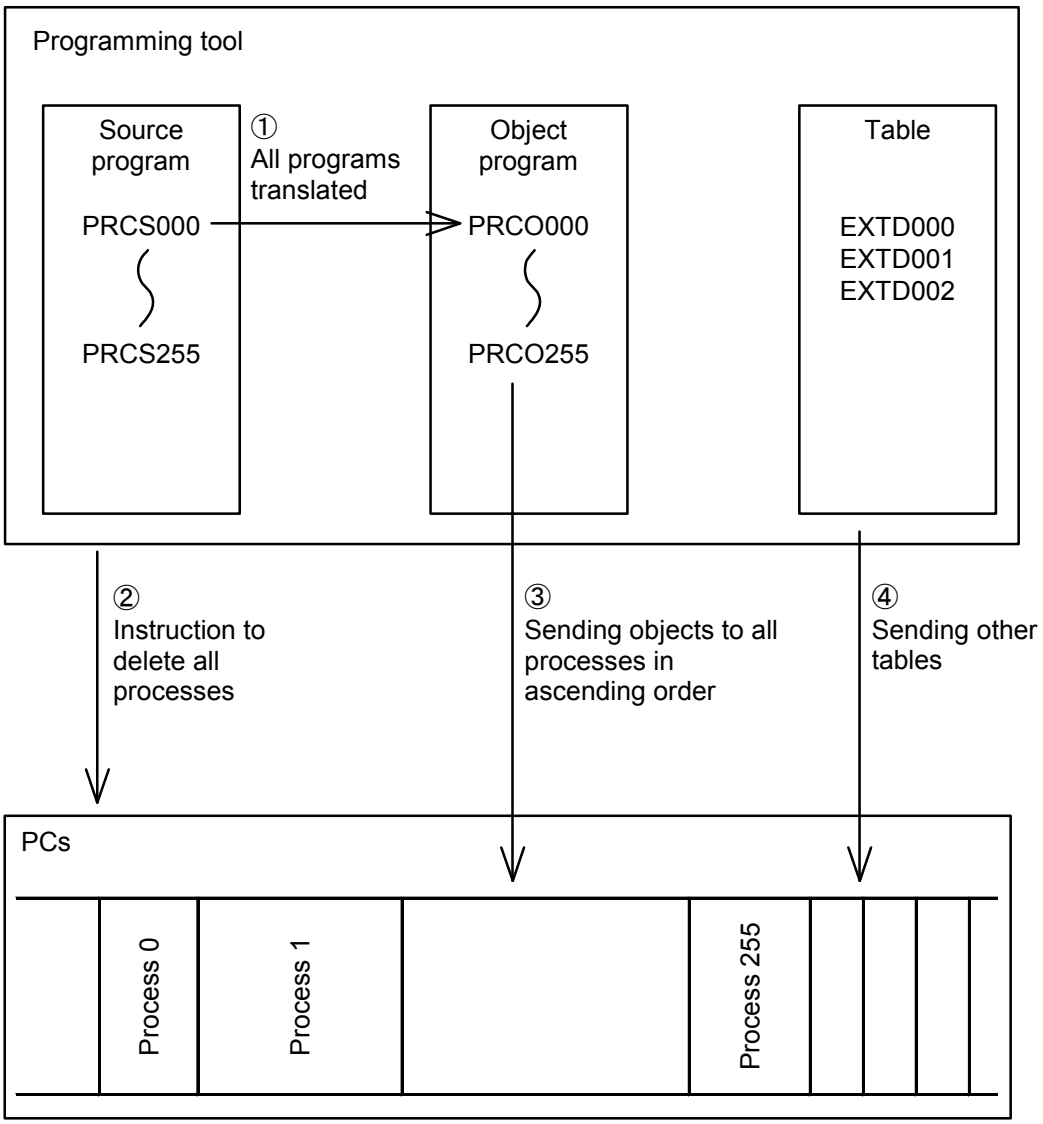
HI-FLOW programs executed on the PCs exist in the areas specified below in the CMU module. They are actually arranged in memory on the PCs. Here is an image of the memory map.



Supplement 3 Online Mode

"Offline" means the mode in which an object to be edited is turned into a program as part of the programming tool regardless of the contents of the PCs memory.
"Online" means the mode in which an object to be edited or put into memory is turned into a PCs program. However, when the object is turned into a PCs program, the tool-side program and the PCs program need to correspond as not all data needed for monitoring is read from the PCs (it takes time to communicate). One method of making them match is by sending and receiving. Alternatively, the HI-FLOW program is completed in the process, so that the process can be edited and monitored if a single process makes a match. All processes/one process are communicated partly to save time.

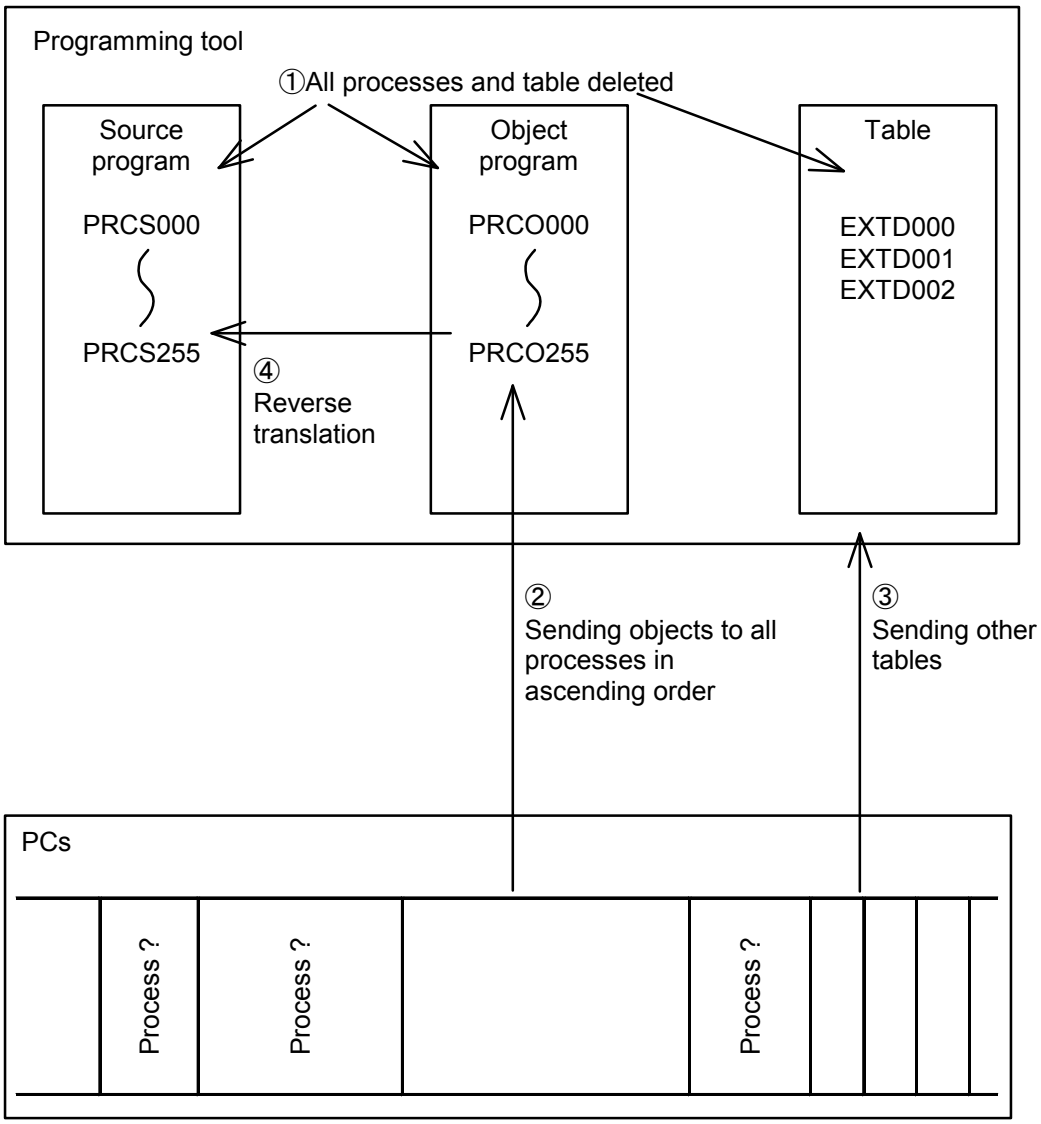
(1) Sending all processes
Here is the flow of data when all HI-FLOW programs existing on the tool are sent to the PCs.



After all processes are sent, the processes and tables on memory will take ascending order.

(2) Receiving all processes

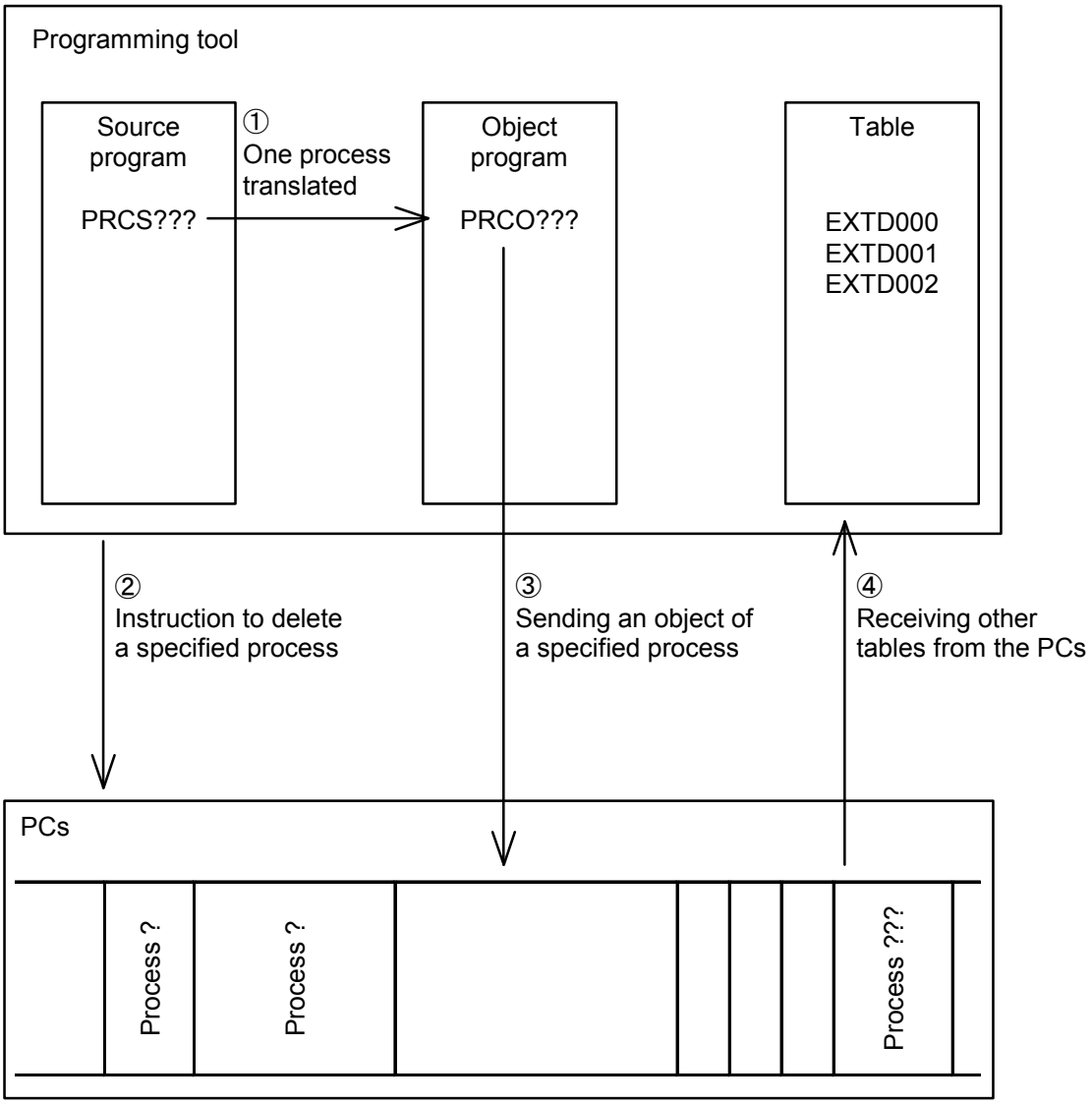
Here is the flow of data when all HI-FLOW programs existing on the PCs are sent to the tool.



There is no guarantee that the processes and tables on memory will be saved in ascending order when received.

(3) Sending one process

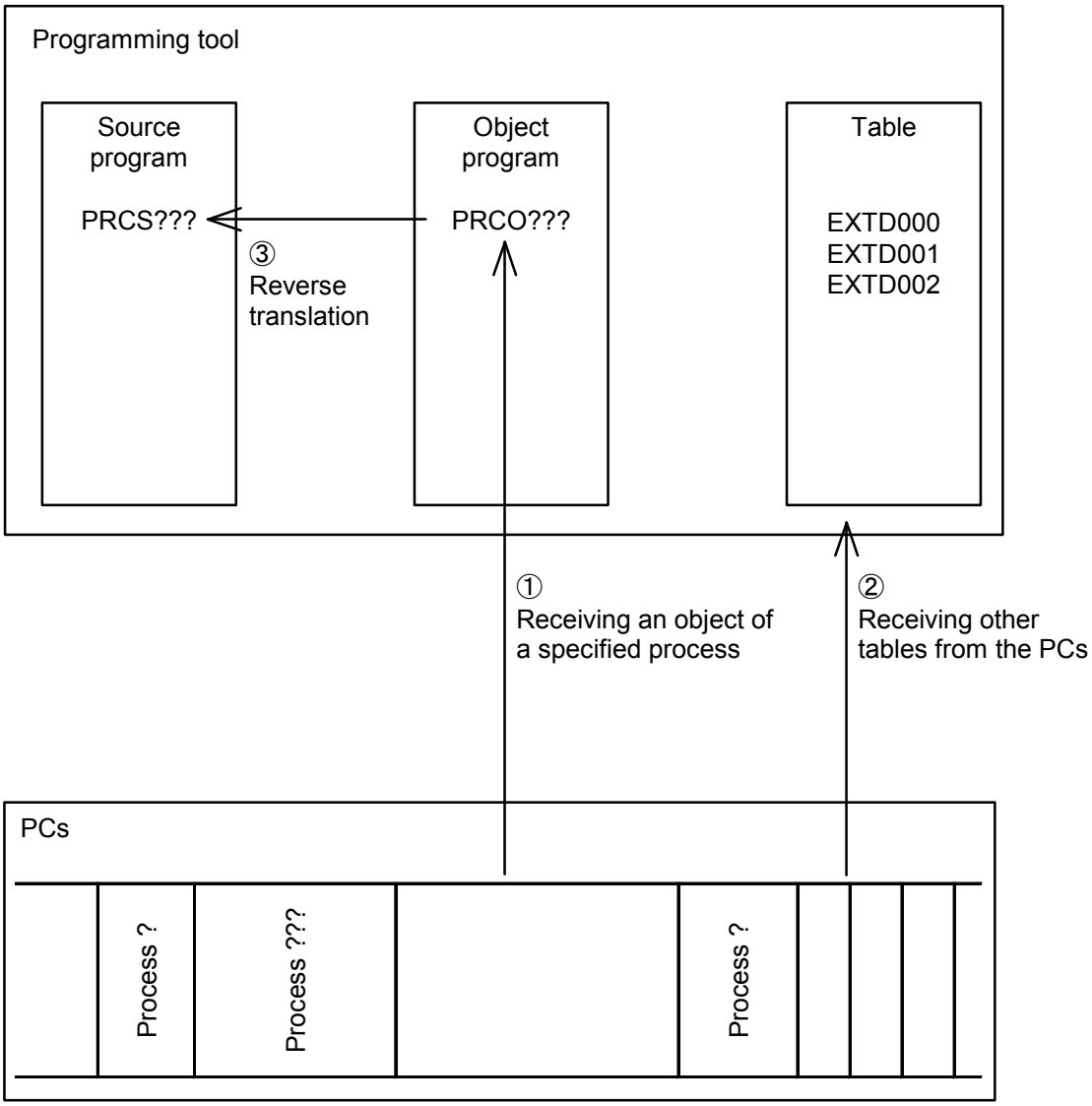
Here is the flow of data when a single particular HI-FLOW program existing on the tool is sent to the PCs.



After the sending, the specified process comes to the last on memory.

(4) Receiving one process

Here is the flow of data when a single HI-FLOW process existing on the PCs is received by the tool.



There is no guarantee that the processes and tables on memory will be saved in ascending order when received.

Supplement 4 Progress Check

HI-FLOW indicates the progress position of an item on a user program with a monitor cursor.
HI-FLOW systems on the PCs manage the progress position of an item at the present.
This supplement shows how a user program transferred to the PCs is checked for progress on the PCs.

Item	Description
Basic rules	The system will check the progress for each scan interval for the PCs. ACT-started processes will be checked in progress in ascending order of process number. The system will check the progress in ascending order of route number in the same process. (The root number increases on the screen as follows: left top < left bottom < right top < right bottom.) It will progress in ascending order of step number in the same route. When one step is complete, the system goes to the next step. If progress is impossible, the system will check the progress of the route of the next number. In the next scan, a progress check of this process and route will be conducted with this step first.
Call process	Called processes will be checked in progress after the source process and source route. When a progress check is complete on the called process, and if the process has not yet been executed, the system will check the progress of the route after the source process and source route. When it is complete, the system will check the progress of the step after the source route.
Process control	ACT-started processes are checked in progress at the next scan if the process number is smaller than the ACT-started process whose progress is being checked at that point in time, and at the same scan if larger. The RST, STP, and CLR will be processed when both the control box and process start.
Constant monitoring	The conditions (ACT, STP, RST, and CLR) for process start and the multi-entry conditions will be checked before the particular process is checked in progress. When the conditions hold, the system will perform the operation before such a progress check of the process. The system will check the Y output conditions with an interlock before the first progress check of the process and turn on and off the Y output.
Branching	After executing a branch step (if and jump), the system will check the progress of the destination step. Therefore, the execution route may not be subjected to a progress check for one scan, or be checked twice in progress. Note that the closed loop without progress conditions may be infinitely executed.
Repetition	The repeat end then checks the progress of repeat start. Note that repeating without progress conditions may constitute an infinite execution.
Shutdown	The system will execute an escape and check the progress of the next step. In the case of a call process, the system will check the progress of the step after the source process and source route.
Synchronization	After executing para start, check the progress of the next step. In the case of para end and route end, the system will check the progress of the step after the para end of the merging route when all synchronization routes are complete. If not all are complete, the system will stop its own route and check the progress of the next route.
Selection	After executing the selection, the system will check the progress of the next step. When the conditions for cell wait hold, the system will stop other selected routes and check the progress of the next step. When they do not hold, the system will check the progress of the next route. Select end and route end check the progress of the step after the select end of the merging route. At that time, the system will start if the merging route is stopped.

Item	Description
Wait for the conditions	If the conditions hold, the wait makes the system go to the next step and conduct a progress check. If the conditions do not hold, the system will check the progress of the route of the next step. In the next scan, the progress check of that process and route will be conducted with this step first. If it is with precondition clear, and if the preceding step is an ON statement before the system goes to the next step, the system will clear it to 0.
Figure without delay	This displays the names of the figures that go to the next step immediately in any case. The process start, route start, para start, select, multi-entry, box, control box, function, process end at the end of the call process, route end at the end of synchronization, para end, route end at selective merging, select end, and branching relationship (repeat start, repeat end, if, and jump) will go to the step where they should go, without a scan delay.
Non-synchronous merging	If a para start is made, then a check must be made to see if the next step is still in progress. If a given route other than non-synchronous routes has already reached its route end at the time the main route reaches its non-synchronous process end, the process start must be performed in the next scan again. Then, a check must be made to see if the next step is still in progress. If a given non-synchronous route is still on the way to its route end at the time the process start is to be performed again, then the process start need not be performed, but instead a check must be made to see if the next step is still in progress.

Supplement 5 HI-FLOW Program and CPU Load

A HI-FLOW program runs as part of the OS on the CMU module of the PCs. Increasing the number of HI-FLOW programs therefore increases the OS load on the CMU module of the PCs. Overloading will stop the sequence cycle or cause any other malfunction in the system in general. The below explains how to create a HI-FLOW program effectively and shows a guide for load judgment.

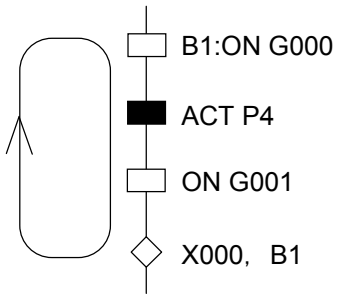
<How to create a HI-FLOW program effectively>

1. The size of the load of the HI-FLOW program depends on the number of routes being executed.

   This does not relate to the vertical (route) length of the HI-FLOW program. In consequence, a program divided into too many processes and routes for executing everything has a heavy load.

2. Be on guard against superfluous loops.

   Be on guard against loops that are unnecessary and have no stops.



The program in the chart shown on the left continues to execute steps in the loop and increases the load until X000 is turned off.

3. Use timer numbers without skipping any of them in ascending order.

   The smaller the timer number, the lighter the load from the timer (wait timer, parallel timer, and counter).

4. The wait timer uses the same number in the same route.

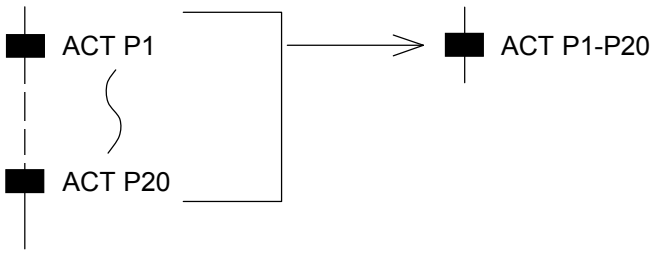
   Wait timers on the same route are never executed simultaneously. Assign an identical timer number and avoid using the rear timer number whenever possible.

5. Minimize the call process.

   Creating a program containing subroutines will make it easier to understand. During execution, however, the load will be heavier than when it is not turned into a call process. When structuralizing a program, give good consideration.

6. Avoid using consecutive control boxes.

   Avoid the continuous use of the execution load of the control box whenever possible, because it is considerably heavy. If usage is absolutely necessary to, use a continuous specification of processes effectively.



7. Minimize the setting of the system control bit.

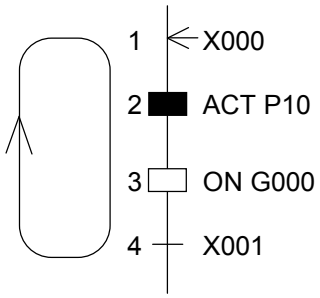
   The system control bit needs a check for each sequence cycle and for each step execution. It is then considerably loaded. Set a necessary minimum.

8. Minimize the use of multi-entry.

   The multi-entry step needs a check for each sequence. The larger the number of steps used, the heavier the load. Minimize their use.

9. Be on guard against in-loops of multi-entry.

For multi-entry, check the conditional expression for each sequence cycle. If it holds, begin with that step. However, if the conditions hold consecutively instead of in an edge form, an in-loop will occur. Set the conditions for multi-entry to edge trigger.



If X001 does not hold and X000 remains on in the left-hand program, the system will continue to execute 1 through 4 for each sequence. To avoid this, set X000 to the edge condition.

10. Minimize the use of STP and RST at process start.

STP and RST at process start are considerably loaded in order to check the conditions for each sequence cycle. Set these to the necessary minimum.

11. Be on guard against the CLR setting of process start.

CLR of process start is heavily loaded in order to clear the PI/O every time the conditions hold. (RST, STP, and ACT will not check the conditions once the conditions hold.) Create check conditions for CLR with care.

12. Avoid using applied instructions consecutively.

Applied instructions perform operations without stoppage. Describing them consecutively may therefore extend their sequence cycle. Create them with sufficient care.

13. Avoid complex conditional expressions whenever possible.

Using a complicated conditional expression in HI-FLOW will cause expression analysis to take a considerable amount of time. Complicated conditions will become lighter in load when handed over to HI-FLOW after being received by the ladder.

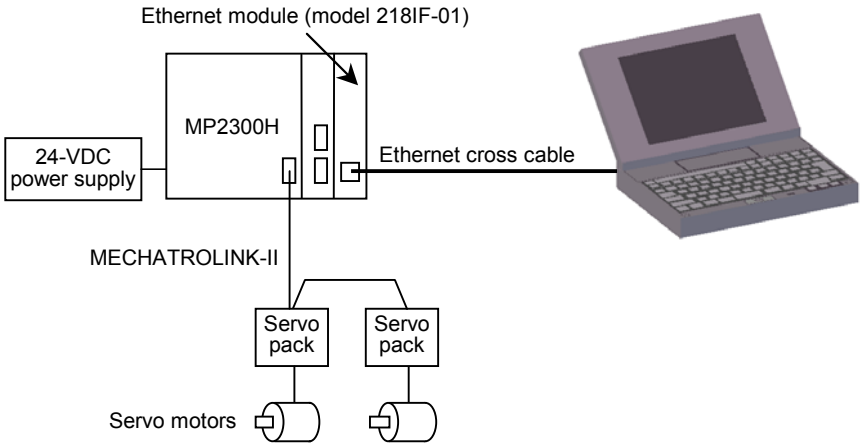
Supplement 6 MP2300H System Reconfiguration Procedure

When adding an optional module or servo pack to the MP2300H motion controller or replacing them with new ones, you have to update the motion controller's system configuration information by using its self-configuration function. Self-configurations using this function are classified into two types: the module self-configuration, which is done when a module is added or replaced, and the all-module self-configuration, which is done when configuring the MP2300H controller anew or for the first time. This supplement describes the procedures used for the two types of self-configurations.

1. Procedure used for module self-configuration

(1) Ensure that the MP2300H controller is in power-off state, and connect a motion module, servo pack, and a servo motor together.

(2) Connect the personal computer and the MP2300H controller by Ethernet cross cable, as shown below.



(3) Set the following switches of the Ethernet module (of model 218IF-01) according to the information given in the table below, and power up the MP2300H controller.

■ Ethernet module switch settings

Switch name	Setting	Operation mode
INIT	OFF→ON	The module starts with the default IP address (192.168.1.1).
TEST	OFF	

(4) Choose “Network and Dial-Up Connection” from the Control Panel in the Windows operating system and then set the IP address and subnet mask for the personal computer according to the information given in the table below.

Item	Setting	Remarks
IP address	192.168.1.xx	The xx portion of this IP address must be a value in the range 2 to 254.
Subnet mask	255.255.255.0	

(5) By using “Ping” of the Windows operating system, check if the connection between the personal computer and MP2300H controller is functioning normally. The steps required for this are as follows:

■ Connection check using “Ping”

Choose [Start] - [Programs] - [Accessories] - [Command Prompt] from the Start menu of the Windows operating system. The Command Prompt then starts. Following the prompt, enter the Ping command, as shown below. A basic communication test will then be performed by Ping between the personal computer and Ethernet module.
The following shows the details of the Ping command entered.

```
C:\WINDOWS> ping 192.168.1.1
```

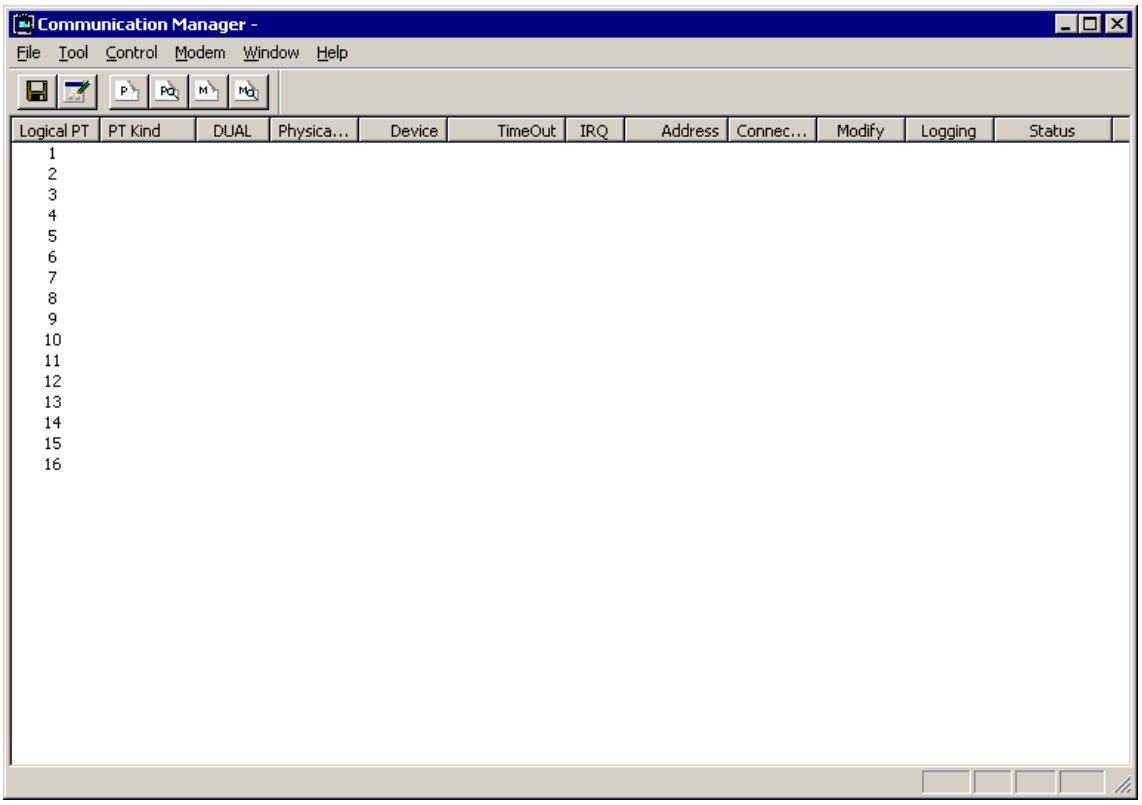
If the Ethernet module is connected properly, the following message appears on-screen:

```
Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time=4ms TTL=254
Reply from 192.168.1.1: bytes=32 time=3ms TTL=254
Reply from 192.168.1.1: bytes=32 time=3ms TTL=254
Reply from 192.168.1.1: bytes=32 time=3ms TTL=254
C:\WINDOWS>
```

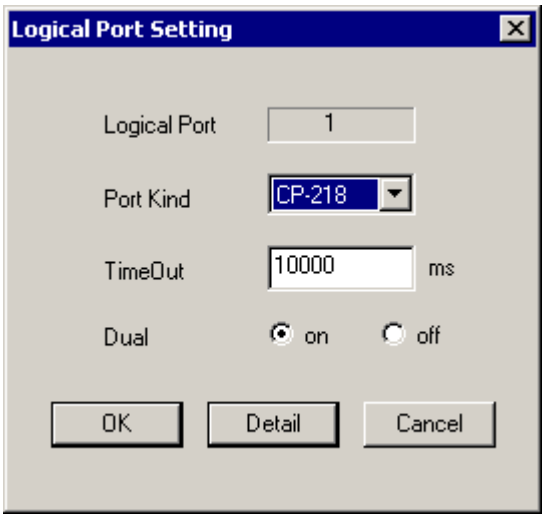
If it is connected improperly, the following timeout message appears instead.

```
Pinging 192.168.1.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
C:\WINDOWS>
```

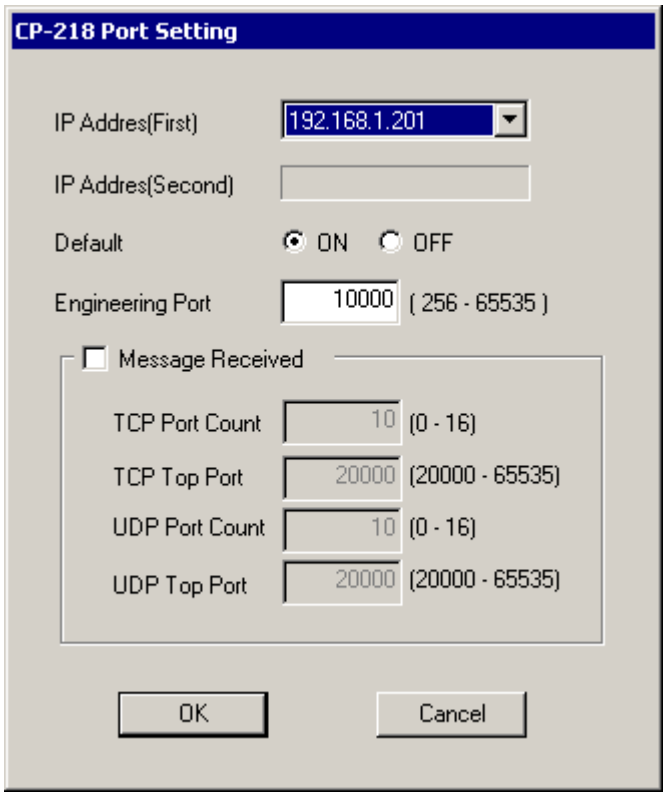
(6) Choose [Start] - [Programs] - [YE_Applications] - [Communication Manager] from the Start menu of the Windows operating system. The Communication Manager then starts.



(7) Double-click on the number "1" under "Logical PT" in the list. The "Logical Port Setting" window as shown below will then appear.



(8) Specify “CP-218” in the “Port Kind” box and click the Detail button. The “CP-218 Port Setting” window as shown below will then appear.

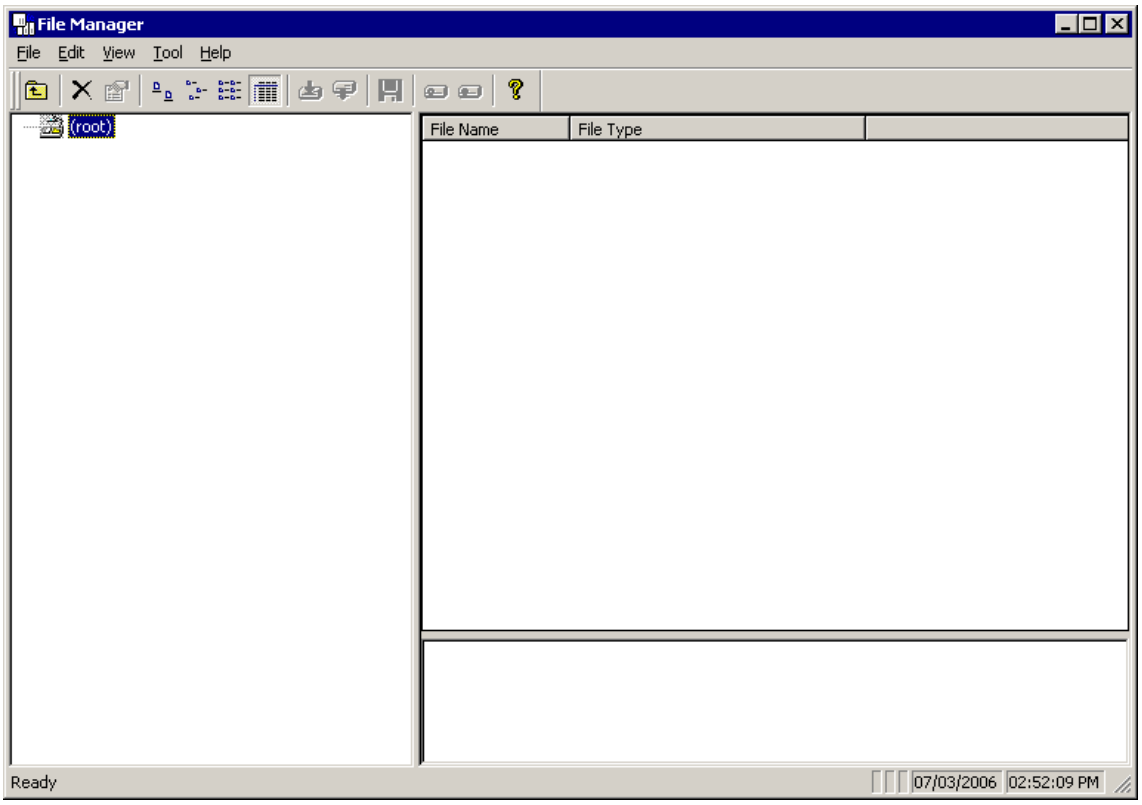


(9) Specify the IP address of the personal computer in the “IP Address(First)” box and click the OK button.

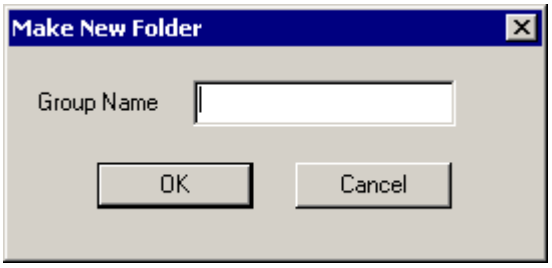
(10) The “Logical Port Setting” window is active again. Click the OK button in this window.

(11) Choose [File] - [Exit] from the Communication Manager’s menu to exit the Communication Manager.

(12) Choose [Start] - [Programs] - [YE_Applications] - [MPE720] from the Start menu on the Windows operating system. The File Manager will then starts.

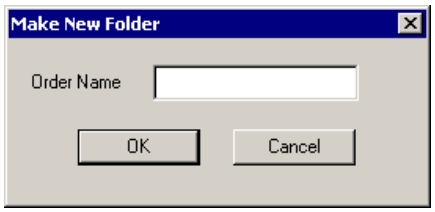


(13) Right-click on "(root)" in the tree display pane of the File Manager window. From the displayed menu, choose [New] - [Group Folder]. The "Make New Folder" window as shown below will then appear.



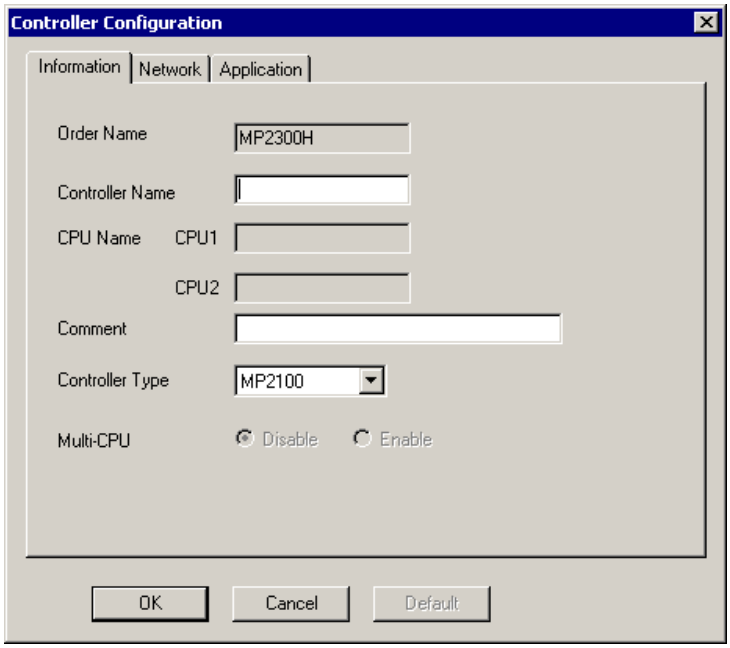
In this window, specify the desired group name (e.g., PLC) of a group you want to create, and then click the OK button.

(14) The specified group is created in the tree display pane. Right-click the group name of the created group. From the displayed menu, choose [New] - [Order Folder]. The “Make New Folder” window as shown below will then appear.



In this window, specify the desired order name (e.g., MP2300H) of an order you want to create., and then click the OK button.

(15) The specified order folder is created in the tree display pane. Right-click the order name of the created order folder. From the displayed menu, choose [Create New Folder] - [Controller Folder]. The “Controller Configuration” window as shown below will then appear.



In this window, specify the items listed below and click the OK button.

■ Information

Controller name: Arbitrarily specified name (e.g., sample)

Controller type: MP2300

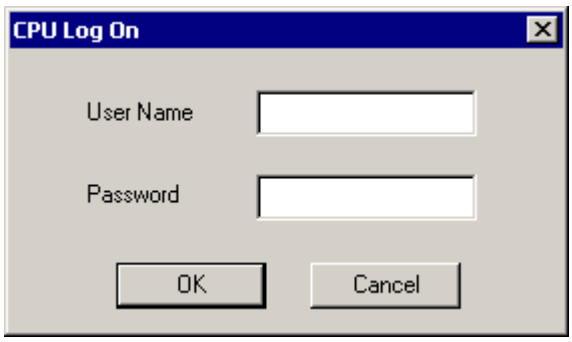
■ Network

Online: Yes

Logical Port No.: 1: CP-218

IP Address: 192.168.1.1 (The above IP address is the MP2300H controller’s.)

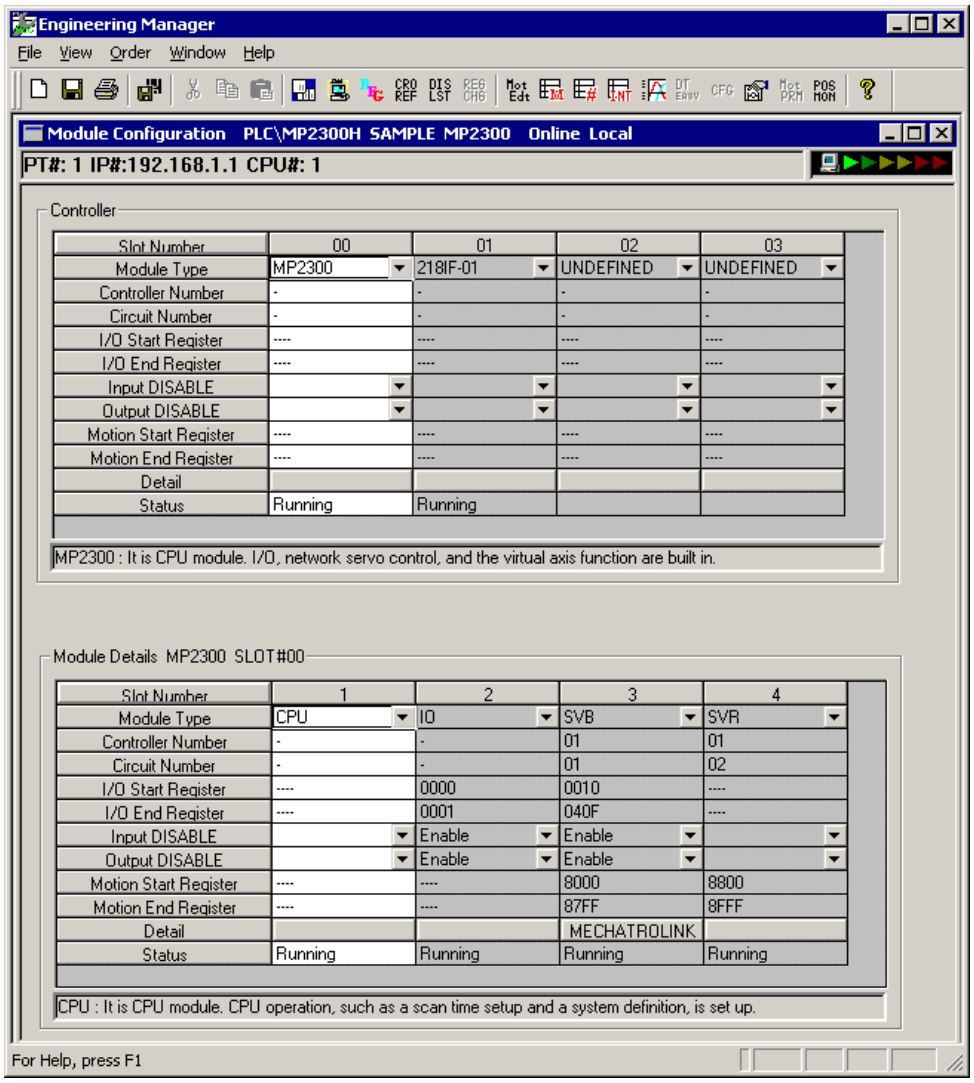
(16) The specified PLC folder is created in the tree display pane. Right-click the created PLC folder in the pane. In the displayed menu, ensure that “Online” is checked, and then click “Log On”. The “CPU Log On” window as shown below will then appear.



(17) In the “CPU Log On” window, specify the desired user name and password, and click the OK button.

With the MP2300H controller, the user name and password are both defaulted to “USER-A”.

(18) Double-click [Definition Folder] - [Module Configuration] in the tree display pane. The "Module Configuration" window as shown below will then appear.

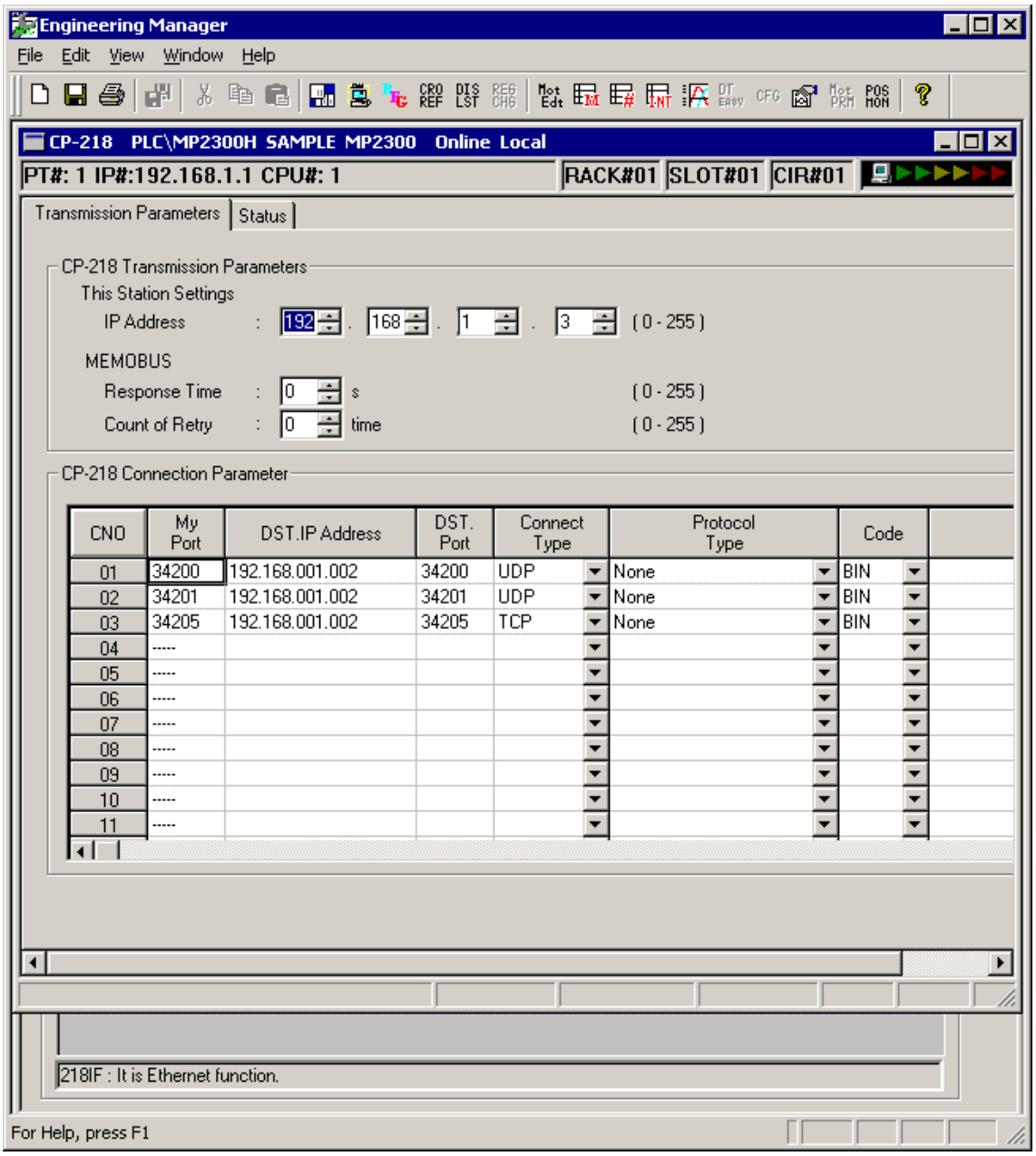


(19) Right-click the desired slot number for a module you want to reconfigure in the "Controller" group of the "Module Configuration" window. From the displayed menu, select [Module Self Configuration]. Self-configuration will then start.

- If you want to set or change the IP address of the Ethernet module, perform Steps (20) through (23).
- If the Ethernet module is self-configured, its communication settings will be automatically initialized. So, set the communication settings again by performing Steps (20) through (23).
- In any other case, proceed to Step (24).

(20) Click the slot number for the module “218IF-01” in the “Controller” group list of the “Module Configuration” window.

(21) The “Module Details” group display is changed to a different one. Double-click the slot number for the module “218IF”. The “CP-218” window as shown below will then appear.



(22) In this window, specify the items listed below in the "Transmission Parameters" pane, and choose [File] - [Save] from the menu bar of the Engineering Manager window.

■ CP-218 Transmission Parameters - This Station Settings

IP Address: The IP address of the Ethernet module (e.g., 192.168.1.3)

■ CP-218 Connection Parameter

CNO	My Port	DST. IP Address	DST. Port	Connect Type	Protocol Type	Code
01	34200	(IP address of ET.NET)	34200	UDP	None	BIN
02	34201	(IP address of ET.NET)	34201	UDP	None	BIN
03	34205	(IP address of ET.NET)	34205	TCP	None	BIN

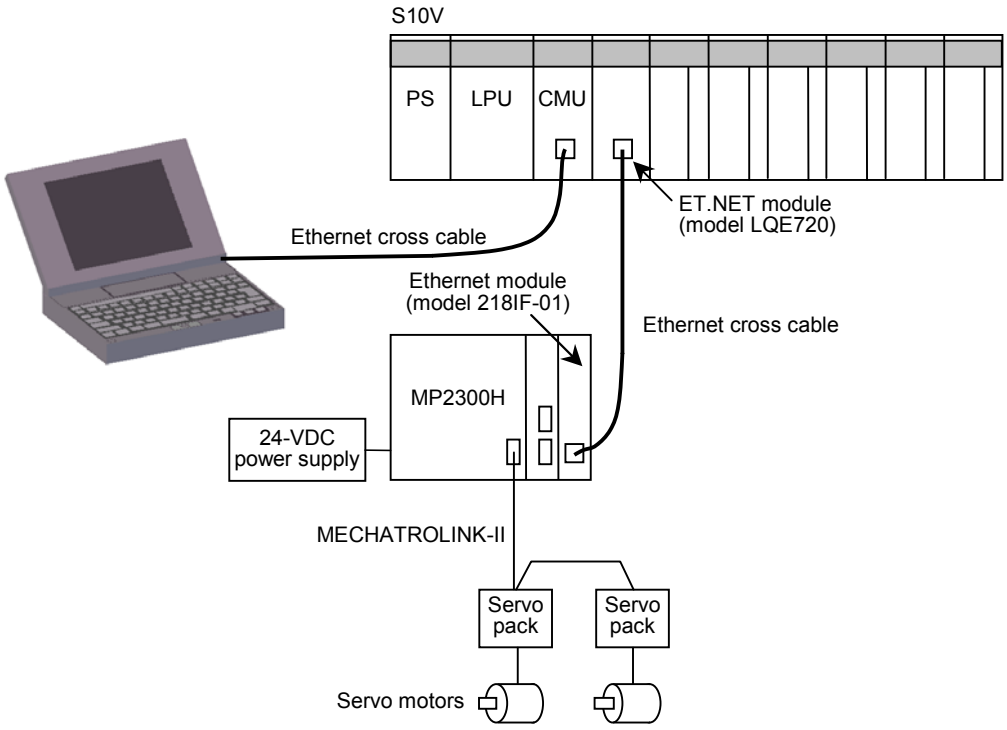
(23) Choose [Window] - [Module Configuration] from the menu bar of the Engineering Manager window. The Module Configuration window will then appear.

(24) When the self-configuration is completed, choose [File] - [Save & Save into flash memory] from the menu bar of the Engineering Manager window. A confirmation message for saving to flash memory will then appear. Click the [Yes] button. Then, another confirmation message for stopping the CPU will appear. Click [Yes]. Saving to flash memory will then start.

(25) When the saving is completed, a confirmation message for switching the CPU status to RUN will appear. Click the [No] button.

(26) Close the Engineering Manager window and go back to the File Manager window. Then, right-click the PLC folder you have created in Step (15), and choose [Log off] from the displayed menu.

(27) Power down the MP2300H controller and set all the switches of the MP2300H controller and Ethernet module in OFF position if any of them are in ON position. Then, connect the personal computer, S10V controller, and the MP2300H controller together by Ethernet cross cable, as shown below.



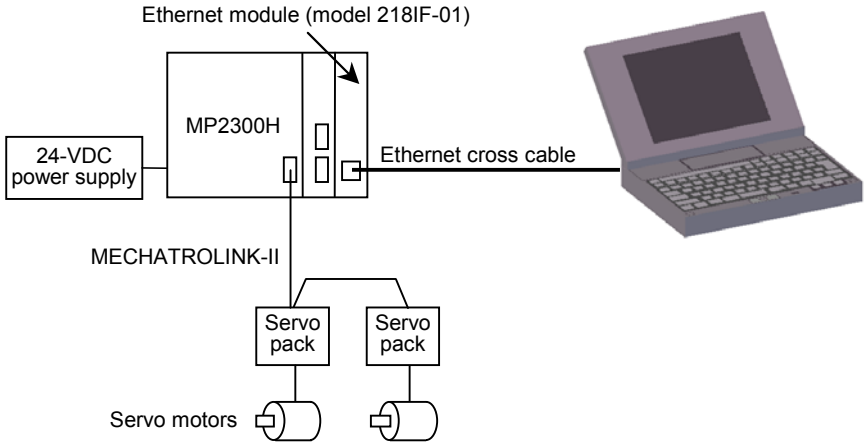
When the connection work is complete, power up the MP2300H controller and S10V controller, in that order.

The MP2300H system reconfiguration procedure is completed.

2. Procedure used for all-module self-configuration

(1) Ensure that the MP2300H controller is in power-off state, and connect a motion module, servo pack, and a servo motor together.

(2) Connect the personal computer and the MP2300H controller by Ethernet cross cable, as shown below.



(3) Set the following switches of the MP2300H controller and Ethernet module (of model 218IF-01) according to the information given in the table below, and power up the MP2300H controller.

■ MP2300H controller switch settings

Switch name	Setting	Operation mode
STOP	OFF	
SUP	OFF	
INIT	OFF→ON	The controller starts with its default settings.
CNFG	OFF→ON	The controller performs self-configuration for the equipment connected to it.
MON	OFF	
TEST	OFF	

■ Ethernet module switch settings

Switch name	Setting	Operation mode
INIT	OFF→ON	The module starts with the default IP address (192.168.1.1).
TEST	OFF	

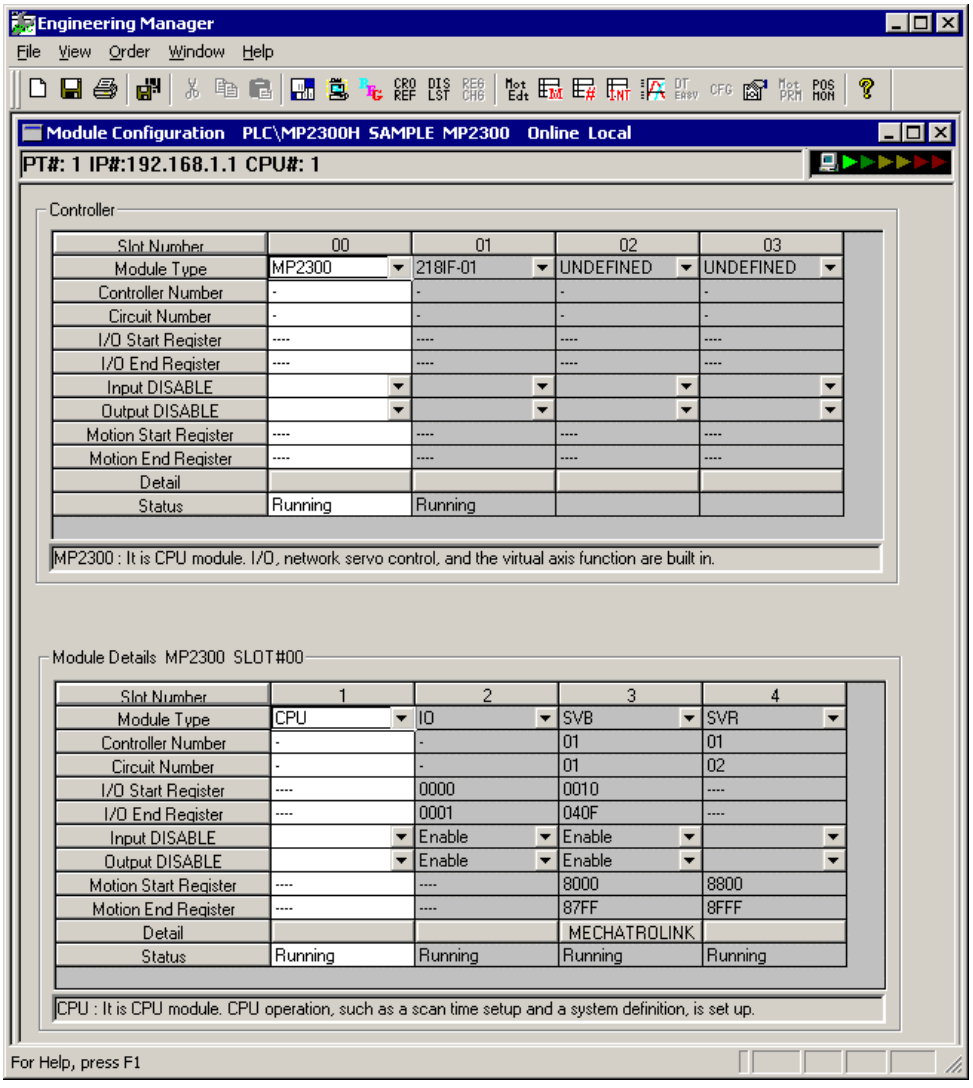
(4) When the MP2300H controller is powered up, all-module self-configuration will start. During the progress of the self-configuration, the RUN indicator (LED) continues blinking. When the blinking is stopped, wait until the RDY indicator (LED) is lit.

(5) Choose “Network and Dial-Up Connection” from the Control Panel on the Windows operating system and then set the IP address and subnet mask for the personal computer according to the information given in the table below.

Item	Setting	Remarks
IP address	192.168.1.xx	The xx portion of this IP address must be a value in the range 2 to 254.
Subnet mask	255.255.255.0	

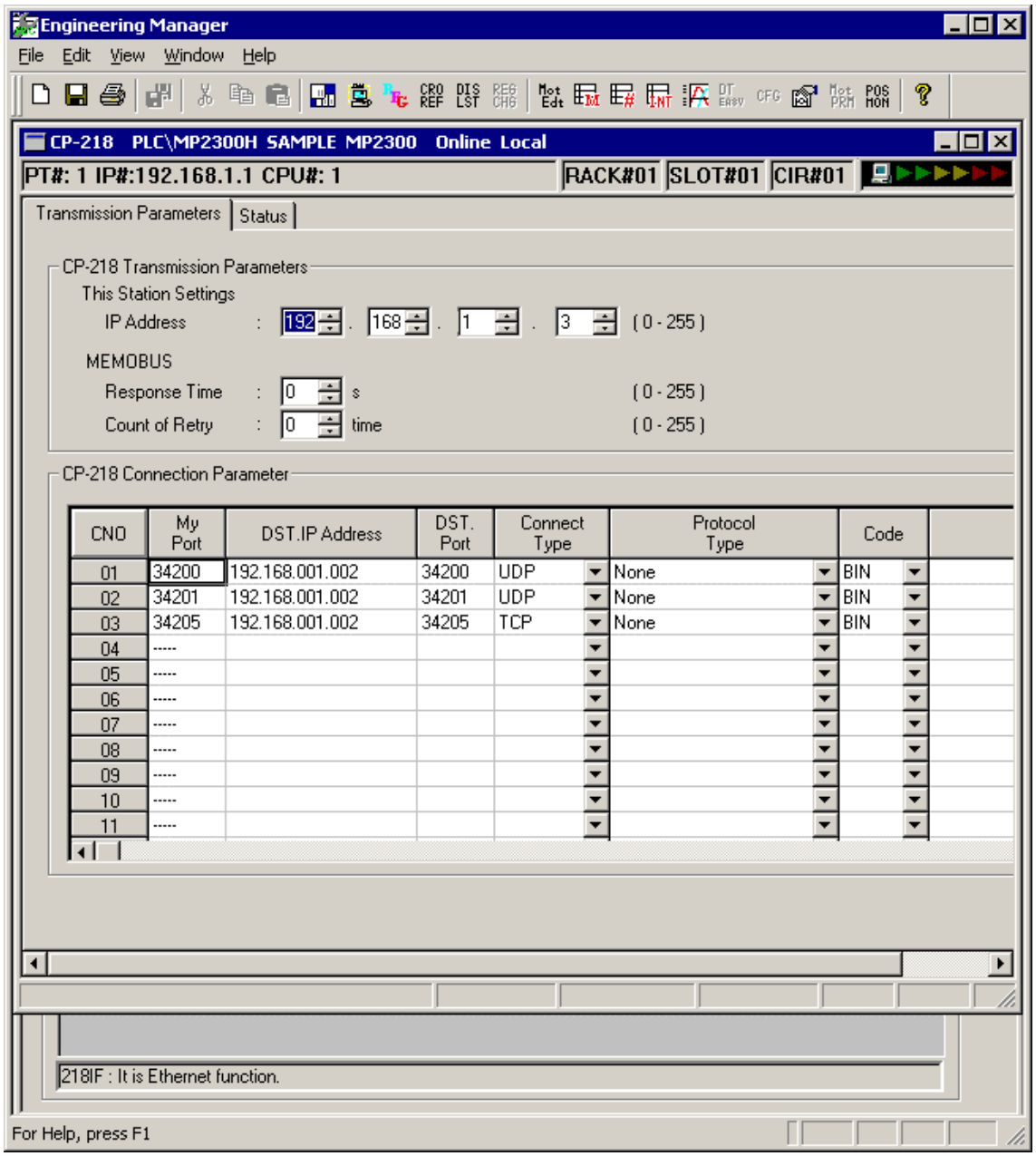
(6) By using “Ping” of the Windows operating system, check if the connection between the personal computer and MP2300H controller is functioning normally. For details, see Step (5) in “1. Procedure used for module self-configuration.”

(7) By performing the same steps as Steps (6) through (18) in “1. Procedure used for module self-configuration”, display the “Module Configuration” window on-screen.



(8) Click the slot number for the module “218IF-01” in the “Controller” group list of the “Module Configuration” window.

(9) The “Module Details” group display is changed to a different one. Double-click the slot number for the module “218IF”. The “CP-218” window as shown below will then appear.



(10) In this window, specify the items listed below in the "Transmission Parameters" pane, and choose [File] - [Save] from the menu bar of the Engineering Manager window.

■ CP-218 Transmission Parameters - This Station Settings

IP Address: The IP address of the Ethernet module (e.g., 192.168.1.3)

■ CP-218 Connection Parameter

CNO	My Port	DST. IP Address	DST. Port	Connect Type	Protocol Type	Code
01	34200	(IP address of ET.NET)	34200	UDP	None	BIN
02	34201	(IP address of ET.NET)	34201	UDP	None	BIN
03	34205	(IP address of ET.NET)	34205	TCP	None	BIN

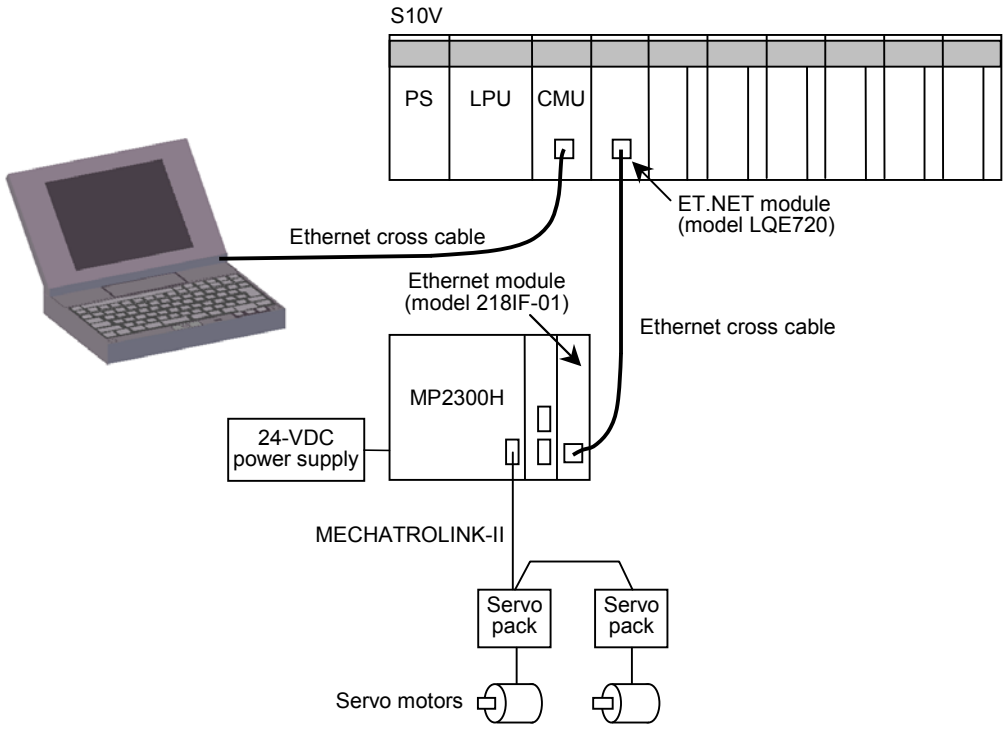
(11) Choose [Window] - [Module Configuration] from the menu bar of the Engineering Manager window. The Module Configuration window will then appear.

(12) When the self-configuration is completed, choose [File] - [Save & Save into flash memory] from the menu bar of the Engineering Manager window. A confirmation message for saving to flash memory will then appear. Click the Yes button. Then, another confirmation message for stopping the CPU will appear. Click Yes. Saving to flash memory will then start.

(13) When the saving is completed, a confirmation message for switching the CPU status to RUN will appear. Click the No button.

(14) Close the Engineering Manager window and go back to the File Manager window. Then, right-click the PLC folder you have created by performing Step (15) of "1. Procedure used for module self-configuration," and choose [Log off] from the displayed menu.

(15) Power down the MP2300H controller and set all the switches of the MP2300H controller and Ethernet module in OFF position if any of them are in ON position. Then, connect the personal computer, S10V controller, and the MP2300H controller together by Ethernet cross cable, as shown below.



When the connection work is complete, power up the MP2300H controller and S10V controller, in that order.

The MP2300H system reconfiguration procedure is completed.