

HITACHI

Software Manual

Programming

HI-FLOW for Windows®

S10VE

SEE-3-122 (A)

Software Manual

Programming

HI-FLOW for Windows®

SIOVE

First Edition, May 2020, SEE-3-122 (A)

All Rights Reserved, Copyright © 2020, Hitachi, Ltd.

The contents of this publication may be revised without prior notice.

No part of this publication may be reproduced in any form or by any means without permission in writing from the publisher.

Printed in Japan.

IC (FL-MW2007)

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 contains information on potential hazards that is intended as a guide for safe use of this product. The potential hazards listed in the manual are divided into four hazard levels of danger, warning, caution, and notice, according to the level of their severity. The following are definitions of the safety labels containing the corresponding signal words DANGER, WARNING, CAUTION, and NOTICE.



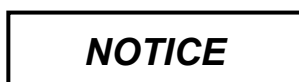
: This safety label identifies precautions that, if not heeded, will result in death or serious injury.





: Identifies precautions that, if not heeded, could result in death or serious injury.



: Identifies precautions that, if not heeded, could result in minor or moderate injury.



: This safety label without a safety alert symbol identifies precautions that, if not heeded, could result in property damage or loss not related to personal injury.

Failure to observe any of the  **CAUTION** and  **NOTICE** statements used in this manual could also lead to a serious consequence, depending on the situation in which this product is used. Therefore, be sure to observe all of those statements without fail.

The following are definitions of the phrases “serious injury,” “minor or moderate injury,” and “property damage or loss not related to personal injury” used in the above definitions of the safety labels.

Serious injury: Is an injury that requires hospitalization for medical treatment, has aftereffects, and/or requires long-term follow-up care. Examples of serious injuries are as follows: vision loss, burn (caused by dry heat or extreme cold), electric-shock injury, broken bone, poisoning, etc.

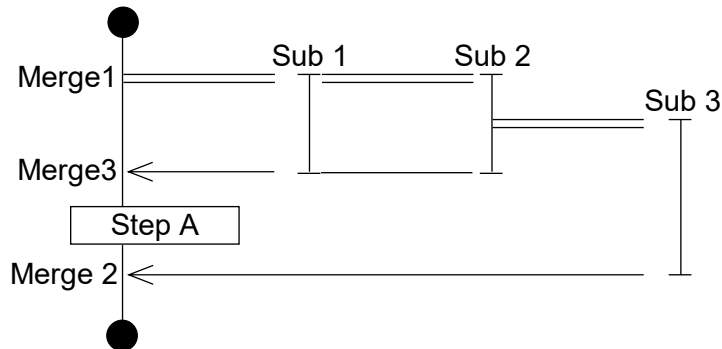
Minor or moderate injury: Is an injury that does not require either hospitalization for medical treatment or long-term follow-up care. Examples of minor or moderate injuries are as follows: burn, electric-shock injury, etc.

Property damage or loss not related to personal injury: Is a damage to or loss of personal property. Examples of property damages or losses not related to personal injury are as follows: damage to this product or other equipment or their breakdown, loss of useful data, etc.

The safety precautions stated in this manual are based on the general rules of safety applicable to this product. These safety precautions are a necessary complement to the various safety measures included in this product. Although they have been planned carefully, the safety precautions posted on this product and in the manual do not cover every possible hazard. Common sense and caution must be used when operating this product. For safe operation and maintenance of this product, establish your own safety rules and regulations according to your unique needs. A variety of industry standards are available to establish such safety rules and regulations.

NOTICE

- In the syntax that makes para start and merges at select end (as in the pattern shown below), do not describe any syntax that makes para start again in the middle of a subroute and merges at select end into the main route, not the subroute. If a merging position is reached, the system will shut down only the route having the same merging position. Therefore, non-merging subroutes that are being executed will remain as they are.

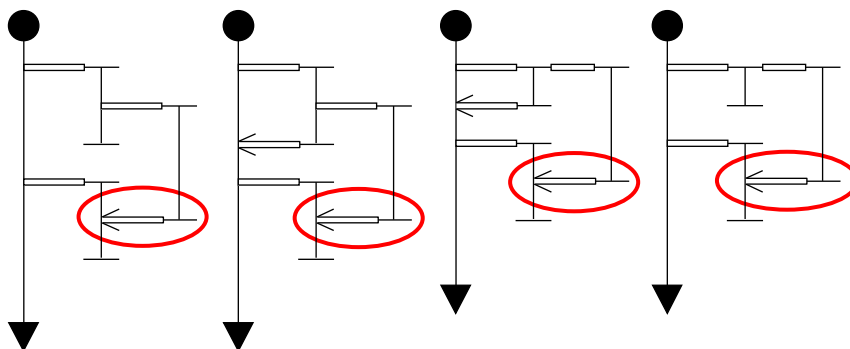


[Description of operation]

- During execution of sub 1 to sub 3, if sub 3 reaches merge 2 before sub 1 and sub 2 reach merge 3, the execution points of sub 1 and sub 2 will not be shut down.

(See page 4-28.)

- Do not describe any syntax that has differing non-synchronous processes between branching and merging, and that has a non-synchronous route as a merging route (as in the pattern shown below). If execution is attempted, a new main route will not be started because the system judges that a subroute is being executed.



- 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.

(See page 4-35.)

Revision History

Revision No.	History (revision details)	Issue date	Remarks
A	First edition	May 2020	

PREFACE

HI-FLOW, a programming language for Hitachi Programmable Controllers, is a language that is programmable in a flowchart form.

This manual describes instruction words designed to develop programs with HI-FLOW.

<Trademarks>

- Microsoft® and Windows® are either registered trademarks or 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>

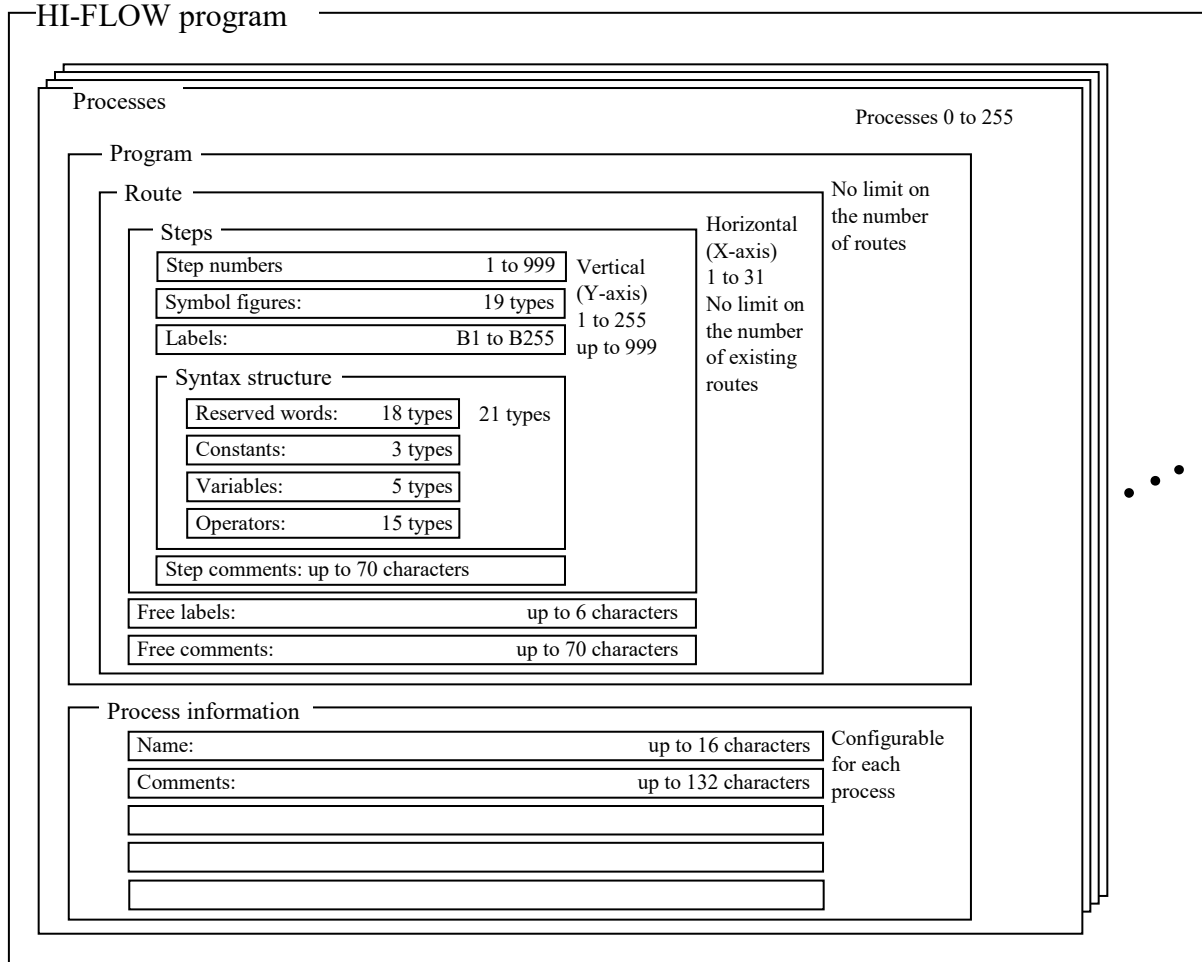
- Memory capacities and requirements, file sizes and storage requirements, etc. must be calculated according to the formula 2^n . The following examples show the results of such calculations by 2^n (to the right of the equals signs).
 - 1 KB (kilobyte) = 1,024 bytes
 - 1 MB (megabyte) = 1,048,576 bytes
 - 1 GB (gigabyte) = 1,073,741,824 bytes
 - 1 TB (terabyte) = 1,099,511,627,776 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
 - 1 TB (terabyte) = $1,000^4$ bytes

CONTENTS

CHAPTER 1	COMPOSITION OF THE HI-FLOW PROGRAM	1-1
CHAPTER 2	HOW TO USE THIS MANUAL	2-1
2.1	Overview	2-1
2.2	Description of Syntax	2-2
2.3	Description of Applied Instructions	2-3
CHAPTER 3	PROCESS	3-1
3.1	What is a Process?	3-1
3.2	Program	3-6
3.3	Process Information	3-20
CHAPTER 4	DESCRIPTION OF SYNTAX	4-1
4.1	Process Start and Process End	4-1
4.2	Route Start and Route End	4-5
4.3	Wait	4-6
4.4	Boxes	4-8
4.5	Control Box	4-15
4.6	Repeat Start and Repeat End	4-19
4.7	If	4-20
4.8	Jump	4-22
4.9	Escape	4-23
4.10	Para Start and Para End	4-24
4.11	Para Start and Select End	4-25
4.12	Select, Cell Wait, and Select End	4-29
4.13	Multi-entry	4-30
4.14	Call	4-31
4.15	Function	4-32
4.16	Wait with Precondition	4-32
4.17	Non-synchronous Process End	4-33
CHAPTER 5	APPLIED INSTRUCTIONS	5-1
5.1	Overview	5-1
5.2	How to Use It	5-1
5.3	Parameters	5-1
5.4	Type Conversion in Operations	5-3
5.5	System Error Flags	5-4
5.6	Function Description	5-5
SUPPLEMENT A	FLOW OF THE HI-FLOW PROGRAM	Z-1
SUPPLEMENT B	PCs MEMORY	Z-2
SUPPLEMENT C	ONLINE MODE	Z-3
SUPPLEMENT D	PROGRESS CHECK	Z-7
SUPPLEMENT E	HI-FLOW PROGRAM AND CPU LOAD	Z-9

CHAPTER 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:



This Page Intentionally Left Blank

CHAPTER 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	3-1
Program	Section 3.2	3-6
• Routes		3-6
• Steps		3-11
• Step number		3-12
• Symbol figures		3-12
• Labels		3-15
• Syntax structures		3-15
• Reserved words		3-16
• Constants		3-16
• Variables		3-16
• Operators		3-18
• Step comments		3-18
• Free labels		3-19
• Free comments		3-19
Process information	Section 3.3	3-20
• Names		3-20
• Comments		3-20

2. HOW TO USE THIS MANUAL

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	4-1
Process start and process end	●	Section 4.1	4-1
• STP	●		4-2
• RST			4-3
• CLR			4-3
• ACT			4-3
Route start and route end	⊥	Section 4.2	4-5
Wait		Section 4.3	4-6
• Conditional expression	+		4-6
• Timers			4-6
• Output bits			4-6
• Wait timer			4-6
Boxes		Section 4.4	4-8
• Assignment expression			4-8
• Special assignment expression			4-9
• ON statements	□		4-12
• OFF statements			4-12
• Parallel timers			4-13
• TUP			4-14
• TRS			4-14
Control box		Section 4.5	4-15
• ACT	■		4-15
• RST			4-16
• STP			4-16
• CLR			4-17
Repeat start and repeat end	↕	Section 4.6	4-19
If	◇	Section 4.7	4-20
Jump	↳	Section 4.8	4-22
Escape	×	Section 4.9	4-23
Para start and para end	⏏	Section 4.10	4-24
Para start and select end	⏏	Section 4.11	4-25
Select, cell wait, and select end	⏏	Section 4.12	4-29
	≡≡≡		
Multi-entry	⏏	Section 4.13	4-30
Call	□	Section 4.14	4-31
Function	○	Section 4.15	4-32
Wait with precondition	+*	Section 4.16	4-32
Non-synchronous process end	↓	Section 4.17	4-33

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	$S+D \rightarrow R$	5-6
	Subtraction	SUB	$S-D \rightarrow R$	5-7
	+1	INC	$S+1 \rightarrow S$	5-8
	-1	DEC	$S-1 \rightarrow S$	5-9
	Multiplication	MUL	$S*D \rightarrow R$	5-10
	Division	DIV	$S/D \rightarrow R$	5-11
	Remainder	MOD	Remainder of $S/D \rightarrow R$	5-12
	Scale conversion	SCL	$S*D1/D2 \rightarrow R$	5-13
Logical operation instructions	Logical product	AND	$S \text{ AND } D \rightarrow R$	5-14
	Logical sum	OR	$S \text{ OR } D \rightarrow R$	5-15
	Exclusive OR	EOR	$S \text{ EOR } D \rightarrow R$	5-16
	Negation	NOT	$\text{NOT } S \rightarrow R$	5-17
Relational operation instructions	=	EQU	Truth/falsehood of $S = D \rightarrow R$	5-18
	⟨⟩	NEQ	Truth/falsehood of $S \langle \rangle D \rightarrow R$	5-19
	>	GT	Truth/falsehood of $S > D \rightarrow R$	5-20
	>=	GE	Truth/falsehood of $S \geq D \rightarrow R$	5-21
	<	LT	Truth/falsehood of $S < D \rightarrow R$	5-22
	<=	LE	Truth/falsehood of $S \leq D \rightarrow R$	5-23
	Test	TST	Code $S \rightarrow R$	5-24
Data transfer instructions	Transfer	MOV	$S \rightarrow D$	5-25
	Collective transfer	MOM	$S \sim S_n \rightarrow D \sim D_n$	5-26
	Replacement	EXC	$S \leftrightarrow D$	5-27
	FIFO write	PSH	$S \rightarrow D$ (FIFO table)	5-28
	FIFO read	POP	S (FIFO table) $\rightarrow D$	5-29
	Address set	AST	Address $S \rightarrow D$	5-30
	Search	SCH	$S = D(n) \rightarrow \text{Set } n \text{ to } R$	5-31
Data conversion instructions	BIN-BCD	BTD	$\text{BIN} \rightarrow \text{BCD}$ $S \text{ -----} \rightarrow R$	5-32
	BCD-BIN	DTB	$\text{BCD} \rightarrow \text{BIN}$ $S \text{ -----} \rightarrow R$	5-33
	BIN-7SEG	SEG	$\text{BIN} \rightarrow \text{7-segment}$ $S \text{ -----} \rightarrow R$	5-34

2. HOW TO USE THIS MANUAL

Category	Type	Symbol	Function overview	Page
Data conversion instructions	BIN-ASC	ASP	BIN \rightarrow ASCII (pack and unpack)	5-35
		ASU	S \rightarrow (R, R+1), (R, R+1, R+2, R+3)	5-36
	ASC-BIN	APB	ASCII (pack and unpack) \rightarrow BIN	5-37
		AUB	(S, S+1), (S, S+1, S+2, S+3) \rightarrow R	5-38
	Absolute value	ABS	S \rightarrow R	5-39
	+/-	NEG	-S \rightarrow R	5-40
	Decode	DCD	S $2^{11} \sim 2^{15}$ \rightarrow Turn ON the 2^n bit of R	5-41
Encode	ECD	First ON bit number of S \rightarrow $2^{11} \sim 2^{15}$ of R	5-42	
Shift instructions	Logic right-shift	LSR	S Logic right-shift D \rightarrow R	5-43
	Logic left-shift	LSL	S Logic left-shift D \rightarrow R	5-44
	Arithmetic right-shift	ASR	S Arithmetic right-shift D \rightarrow R	5-45
	Arithmetic left-shift	ASL	S Arithmetic left-shift D \rightarrow R	5-46
Rotation instructions	CW rotation	ROR	S CW rotation R	5-47
	CCW rotation	ROL	S CCW rotation R	5-48
Function processing instructions	Limiter	LIM		5-49
	Dead band	BND		5-50
	Dead zone	ZON		5-51
	Square root	ROT		5-52
	Maximum value	MAX		5-53
	Minimum value	MIN		5-54
Special instructions	Clear	XCLR YCLR GCLR RCLR KCLR TCLR UCLR CCLR VCLR ECLR FCLR JCLR QCLR HHCLR		5-55

CHAPTER 3 PROCESS

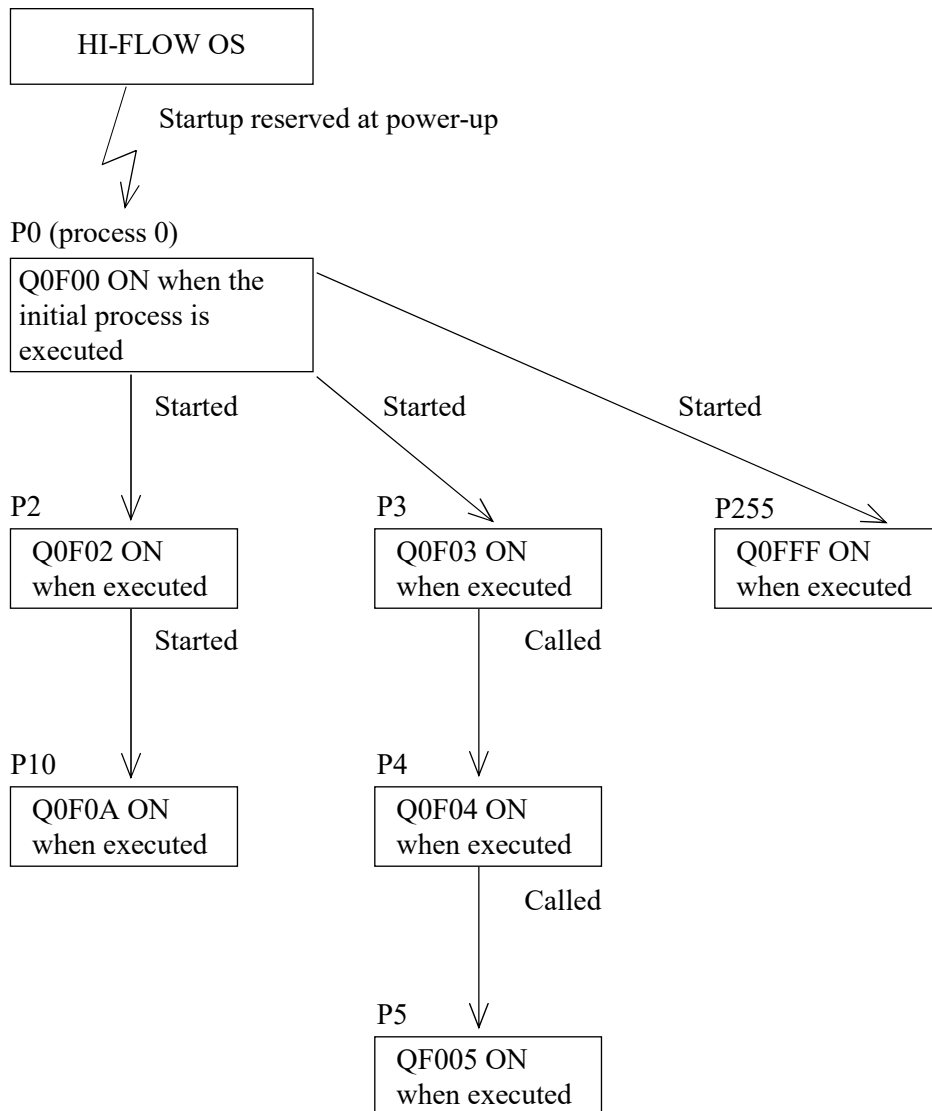
3.1 What is a Process?

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 “4.7.7 Laying out the system bits” of “*S10VE Software Manual Operation HI-FLOW for Windows*® (manual number SEE-3-132)” of standards Q0F00 through Q0FFF.)



3. PROCESS

Process statuses

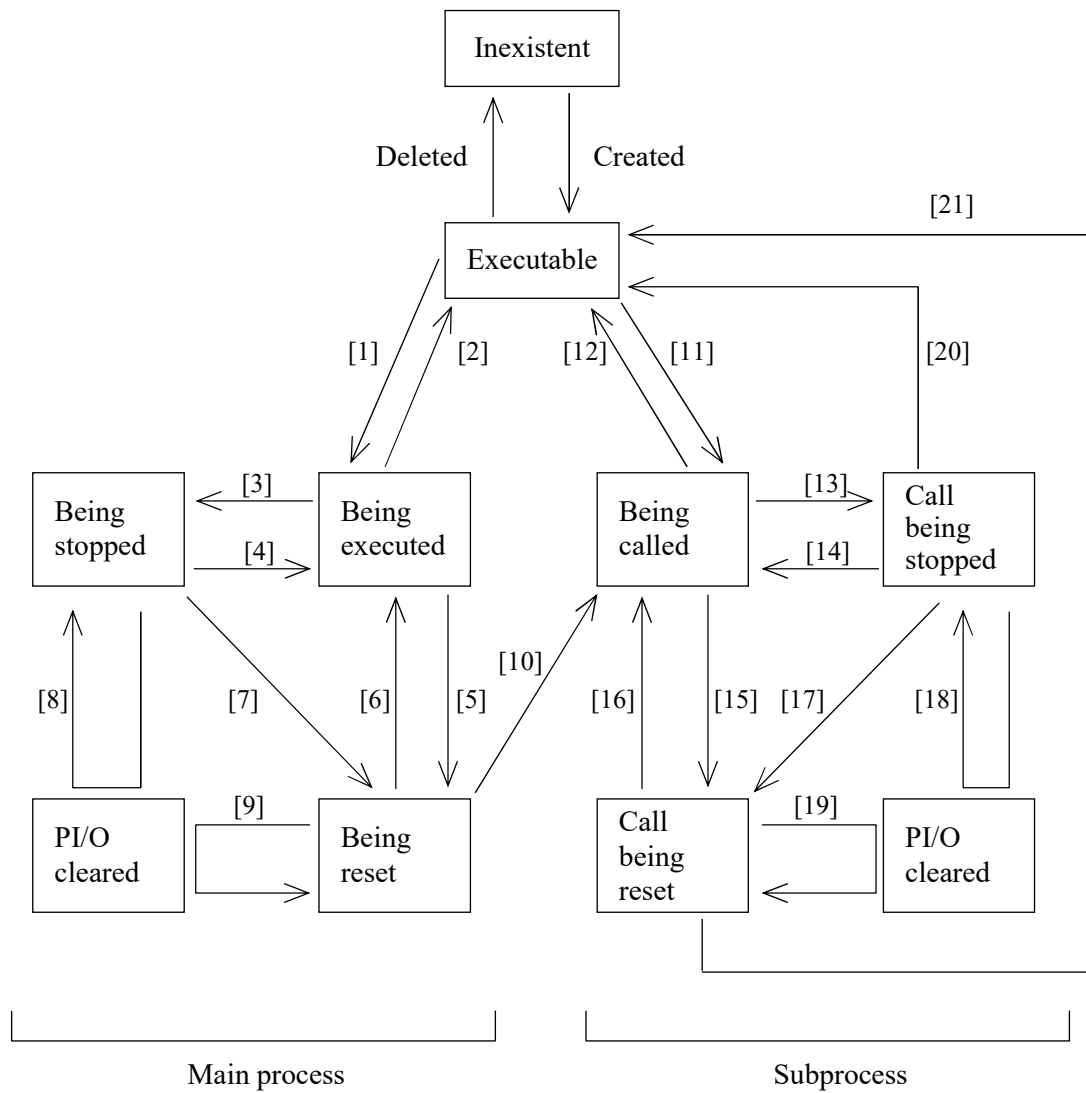
The PCs can be in nine different statuses.

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

3. PROCESS

- [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 (●)
- [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.

[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. PROCESS

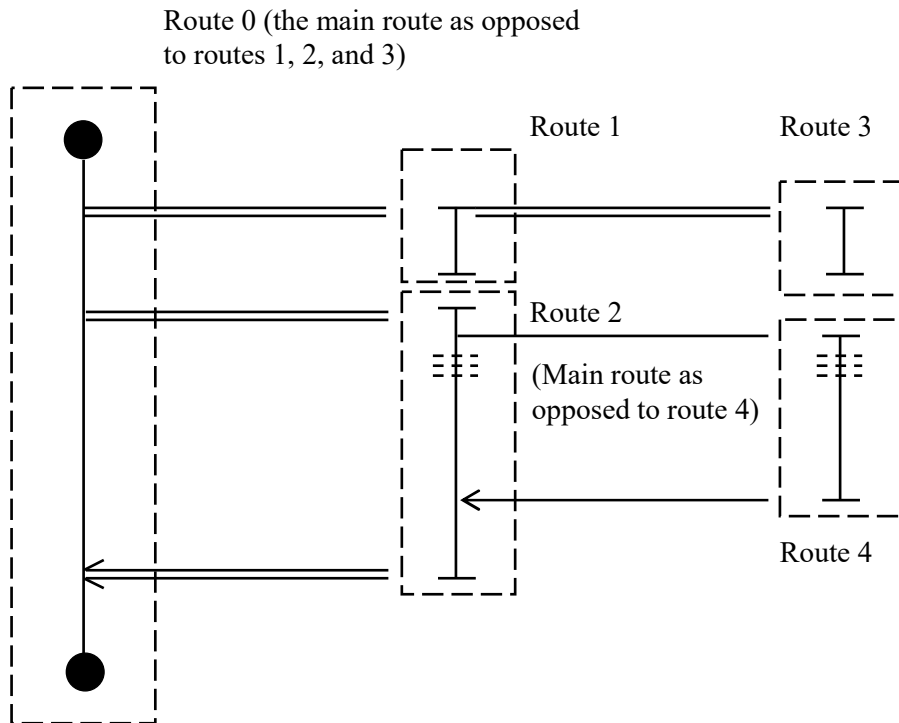
3.2 Program

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.

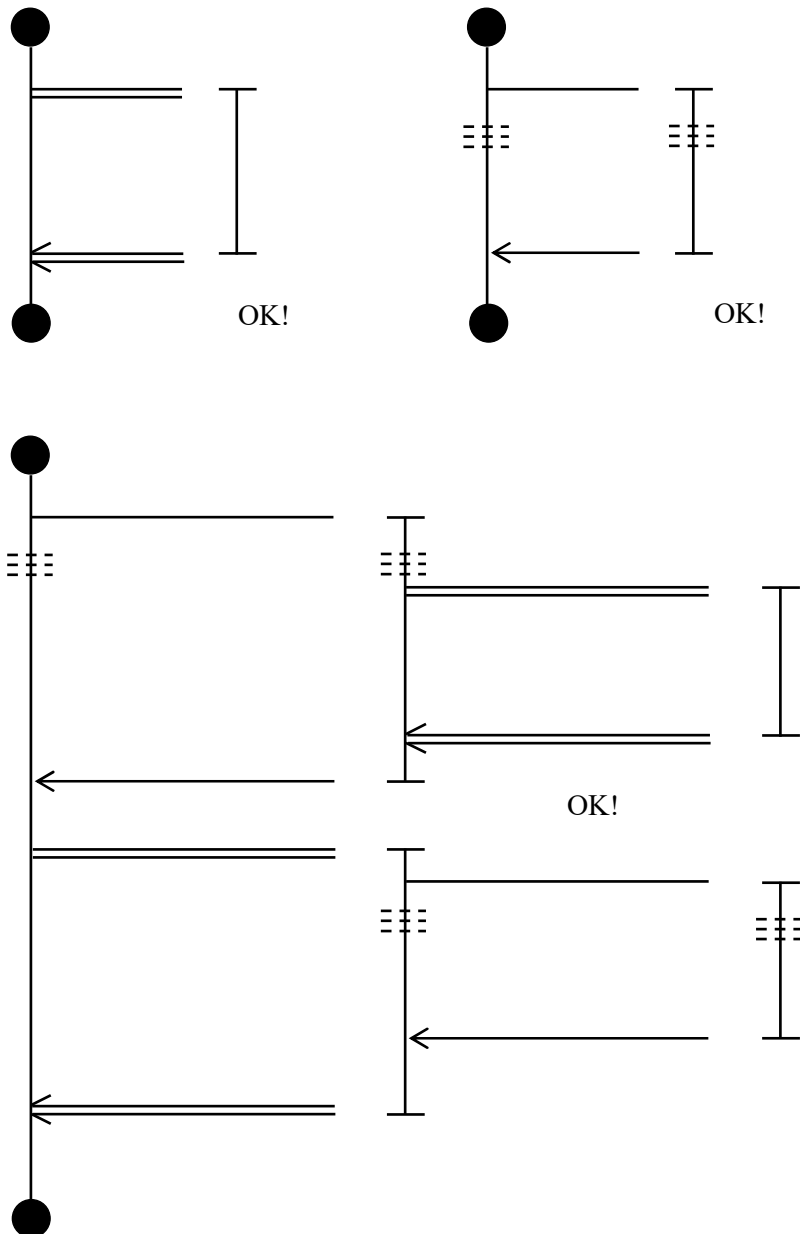


Synchronization routes do not necessarily need to merge. In that case, the source route will only start the route.

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.

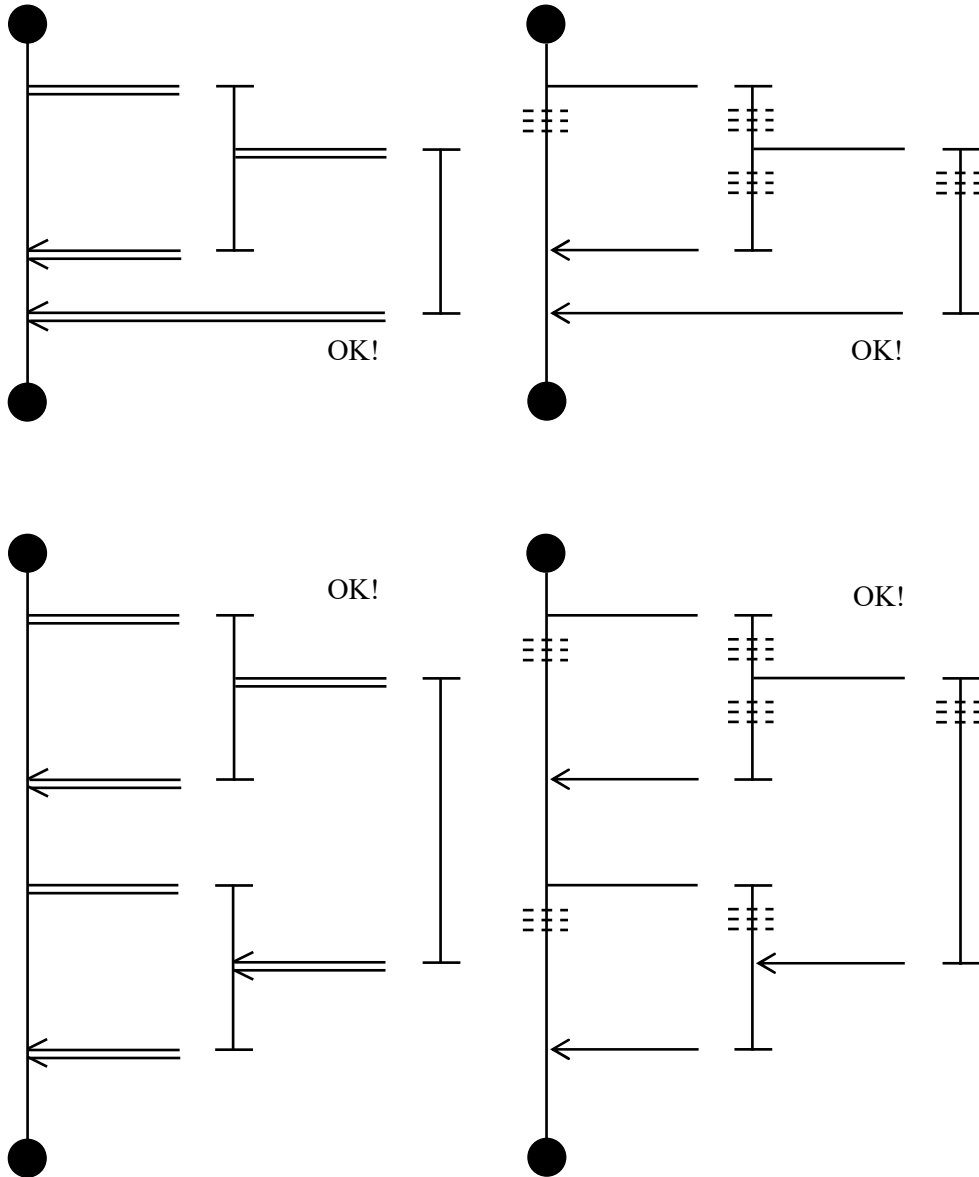
- (a) The route for starting branching is the same as the route for merging branches.
 All patterns are possible for both synchronization and selection.



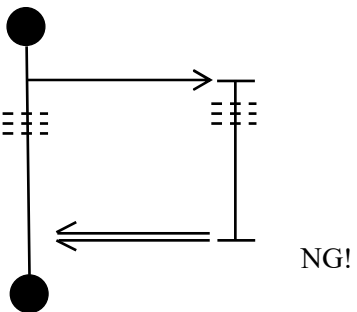
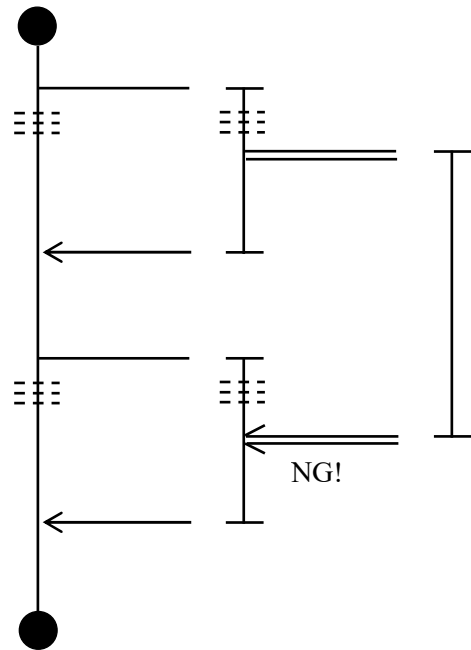
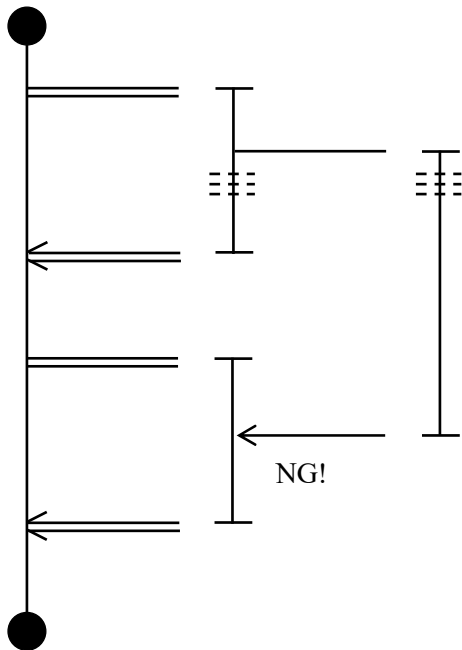
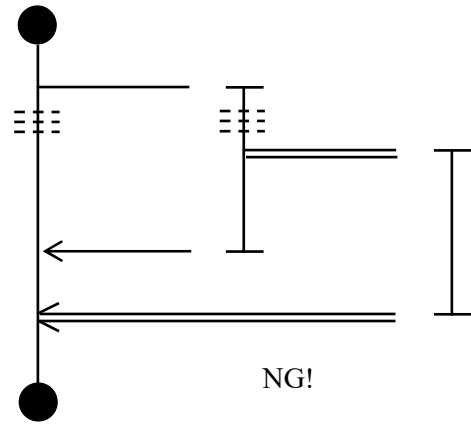
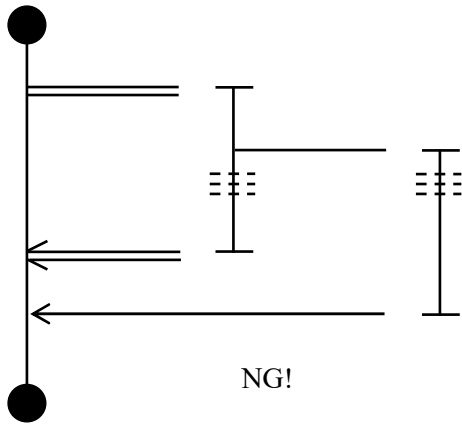
3. PROCESS

- (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]

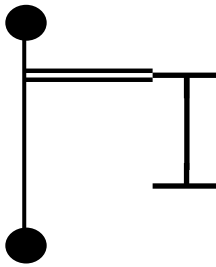


3. PROCESS

(2) General syntax structure of non-synchronous routes

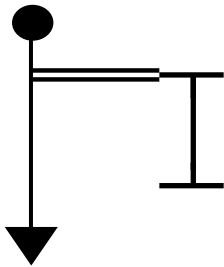
The general syntax structure of non-synchronous branching routes or, simply, non-synchronous routes is a branching route that does not merge to the route from which it has branched and that is used in conjunction with a non-synchronous process end. The major differences between the synchronous and non-synchronous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

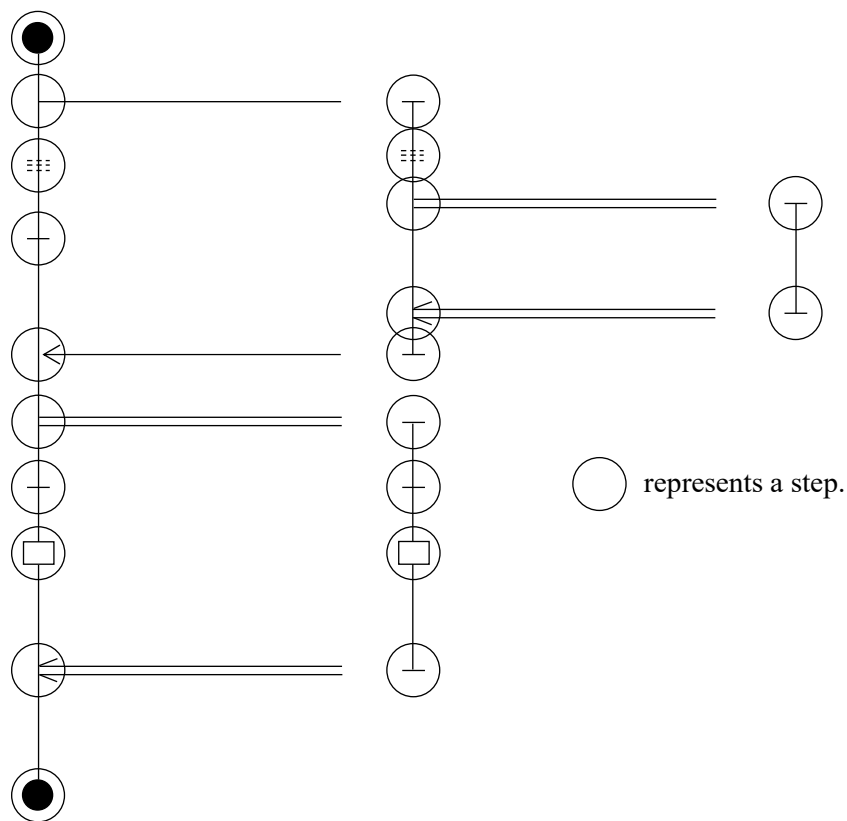
If all of the given branching routes merge to the route from which they have branched, the effect of the non-synchronous process end is the same as that of a process end.

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 \geq .
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.
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.
			Merges synchronization subroutines	No	You have to merge to the source route. End the synchronization subroutines.
		Multi-entry	Re-executes the process, starting with this step, when configured conditions hold	Yes	

3. PROCESS

(2/2)

No.	Figure	Name	Function	Syntax	Remark
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	

Labels

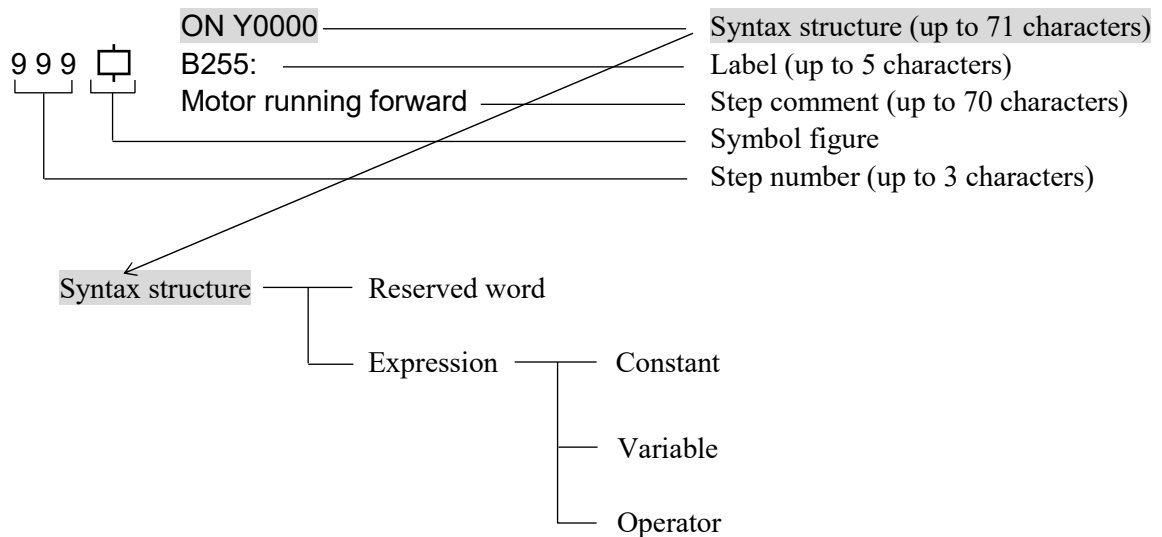
A label consists of a code between B1 and B255 (which can be created for each process and cannot be set to branch to another process). A colon (:) represents a jump destination from a branch figure and can only be added to a step.



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.

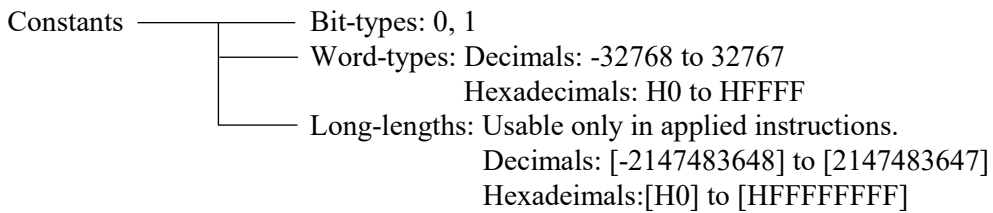
Reserved words

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

<p>List of reserved words</p> <p>ACT, CLR, MRST, ON, OFF, RST, STP, TASK, TUP, TRS, TCNT, CNxxx, CNExxx, PTxxx, WTxxx, WTSxxx, Bxxx, Pxxx, H????????</p> <p>Name of applied instruction (see Chapter 5.)</p> <p>xxx: It means a decimal constant.</p> <p>????????: It means a hexadecimal constant.</p>

Constants

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

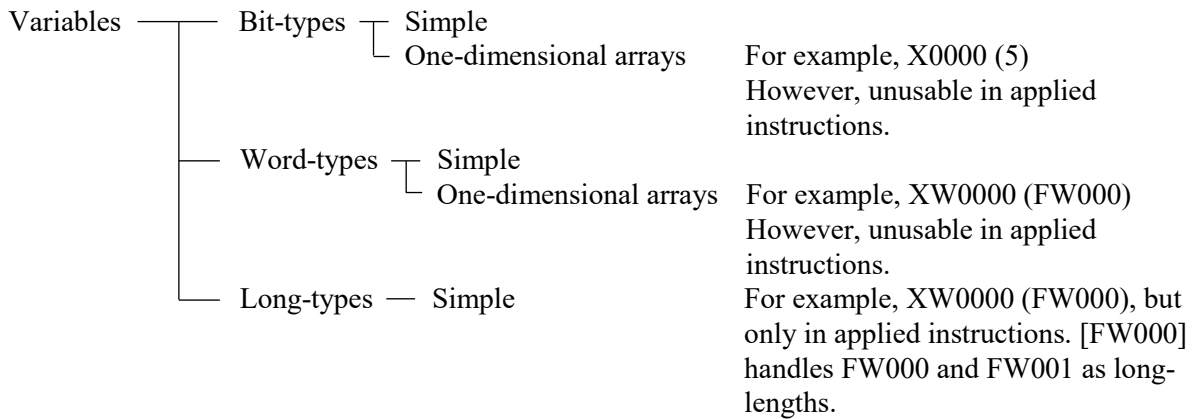


Variables

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.



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	0000 to FFFF	Bit	
		XW	0000 to FFF0	Word	
	External outputs	Y	0000 to FFFF	Bit	
		YW	0000 to FFF0	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	0000 to FFFF	Bit	
		MW	0000 to FFF0	Word	
		E	0000 to FFFF	Bit	
		EW	0000 to FFF0	Word	
		Z	000 to 3FF	Bit	
		ZW	000 to 3F0	Word	
		S	0000 to BFFF	Bit	
		SW	0000 to BFF0	Word	
	Other registers	J	000 to FFF	Bit	For linkage with a ladder
		JW	000 to FF0	Word	
		Q	0000 to FFFF	Bit	
		QW	0000 to FFF0	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
WTS		000 to 255			
PT		000 to 255			
Counters	CN	000 to 127		Decimal notation	
	CNE	000 to 127			
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 6-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)**

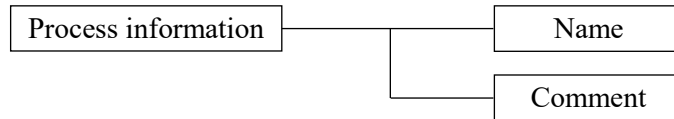
* When you combine a free label with a free comment, the system receives up to a total of 70 characters (including a colon (:)) as a free label).

3. PROCESS

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.

CHAPTER 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 omissible. 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]

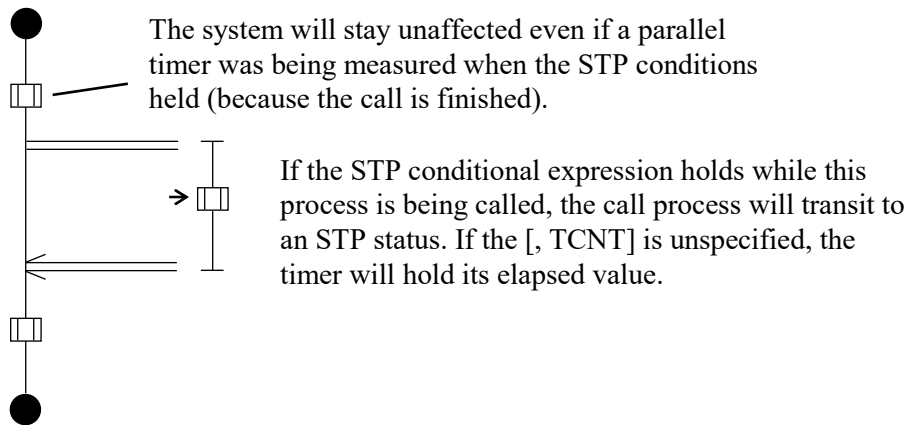
● [{ STP Conditional expression [,TCNT] [{ ON Group of PI/O bits [:OFF Group of PI/O bits] } }]
 {,RST Conditional expression [,TUP] [{ ON Group of PI/O bits [:OFF Group of PI/O bits] } }] }
 {,CLR Conditional expression}
 {,ACT Conditional expression}]
 *Group of PI/O bits - PI/O bit formula [,PI/O bit formula] ~

● (or ▼) Without syntax

4. DESCRIPTION OF SYNTAX

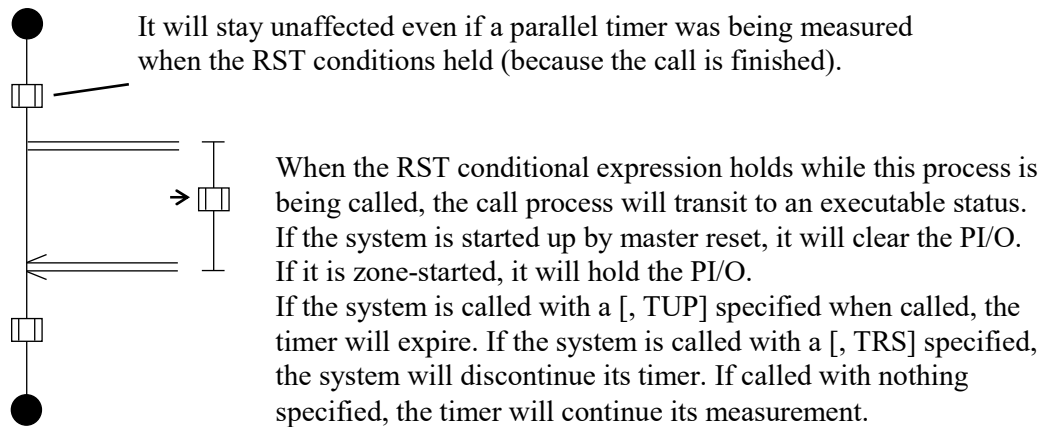
STP

- 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).

4. DESCRIPTION OF SYNTAX

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

[1] ● STP X0000, RST X001, CLR X0002, ACT X0003

- When X0000 is ON, go to a stop status (the elapsed value of the timer is held).
- When X0001 is ON, go to a reset status (the timer is discontinued).
- When X0002 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 X0000 and X0001 are OFF and X0003 is ON, go to the process being executed.

[2] ● STP G000&X0020, TCNT [ON J000:OFF J001] ~, RST Q0000, TUP

- When G000 and X0020 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 Q0000 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 Q0001, TUP [ON J001, G200], CLR X0200

- When Q0001 is ON, go to a reset status (timer is discontinued).
- When transiting to a reset status, turn ON J001 and G200.
- When X0200 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

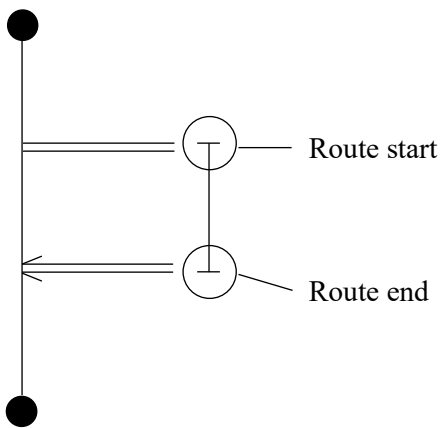
⌋ Without syntax

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.

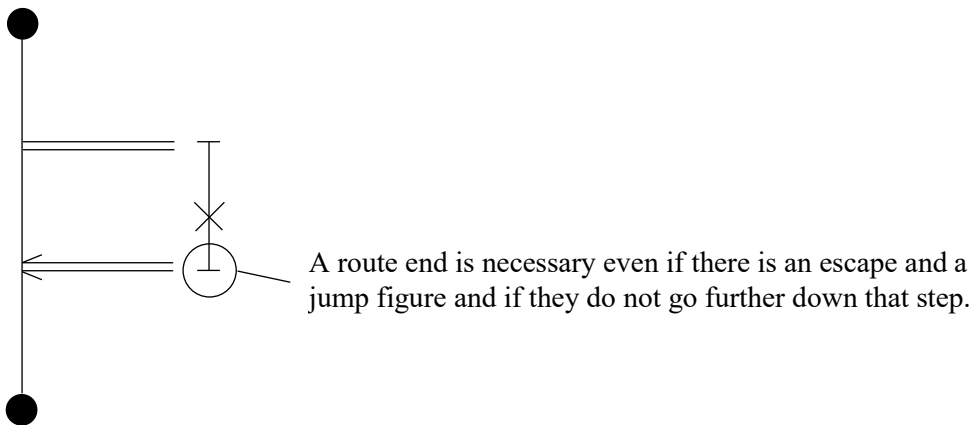
Creating a subroute builds up a synchronization syntax structure or a selection branch syntax structure.

[Typical programs with route start ⌊ and route end ⌋]

[1]



[2]



4. DESCRIPTION OF SYNTAX

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]

$$\begin{aligned} &+ \{ \text{Conditional expression [, timer, output bit]} \} \\ &\{ \text{WTxxx expression [, conditional expression]} \} \end{aligned}$$

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 (\dagger)]

[1] \dagger X0000

Go to the next step when X0000 is ON.

[2] \dagger GW000<H2000

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

[3] \dagger X0001 (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] \dagger WT000 (100)

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

[5] \dagger WT255 (10, X001F)

Go to the next step if X001F remains ON for one consecutive second after arriving at this step.

[6] \dagger GW000 > H2000, 100, Y0000

Go to the next step when GW000 becomes larger than H2000. Turn on Y0000 if the GW000 fails to become larger than H2000 within 10 seconds. Do not turn Y0000 OFF even if GW000 becomes larger than H2000 after Y0000 is turned ON.

[7] \dagger WT000 (FW000, X001F)

Go to the next step if X001F remains ON for the time (in 100 ms) of the FW000 value after arriving at this step. If the FW000 value is changed during the wait, the starting value is used for operation and reflected when a process is executed next time.

4. DESCRIPTION OF SYNTAX

4.4 Boxes

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

[Syntax]

```

□ {ON PI/O bit expression [, PI/O bit expression] ~}
  {OFF PI/O bit expression [, PI/O bit expression] ~}
  {Assignment expression}
  {Special assignment expression}
  {PT number (t1 [,t2] ), {ON Bit PI/O [, Bit PI/O] ~ [:OFF Bit PI/O [, Bit PI/O] ~] } }
  {OFF Bit PI/O [, Bit PI/O] ~ }
  { {TUP} {WT number} [ {, WT number} ] }
  {TRS} {PT number} [ {, PT number} ~ ]
  {CN number} [ {, CN number} ]

```

```

[ {:ON PI/O bit expression [,PI/O bit expression ~ Repetition] }
  {:OFF PI/O bit expression [,PI/O bit expression ~ Repetition] }
  {: Assignment expression}
  {: PT number (t1 [,t2], {ON Bit PI/O [, Bit PI/O] ~ [:OFF Bit PI/O [, Bit PI/O] ~] ) } } ~
  {OFF Bit PI/O [, Bit PI/O] ~ }
  {:{TUP} {WT number} [ {, WT number} ] }
  {TRS} {PT number} [ {, PT number} ~ ]
  {CN number} [ {, CN number} ] ]

```

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.

Bit-type variables

Y, G, A R, K, M E, Z, J Q, HH, LB	=	Y, G, A R, K, M E, Z, J Q, HH, LB X, S 0, 1	() & ~ ^	Y, G, A R, K, M E, Z, J Q, HH, LB X, S 0, 1
--	---	--	-------------------------	--

Word-type variables

YW, GW AW, RW KW, MW EW, ZW JW, QW DW, FW LBW LWW LXW	=	YW, GW AW, RW KW, MW EW, ZW JW, QW DW, FW XW, SW LBW LWW LXW Decimal constants Hexadecimal constants	() & ~ ^ * / + -	YW, GW AW, RW KW, MW EW, ZW JW, QW DW, FW XW, SW LBW LWW LXW Decimal constants Hexadecimal constants
---	---	---	---	---

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.

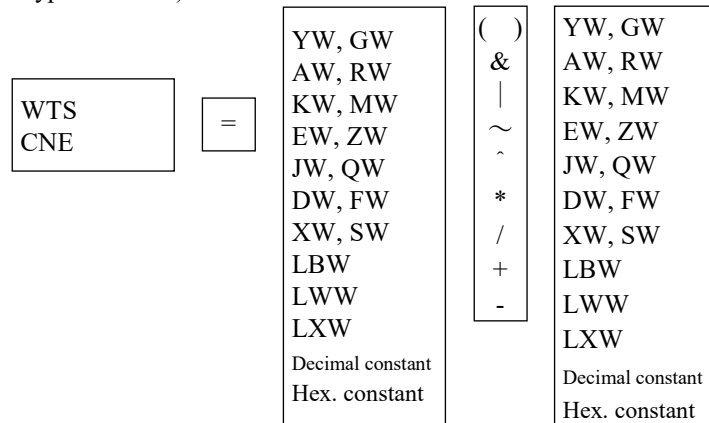
In divisions, too, both the divisor and the dividend are one-word. The portion that cannot be expressed in one word is rounded off, with the result being one-word. Dividing a number by 0 results in the answer remaining unchanged.

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.

Special assignment expression

Support a special assignment expression for the timer (WR) and counter (CN).

(Word-type variables)



WTS is a timer setting value which can be changed any time to make the timer in the measurement process end earlier than when measurement started. Such changes have no effect if the timer does not start measurement or has already expired.

CNE is a counter end value that can be effectively used in repeat start/repeat end syntax. During the repetition process, the end value can be changed to finish the timer earlier or later than the end value when the timer started. CNE is not effective in syntax other than repeat start/repeat end. Note that changing an end value improperly might result in an infinite loop and system failure.

4. DESCRIPTION OF SYNTAX

[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] □ $YW0000 (DW001) = HFFFF$

Assign /FFFF to the array of YW0000 with DW001 at the time as a subscript value.

[3] □ $WTS001 = 50$

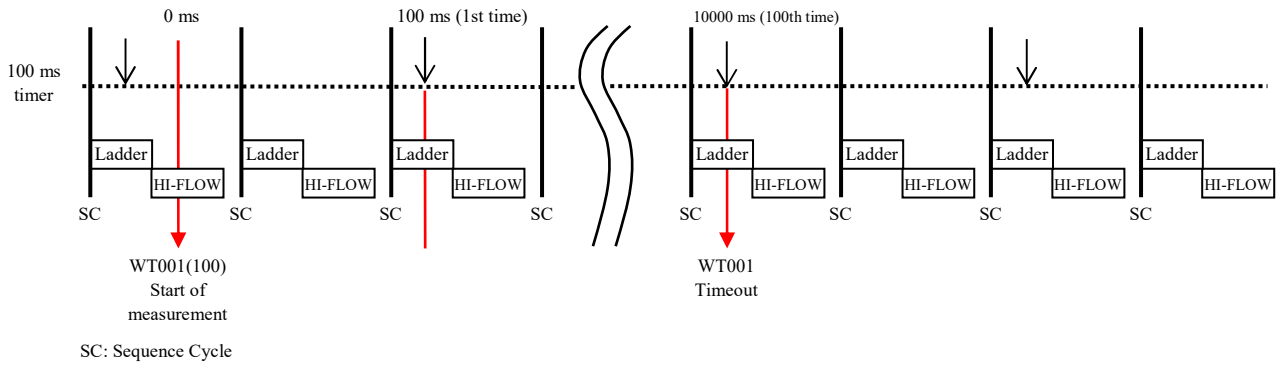
Change the setting value of WT001 to 50.

[4] □ $CNE001 = DW000$

Change the end value of CN001 to the content of DW000.

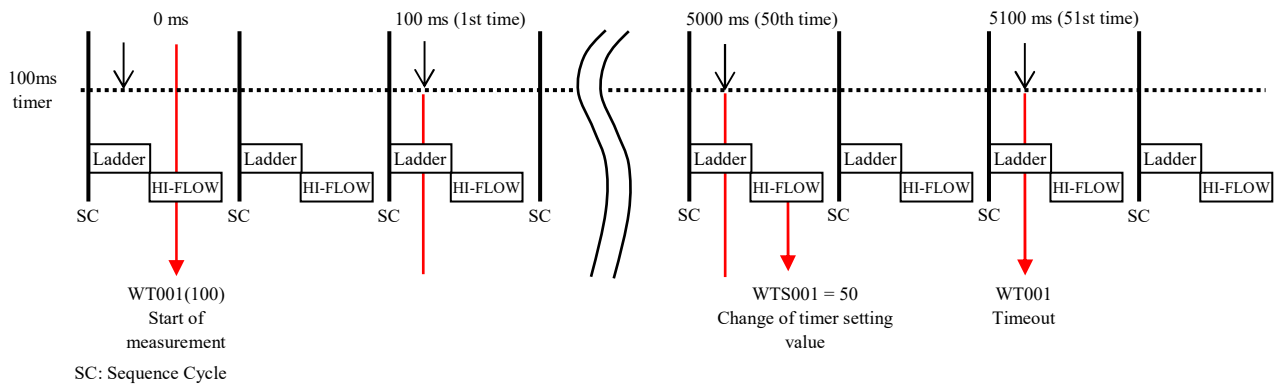
The timer (WT) operations are shown below.

<Normal timeout>



If measurement starts with WT001 set to 100, a timeout is caused by the 100th timer interrupt after the start of measurement.

<Timeout when timer setting value is changed>





If measurement starts with WT001 set to 100, and then, after 5000 ms, is set to 50, a timeout is caused by the 51st timer interrupt after the start of measurement. In other words, the timeout occurs when the timer interrupt occurs 100 ms after the timer setting value is changed (not at the moment of change).

4. DESCRIPTION OF SYNTAX

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 Y0000, Y000F: OFF Y0001

Turn ON Y0000 and Y000F, turn OFF Y0001, 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 Y0000, Y0001

Turn OFF Y0000 and Y0001 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

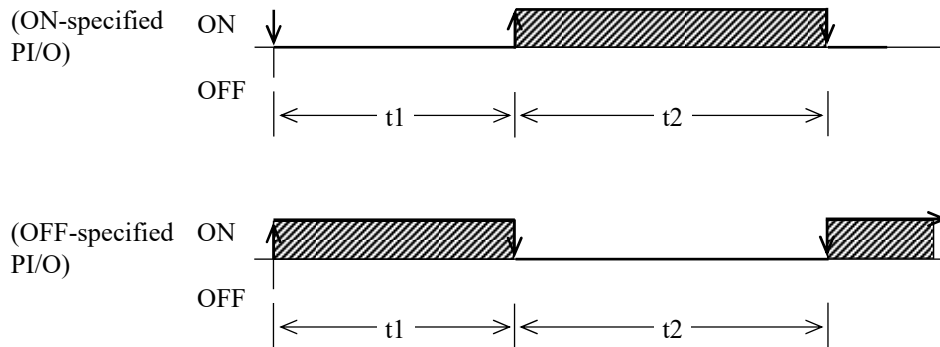
The system will produce a waveform onto a desired PI/O. The t1 represents rising time, while the t2 represents falling time.

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 65535, 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 Y0000: OFF Y0001)

	On passing by this step (to a next step at once)	1 second later	2 seconds later
Y0000	? →OFF	→ON	→OFF
Y0001	? →ON	→OFF	→ON


?: ON or OFF

[2] □ PT010 (20, ON G000: OFF G001)

	On passing by this step (to a next step at once)	2 seconds later	
G000	? →OFF	→ON	→
G001	? →ON	→OFF	→

?: ON or OFF

4. DESCRIPTION OF SYNTAX

[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	→

?: ON or OFF


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	The system issues a RLEAS or QUEUE macro by specifying Pxxx as the CPMS task (1 to 255) on CP with the , TASK, Factor number option.

4. DESCRIPTION OF SYNTAX

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	The system issues an ABORT macro by specifying Pxxx as the CPMS task (1 to 255) on CP with the TASK option. (This function is unsupported).

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 clear PI/O and go to the next step.

[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.

4. DESCRIPTION OF SYNTAX

[6] ■ RST P12, TASK

Issue an ABORT macro to CPMS task 12, and then go to the next step. (This function is unsupported.)

[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 65535. Setting the range between -32768 and -1 will operate the system on the assumption that it is set between 32768 and 65535.

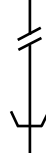
[Syntax]

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

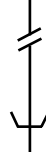
 Without syntax

[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)

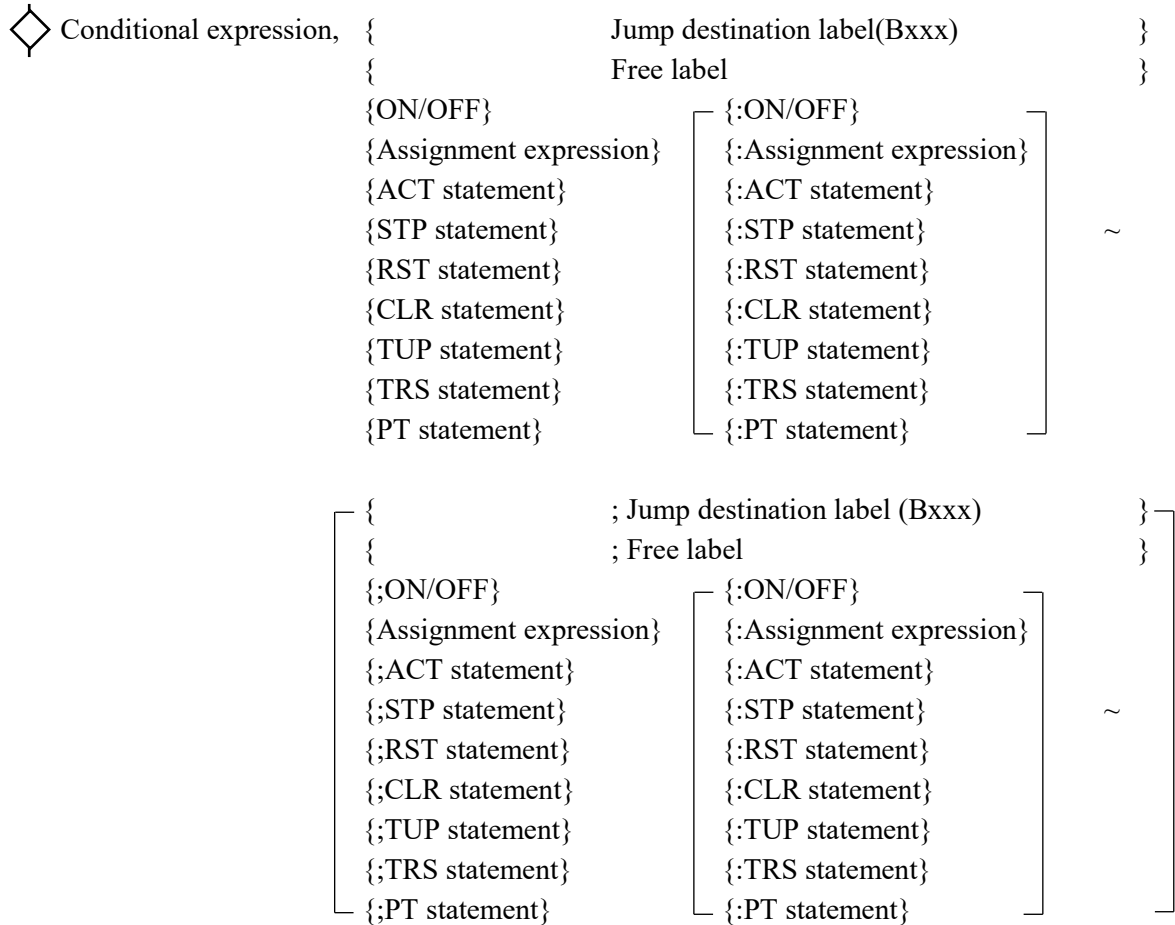
 The value between FW000 and FW002 that was used when the system passed through the repeat start for the first time will be the initial value, final value, or an increment.

4. DESCRIPTION OF SYNTAX

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]



(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] ◇ X0000, B1; LABEL

When X0000 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<> (YW0000&H3000), ON Q0005

If the logical product of YW0000 and H3000 is not 0, turn ON Q0005. If it is 0, do nothing and go to the next step.

[3] ◇ Q0000, FW100=FW100+1; ACT P10

If Q0000 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; EW0000=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 EW0000 to 8, turn on J000, and go to the next step.

[5] ◇ X0010, ON J000, J001, J002, J003; ERRLB

When X0010 is ON, turn ON J000, J001, J002, and J003 and go to the next step.
When X0010 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).

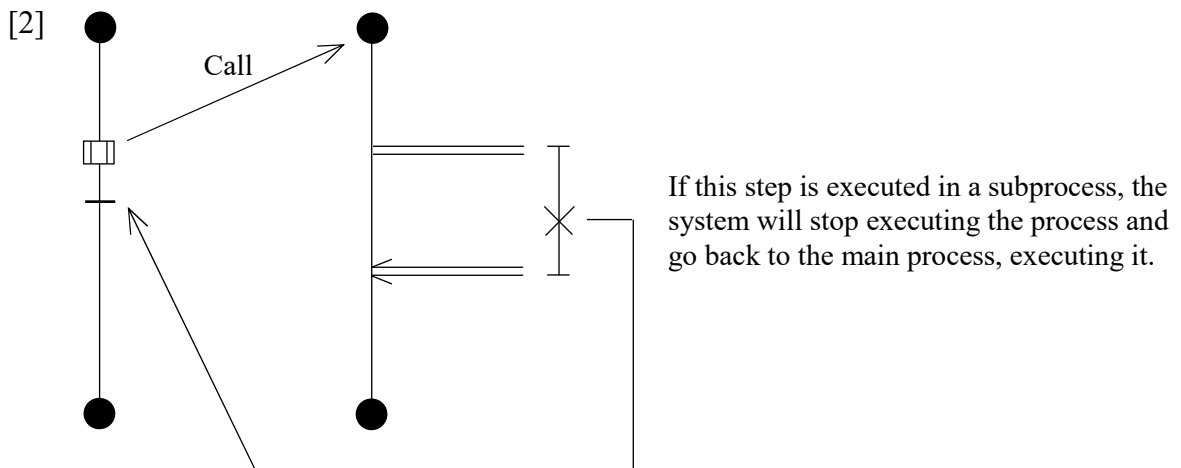
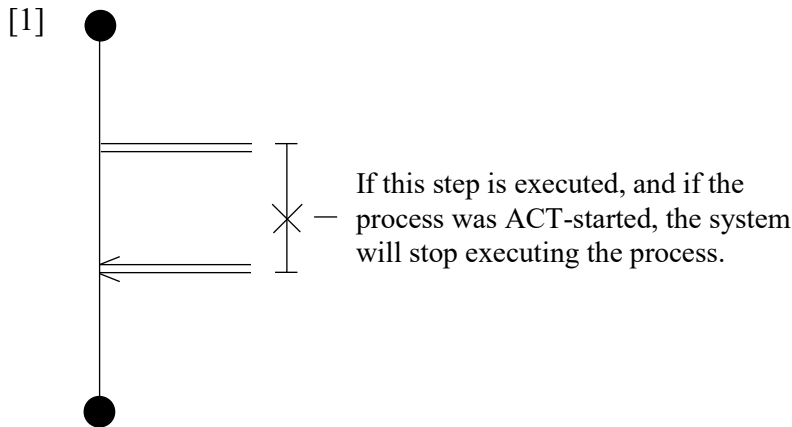
Subprocesses are basically handled in the same way as the main process. The system will restore the executed place to the main process with the same scan.

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 (✕)]



4. DESCRIPTION OF SYNTAX

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.

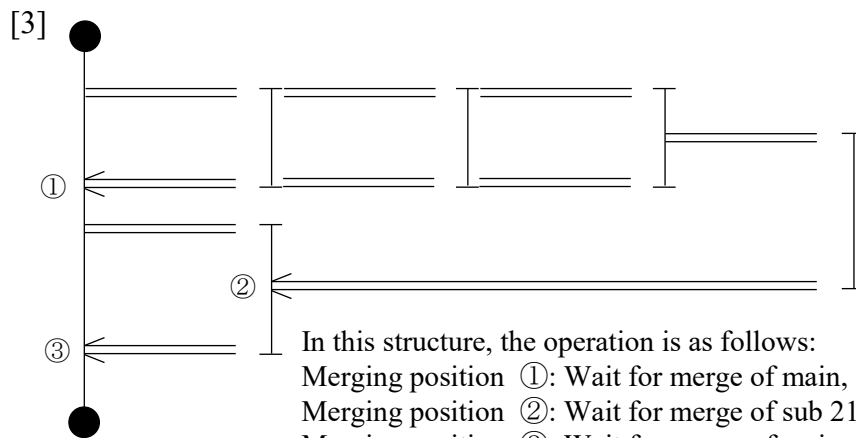
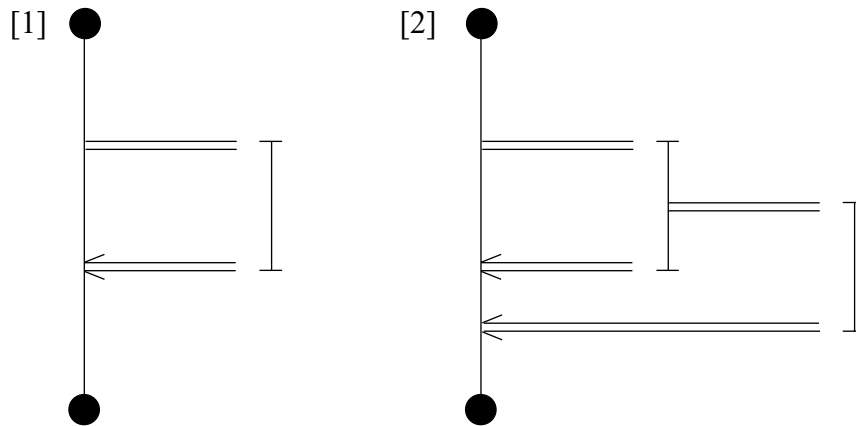
In conventional practice, the system used to monitor the end of the subroute where the para end merged (that is, the main route was being executed), so that the execution of the next step is delayed by one scan. The system checks whether both para end and route end merged for the last time or not. If merged, execution of the next step at the main route merging position is finished. If not, execution of the system's own route is finished (the main route is not always being executed), so no scan delay occurs.

[Syntax]

\vDash Without syntax

\vDash Without syntax

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



In this structure, the operation is as follows:
 Merging position ①: Wait for merge of main, sub 11, sub 12, and sub 12.
 Merging position ②: Wait for merge of sub 21 and sub 31.
 Merging position ③: Wait for merge of main and sub 21.

4.11 Para Start and Select End

A pair consisting of para start and select end represents a portion to be processed in parallel. The para start will start multiple subroutes, and then go to the step following the system's own route.

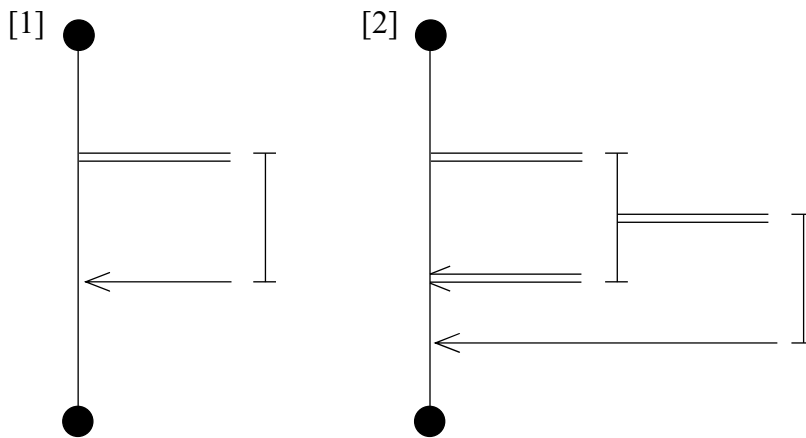
When one of the merging routes is finished, the select end shuts down the other unfinished routes, and then executes the step following the system's own route.

[Syntax]

⊨ Without syntax

⊨ Without syntax

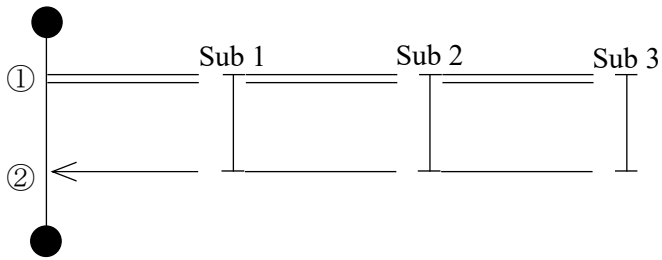
[Typical programs with para start (⊨) and select end (⊨)]



4. DESCRIPTION OF SYNTAX

[Description of operation 1]

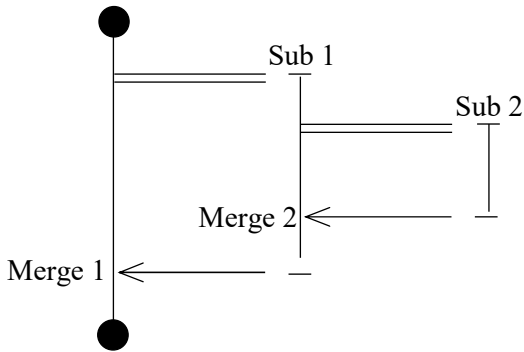
If there are para start + select end only



- If step ① of the main route is executed, the system will execute four routes of main route, sub 1, sub 2, and sub 3 in parallel.
- If the main route, sub 1, sub 2, or sub 3 is finished (merged), the system will shut down the others, and then execute the next step of the main route.

[Description of operation 2]

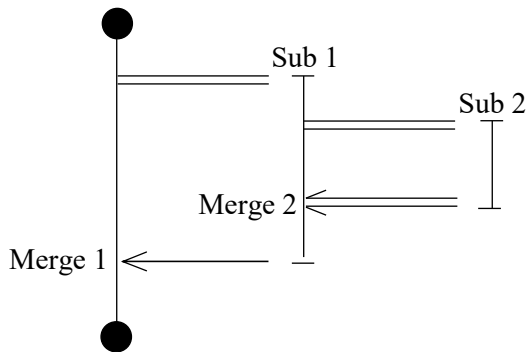
If para start + select end is included in para start + select end



- If merge 1 arrives first (in this case, the main route arrives first), the system will shut down sub 1 and sub 2.
- If merge 2 arrives first, the system will execute the normal para start + select end operation of sub 1 and sub 2.

[Description of operation 3]

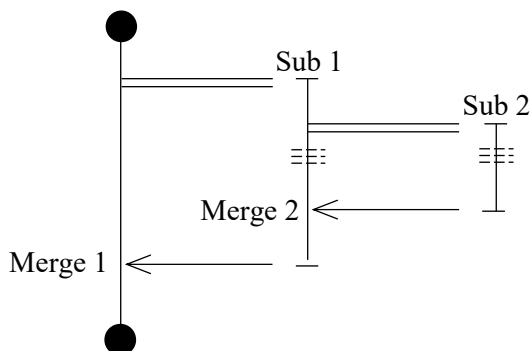
If para start + para end is included in para start + select end



- If merge 1 arrives first (in this case, the main route arrives first), the system will shut down sub 1 and sub 2.
- If merge 2 arrives first, the system will execute the normal para start + para end operation of sub 1 and sub 2.

[Description of operation 4]

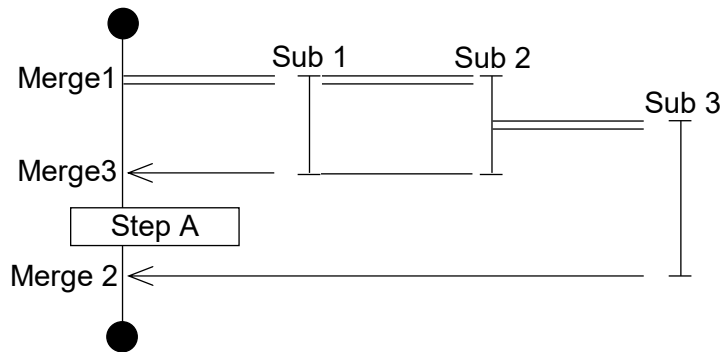
If select + select end is included in para start + select end



- If merge 1 arrives first (in this case, the main route arrives first), the system will shut down sub 1 and sub 2.
- If merge 2 arrives first, the system will execute the normal select + select end operation of sub 1 and sub 2.

NOTICE

In the syntax that makes para start and merges at select end (as in the pattern shown below), do not describe any syntax that makes para start again in the middle of a subroute and merges at select end into the main route, not the subroute. If a merging position is reached, the system will shut down only the route having the same merging position. Therefore, non-merging subroutes that are being executed will remain as they are.



[Description of operation]

- During execution of sub 1 to sub 3, if sub 3 reaches merge 2 before sub 1 and sub 2 reach merge 3, the execution points of sub 1 and sub 2 will not be shut down.

4.12 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.

If a subroute is selected, the route end of the route will start the main route and execute the step following the merging position. Therefore, no scan delay will result.

It is acceptable for both select end and select to not exist on the same route (not merge into the source route).

<Notice>

The cell wait must be at the step following the select.

[Syntax]

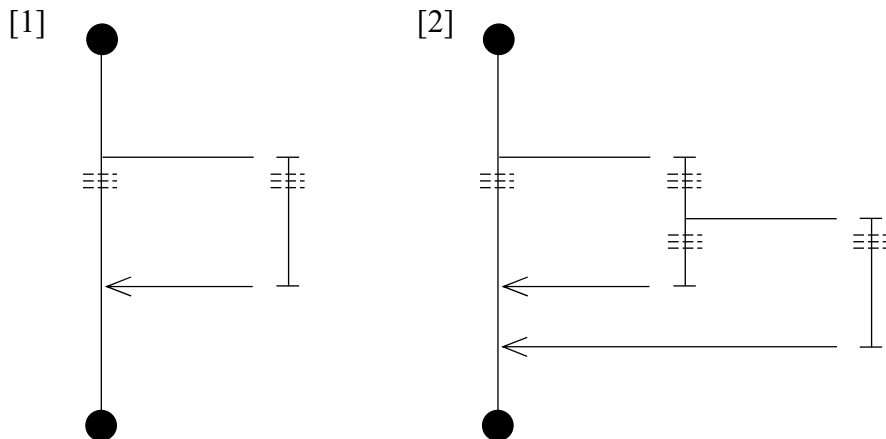
┆ Without syntax

≡≡≡ 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 (⌞)]



4.13 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] ┌ X0000

Re-execute the operation, beginning with this step, when X0000 is ON.

[2] ┌ GW000<H2000

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

4. DESCRIPTION OF SYNTAX

4.15 Function

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

[Syntax]

\bigcirc Name of applied instruction [,Parameter] ~

4.16 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]

\dagger^* { Conditional expression }
{ WTxxx (formula [,Conditional expression]) }

4.17 Non-synchronous Process End

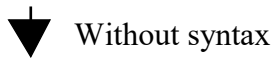
A non-synchronous process end, used in conjunction with a process start, causes the process to be terminated without waiting for any of the given non-synchronous branching routes to reach their ends. To make a non-synchronous process end function asynchronously with all of the given

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]

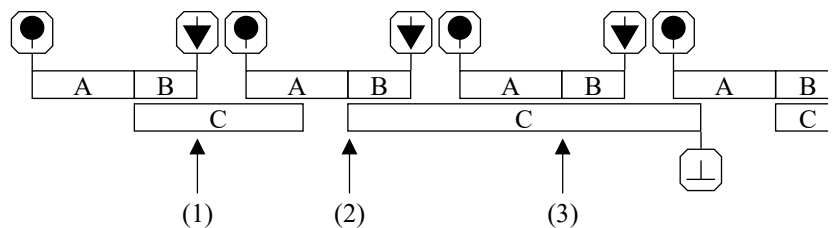
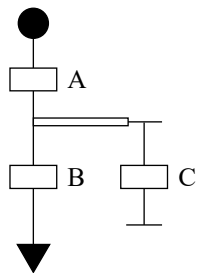


[Sample programs]

[1] Sample program using only one non-synchronous route

<Sample circuit>

<Timing chart>

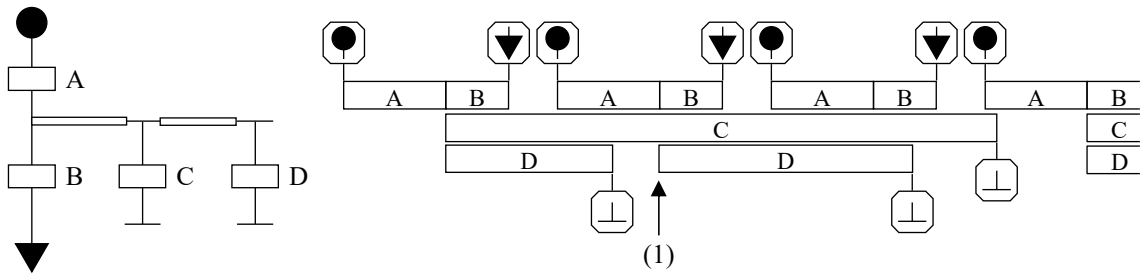


- (1) When the main route reaches its non-synchronous process end, the process is terminated, but the non-synchronous route is left undisturbed because it is still on the way to its route end.
- (2) Because the non-synchronous route has already reached its end, it is started again at its route start.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

4. DESCRIPTION OF SYNTAX

[2] Sample program using two non-synchronous routes

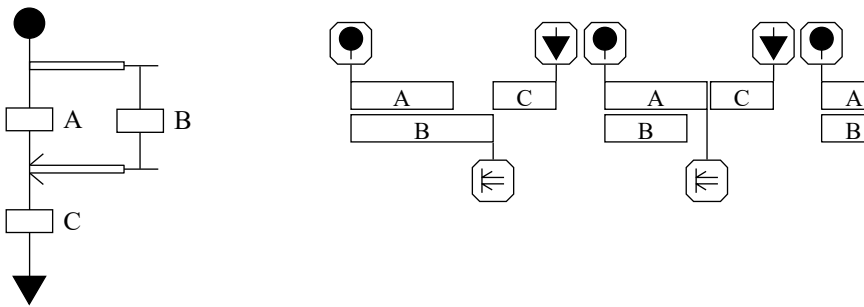
<Sample circuit> <Timing chart>



(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

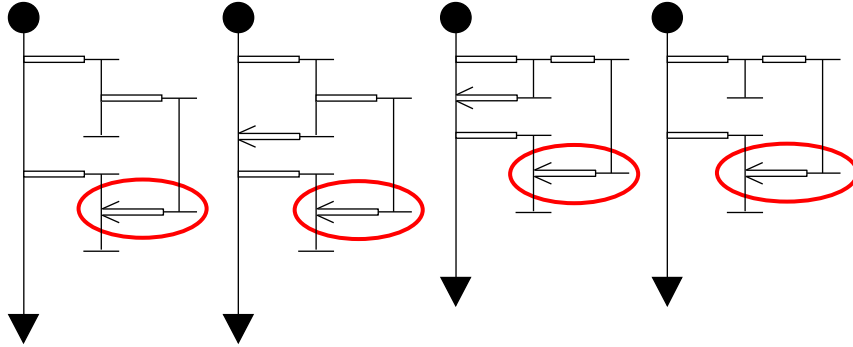
<Sample circuit> <Timing chart>



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

- Do not describe any syntax that has differing non-synchronous processes between branching and merging, and that has a non-synchronous route as a merging route (as in the pattern shown below). If execution is attempted, a new main route will not be started because the system judges that a subroute is being executed.



- 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.

This Page Intentionally Left Blank

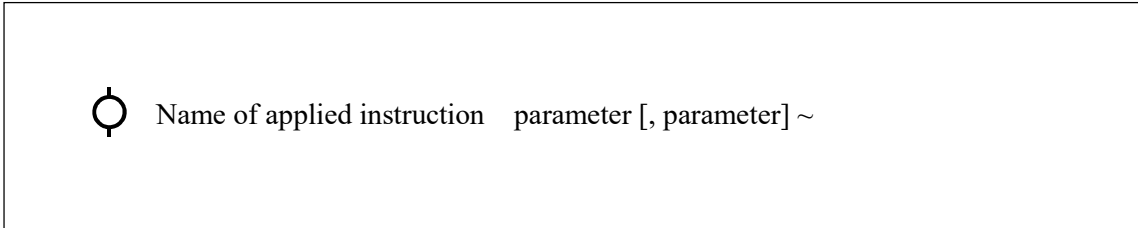
CHAPTER 5 APPLIED INSTRUCTIONS

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

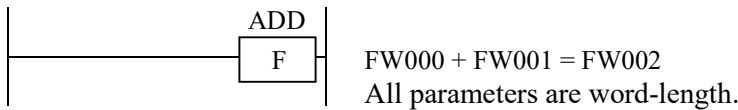
Applied instructions are programmed as follows:



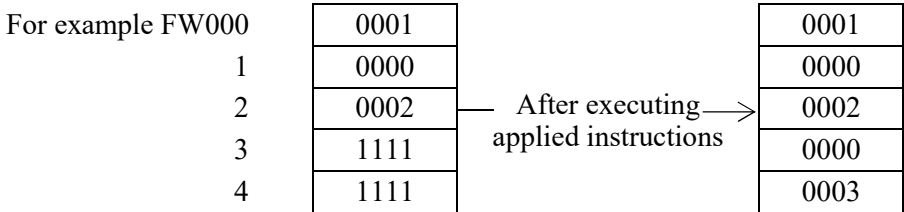
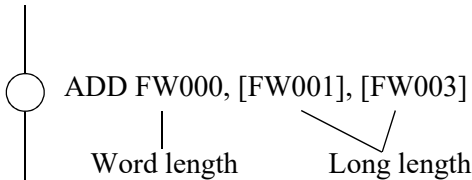
5.3 Parameters

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



5. APPLIED INSTRUCTIONS

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														
	X0000 <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Data 1</td></tr><tr><td>Data 2</td></tr></table> X0001 <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Data 2</td></tr></table> Data 1, 2 <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Data a</td></tr><tr><td>Data b</td></tr></table>	Data 1	Data 2	Data 2	Data a	Data b	FW000 <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Data 3</td></tr><tr><td>Data 4</td></tr></table> FW001 <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Data 4</td></tr></table> Data 3, 4 <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Data c</td></tr><tr><td>Data d</td></tr></table>	Data 3	Data 4	Data 4	Data c	Data d	XXXX YYYYYYYYY XXXX <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Data e</td></tr><tr><td>Data f</td></tr><tr><td>Data g</td></tr><tr><td>Data h</td></tr></table> YYYYYYYYY	Data e	Data f	Data g	Data h
Data 1																	
Data 2																	
Data 2																	
Data a																	
Data b																	
Data 3																	
Data 4																	
Data 4																	
Data c																	
Data d																	
Data e																	
Data f																	
Data g																	
Data h																	
[1] Direct word length	AND result of data 1 and 1	Data 3	XXXX, but long-length YYYYYYYYY is a low-level word only.														
Example	X0000	FW000	1230 H20000000														
[2] Direct long length	AND result of data 1, 2 and 1	Data 3, 4	XXXX, YYYYYYYYY However, XXXX is handled as a long-length.														
Example	[X0000]	[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 YYYYYYYYY, data g. If XXXX and YYYYYYYYY 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 YYYYYYYYY, data g, h. If XXXX and YYYYYYYYY are odd numbers, it is an error.														
Example		@[FW000]	@[HFFF0] @[H180000]														

<Notice>

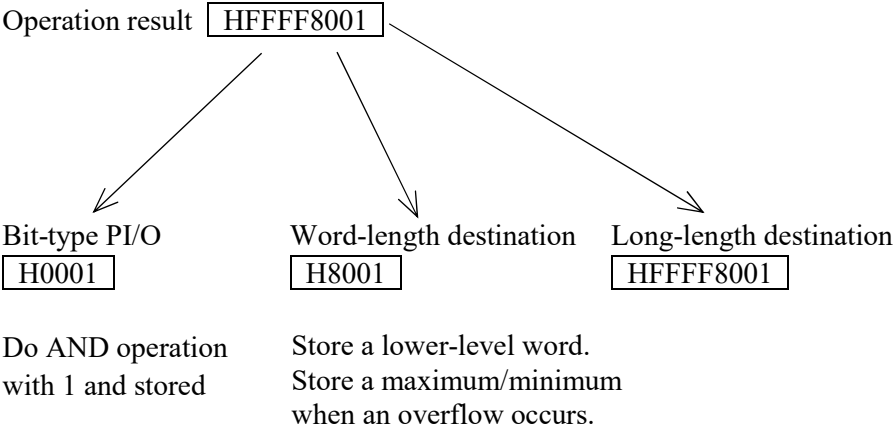
In the ranges of bit-type PI/O areas, do not specify parameters of applied instructions with an indirect long length (H0220 0000 to H023F 0000 and H0270 0000 to H027F 0000) because long-word access to those areas cannot be performed.

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.

FW000 8001 ————— Handled as HFFFF8001 during an operation.

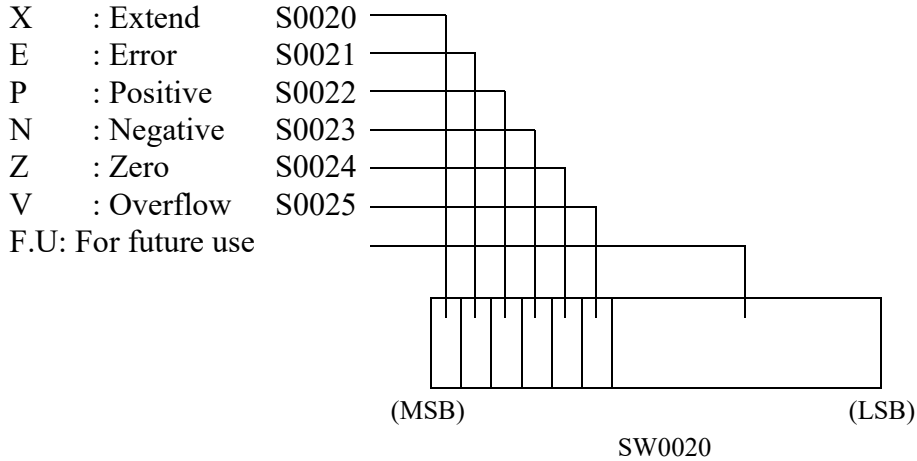
When storing an operation result, the system converts the type of the result according to the destination.



5.5 System Error Flags

Various flags are set to SW0020 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.


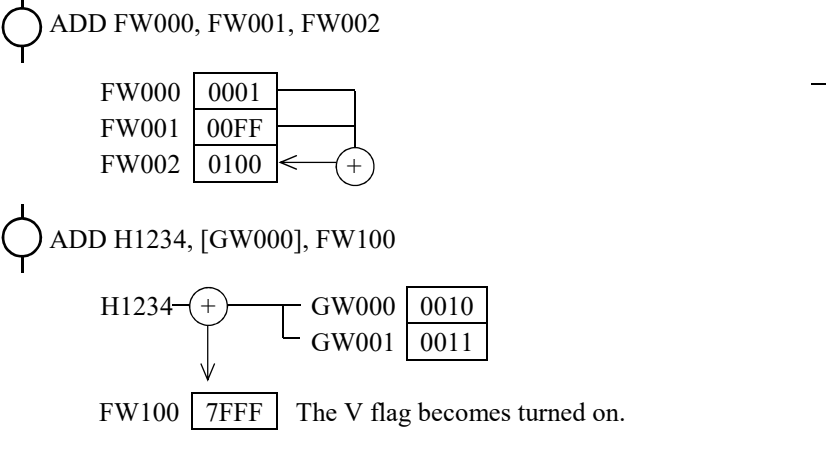
Error flags When the number of parameters of applied instructions used differs
 When the CPU is memory-protected, and when the address and PI/O specified by result (R) indicate the inside of the protect area
 When a specified PI/O is defective (such as when it is unserviceable)

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/H7FFF
 Negative overflow/H8000
 In regard to word length, bit-type PI/O can be specified for only the LSB.
 If it is specified for a bit other than the LSB, no overflow flag will be set.

Long length: Positive overflow/H7FFFFFFF
 Negative overflow/H80000000

5.6 Function Description



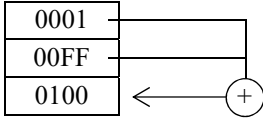

This section specifies the applied instructions. Here is the way they will be described.


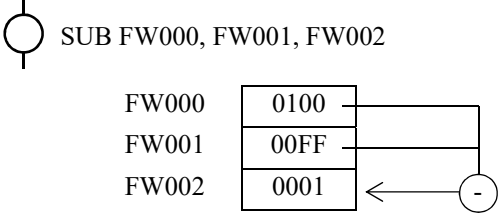
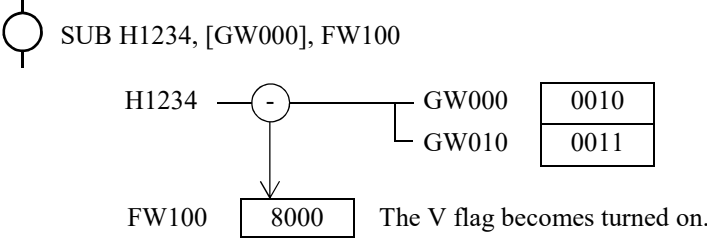
Name of applied instruction	Function name																																										
ADD	Addition		Outlines how the applied instruction is operated.																																								
Function description	This function calculates the sum of the source and destination and stores it in the result.																																										
Parameter and operation	 ADD S, D, R S: Source D: Destination R: Result	S+D → R	Schematizes the operation. Shows the array of parameters.																																								
Flag configuration	The E and V will change. The others will become turned off.		Shows the flag to be changed after the instruction.																																								
Remark			Provides cautions.																																								
Typical use			Shows how it is typically used.																																								
Effective parameter	<table border="1" data-bbox="327 1310 1157 1825"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <tr> <td>Direct word length</td> <td>√</td> <td>√</td> <td>√</td> <td>Direct word length</td> <td>√</td> <td>√</td> <td>-</td> </tr> <tr> <td>Direct long length</td> <td>-</td> <td>√</td> <td>√</td> <td>Direct long length</td> <td>-</td> <td>√</td> <td>-</td> </tr> <tr> <td>Indirect word length</td> <td>-</td> <td>△</td> <td>△</td> <td>Indirect word length</td> <td>-</td> <td>△</td> <td>△</td> </tr> <tr> <td>Indirect long length</td> <td>-</td> <td>△</td> <td>△</td> <td>Indirect long length</td> <td>-</td> <td>△</td> <td>△</td> </tr> </tbody> </table>		S, D	Bit-type PI/O	Word-type PI/O	Constant	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct word length	√	√	-	Direct long length	-	√	√	Direct long length	-	√	-	Indirect word length	-	△	△	Indirect word length	-	△	△	Indirect long length	-	△	△	Indirect long length	-	△	△	The S (source), D (destination), and R (result) show the parameter types that can be effectively specified.
S, D	Bit-type PI/O	Word-type PI/O	Constant	R	Bit-type PI/O	Word-type PI/O	Constant																																				
Direct word length	√	√	√	Direct word length	√	√	-																																				
Direct long length	-	√	√	Direct long length	-	√	-																																				
Indirect word length	-	△	△	Indirect word length	-	△	△																																				
Indirect long length	-	△	△	Indirect long length	-	△	△																																				

Unspecifiable Effective conditional specification Effective specification


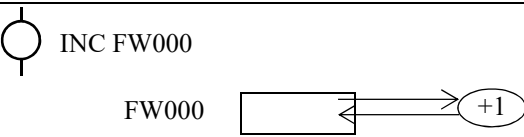
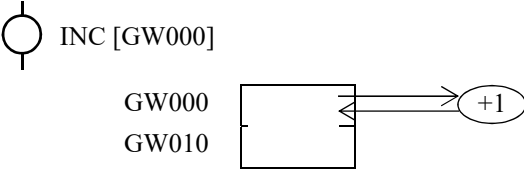
Target parameter types (source, destination, and result)


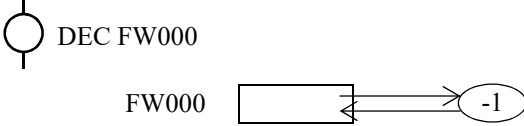
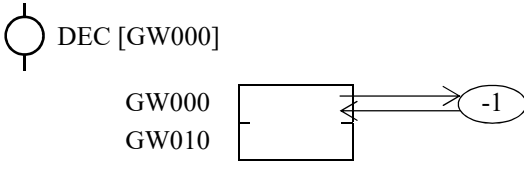
5. APPLIED INSTRUCTIONS

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 <div style="float: right; border: 1px solid black; padding: 5px; margin-left: 20px;"> $S+D \rightarrow R$ </div> <p>S: Source D: Destination R: Result</p>																																								
Flag configuration	The E and V will change. The others will become turned off.																																								
Remark																																									
Typical use	 ADD FW000, FW001, FW002 <div style="margin-left: 40px;"> <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>FW000</td><td>0001</td></tr> <tr><td>FW001</td><td>00FF</td></tr> <tr><td>FW002</td><td>0100</td></tr> </table>  </div>  ADD H1234, [GW000], FW100 <div style="margin-left: 40px;"> <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>H1234</td><td>7FFF</td></tr> </table> <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>GW000</td><td>0010</td></tr> <tr><td>GW010</td><td>0011</td></tr> </table> <p>The V flag becomes turned on.</p> </div>	FW000	0001	FW001	00FF	FW002	0100	H1234	7FFF	GW000	0010	GW010	0011																												
FW000	0001																																								
FW001	00FF																																								
FW002	0100																																								
H1234	7FFF																																								
GW000	0010																																								
GW010	0011																																								
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <p>△ is an address. Parameter error if the number is odd.</p>	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	–	△	△
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	–	△	△																																						



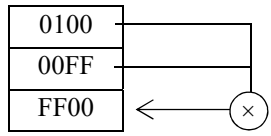

SUB	Subtraction																																										
Function description	This function subtracts the contents of the destination from the source and stores it in the result.																																										
Parameter and operation	 SUB S, D, R S: Source D: Destination R: Result	$S - D \rightarrow R$																																									
Flag configuration	The E and V will change. The others will become turned off.																																										
Remark																																											
Typical use	 																																										
Effective parameter	<p>Δ is an address. Parameter error if the number is odd.</p> <table border="1" data-bbox="359 1131 869 1556"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table> <table border="1" data-bbox="917 1131 1428 1556"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table>			S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	Δ	Δ	Indirect long length	–	Δ	Δ	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	Δ	Δ	Indirect long length	–	Δ	Δ
S, D	Bit-type PI/O	Word-type PI/O	Constant																																								
Direct word length	√	√	√																																								
Direct long length	–	√	√																																								
Indirect word length	–	Δ	Δ																																								
Indirect long length	–	Δ	Δ																																								
R	Bit-type PI/O	Word-type PI/O	Constant																																								
Direct word length	√	√	–																																								
Direct long length	–	√	–																																								
Indirect word length	–	Δ	Δ																																								
Indirect long length	–	Δ	Δ																																								



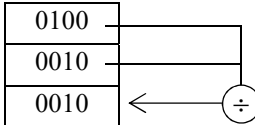


5. APPLIED INSTRUCTIONS

INC	+1 (Increment)																				
Function description	This function adds 1 to the contents of the source.																				
Parameter and operation	 INC S <div style="float: right; border: 1px solid black; padding: 5px; margin-left: 20px;"> S+1 → S </div> <p>S: Source</p>																				
Flag configuration	The E and V will change. The others will become turned off.																				
Remark																					
Typical use	 <p>INC FW000</p> <p>FW000</p>  <p>INC [GW000]</p> <p>GW000 GW010</p> <p>The system will increment GW000 and GW001, regarding them as long variables.</p>																				
Effective parameter	<p>△ is an address. Parameter error if the number is odd.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	-	Direct long length	-	√	-	Indirect word length	-	△	△	Indirect long length	-	△	△
S	Bit-type PI/O	Word-type PI/O	Constant																		
Direct word length	√	√	-																		
Direct long length	-	√	-																		
Indirect word length	-	△	△																		
Indirect long length	-	△	△																		


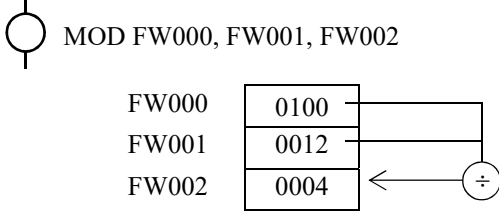
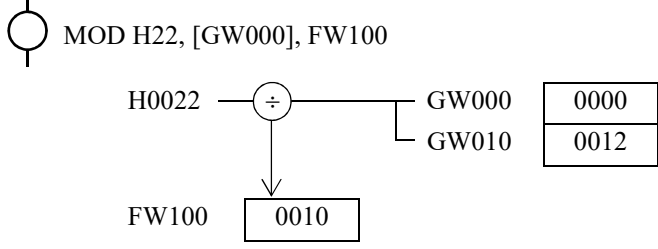
DEC	-1 (decrement)																				
Function description	This function subtracts 1 from the contents of the source.																				
Parameter and operation	 DEC S <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 200px;">S-1 → S</div> S: Source																				
Flag configuration	The E and V will change. The others will become turned off.																				
Remark																					
Typical use	  The system will decrement GW000 and GW001, regarding them as long variables.																				
Effective parameter	<p>△ is an address. Parameter error if the number is odd.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	-	Direct long length	-	√	-	Indirect word length	-	△	△	Indirect long length	-	△	△
S	Bit-type PI/O	Word-type PI/O	Constant																		
Direct word length	√	√	-																		
Direct long length	-	√	-																		
Indirect word length	-	△	△																		
Indirect long length	-	△	△																		




5. APPLIED INSTRUCTIONS

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 <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 200px;"> $S \times D \rightarrow R$ </div> <p>S: Source D: Destination R: Result</p>																																								
Flag configuration	The E and V will change. The others will become turned off.																																								
Remark																																									
Typical use	 MUL FW000, FW001, FW002 <div style="margin-left: 100px;"> <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>FW000</td><td>0100</td></tr> <tr><td>FW001</td><td>00FF</td></tr> <tr><td>FW002</td><td>FF00</td></tr> </table>  </div>  MUL H22, [GW000], FW100 <div style="margin-left: 100px;"> <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>H0022</td><td>7FFF</td></tr> </table> <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>GW000</td><td>0010</td></tr> <tr><td>GW010</td><td>0011</td></tr> </table> <p>The V flag becomes turned on.</p> </div>	FW000	0100	FW001	00FF	FW002	FF00	H0022	7FFF	GW000	0010	GW010	0011																												
FW000	0100																																								
FW001	00FF																																								
FW002	FF00																																								
H0022	7FFF																																								
GW000	0010																																								
GW010	0011																																								
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <p>△ is an address. Parameter error if the number is odd.</p>	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	–	△	△
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	–	△	△																																						


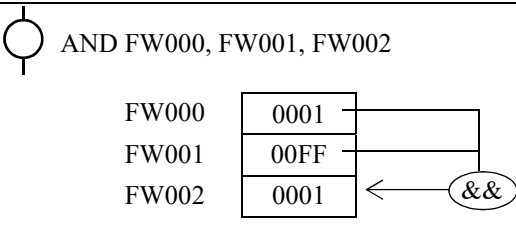
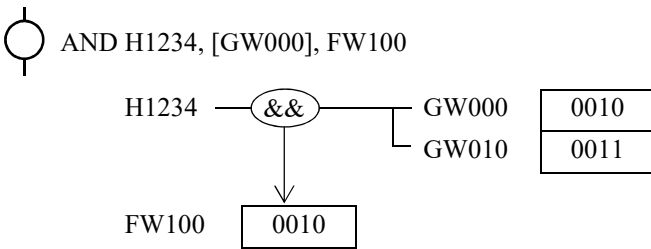
<p>DIV</p>	<p>Division</p>																																									
<p>Function description</p>	<p>This function divides the source by the contents of the destination and stores the quotient in the result.</p>																																									
<p>Parameter and operation</p>	<p> DIV S, D, R</p> <p>S: Source D: Destination R: Result</p>	<p>$S \div D \rightarrow R$</p>																																								
<p>Flag configuration</p>	<p>The E and V will change. The others will become turned off.</p>																																									
<p>Remark</p>	<p>When D = 0, the system will turn on the E flag and do nothing.</p>																																									
<p>Typical use</p>	<p> DIV FW000, FW001, FW002</p> <p>FW000: 0100 FW001: 0010 FW002: 0010</p> <p></p> <p> DIV H22, [GW000], FW100</p> <p>H022:  GW000: 0000 GW010: 0011</p> <p>FW100: 0002</p>																																									
<p>Effective parameter</p> <p>Δ is an address. Parameter error if the number is odd.</p>	<table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>-</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	-	√	√	Indirect word length	-	Δ	Δ	Indirect long length	-	Δ	Δ	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>-</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	-	Direct long length	-	√	-	Indirect word length	-	Δ	Δ	Indirect long length	-	Δ	Δ
S, D	Bit-type PI/O	Word-type PI/O	Constant																																							
Direct word length	√	√	√																																							
Direct long length	-	√	√																																							
Indirect word length	-	Δ	Δ																																							
Indirect long length	-	Δ	Δ																																							
R	Bit-type PI/O	Word-type PI/O	Constant																																							
Direct word length	√	√	-																																							
Direct long length	-	√	-																																							
Indirect word length	-	Δ	Δ																																							
Indirect long length	-	Δ	Δ																																							


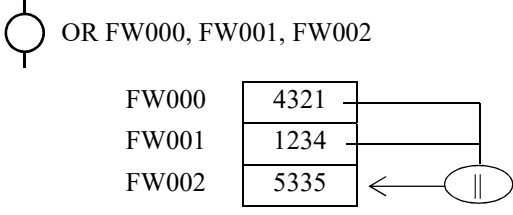
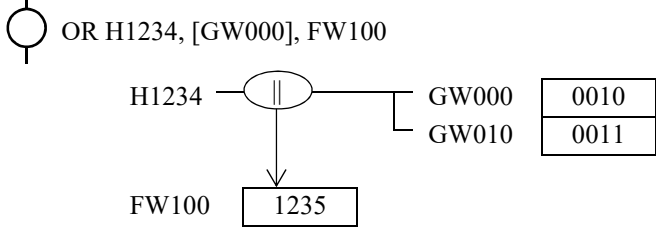
5. APPLIED INSTRUCTIONS

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 S: Source D: Destination R: Result	Remainder of $S \div D \rightarrow R$																																								
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	 																																									
Effective parameter	<table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	-	√	√	Indirect word length	-	△	△	Indirect long length	-	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	-	Direct long length	-	√	-	Indirect word length	-	△	△	Indirect long length	-	△	△
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	-	△	△																																							


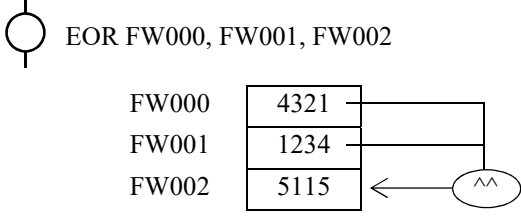
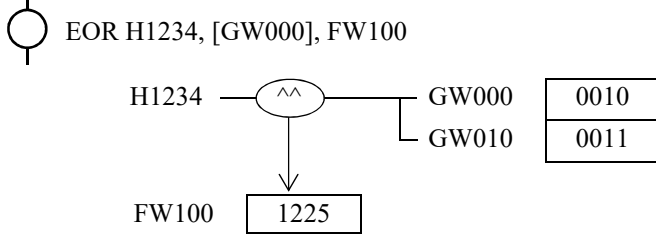
SCL	Scale conversion																						
Function description	This function converts the scale of the source by the contents of the destination and stores it in the result.																						
Parameter and operation	 SCL S, D1, D2, R	$S \times D1 \div D2 \rightarrow R$																					
	S: Source D1: Destination 1 D2: Destination 2 R: Result																						
Flag configuration	The E and V will change. The others will become turned off.																						
Remark	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.																						
Typical use	 SCL FW000, FW001, FW002, FW003 <div style="display: flex; align-items: center; margin-top: 10px;"> <div style="margin-right: 10px;">FW000</div> <div style="border: 1px solid black; padding: 2px 5px;">3320</div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> </div> <div style="margin-top: 10px;"> <div style="margin-right: 10px;">FW001</div> <div style="border: 1px solid black; padding: 2px 5px;">0010</div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> </div> <div style="margin-top: 10px;"> <div style="margin-right: 10px;">FW002</div> <div style="border: 1px solid black; padding: 2px 5px;">0066</div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> </div> <div style="margin-top: 10px;"> <div style="margin-right: 10px;">FW003</div> <div style="border: 1px solid black; padding: 2px 5px;">0805</div> <div style="margin: 0 5px;">←</div> <div style="border: 1px solid black; border-radius: 15px; padding: 2px 5px; display: inline-block;">/3320×/10÷/66</div> </div>  SCL GW000, GW001, H1110, FW100 <div style="margin-top: 10px;"> <div style="margin-right: 10px;">GW000</div> <div style="border: 1px solid black; padding: 2px 5px;">2222</div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> </div> <div style="margin-top: 10px;"> <div style="margin-right: 10px;">GW010</div> <div style="border: 1px solid black; padding: 2px 5px;">0012</div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> <div style="margin: 0 5px;">—</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 15px; width: 10px;"></div> </div> <div style="margin-top: 10px;"> <div style="margin-right: 10px;">FW100</div> <div style="border: 1px solid black; padding: 2px 5px;">0024</div> <div style="margin: 0 5px;">←</div> <div style="border: 1px solid black; border-radius: 15px; padding: 2px 5px; display: inline-block;">/2222×/12÷/1110</div> </div>																						
Effective parameter	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>S, D1, D2</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>			S, D1, D2	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	—	√	√	Indirect word length	—	△	△	Indirect long length	—	△	△
S, D1, D2	Bit-type PI/O	Word-type PI/O	Constant																				
Direct word length	√	√	√																				
Direct long length	—	√	√																				
Indirect word length	—	△	△																				
Indirect long length	—	△	△																				
△ is an address. Parameter error if the number is odd.	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>			R	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	—	△	△																				


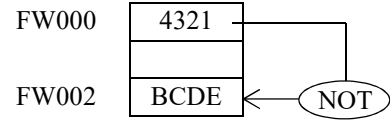
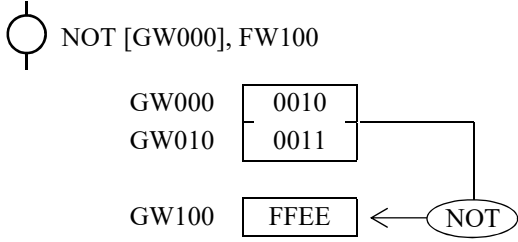
5. APPLIED INSTRUCTIONS

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	$S \ \&\& \ D \ \rightarrow \ R$																																										
	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	 																																											
Effective parameter	<table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△	<p>△ is an address. Parameter error if the number is odd.</p>	
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	$S \parallel D \rightarrow R$																																								
	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	 																																									
Effective parameter	<p>Δ is an address. Parameter error if the number is odd.</p> <table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	Δ	Δ	Indirect long length	–	Δ	Δ	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	Δ	Δ	Indirect long length	–	Δ	Δ
S, D	Bit-type PI/O	Word-type PI/O	Constant																																							
Direct word length	√	√	√																																							
Direct long length	–	√	√																																							
Indirect word length	–	Δ	Δ																																							
Indirect long length	–	Δ	Δ																																							
R	Bit-type PI/O	Word-type PI/O	Constant																																							
Direct word length	√	√	–																																							
Direct long length	–	√	–																																							
Indirect word length	–	Δ	Δ																																							
Indirect long length	–	Δ	Δ																																							



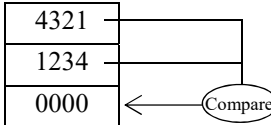

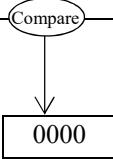
5. APPLIED INSTRUCTIONS




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 <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> $S \wedge D \rightarrow R$ </div> <p>S: Source D: Destination R: Result</p>																																								
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	 <p>EOR FW000, FW001, FW002</p> <p>FW000: 4321 FW001: 1234 FW002: 5115</p>  <p>EOR H1234, [GW000], FW100</p> <p>H1234: ^ FW100: 1225 GW000: 0010 GW010: 0011</p>																																								
Effective parameter	<p>Δ is an address. Parameter error if the number is odd.</p> <table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	Δ	Δ	Indirect long length	–	Δ	Δ	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	Δ	Δ	Indirect long length	–	Δ	Δ
S, D	Bit-type PI/O	Word-type PI/O	Constant																																						
Direct word length	√	√	√																																						
Direct long length	–	√	√																																						
Indirect word length	–	Δ	Δ																																						
Indirect long length	–	Δ	Δ																																						
R	Bit-type PI/O	Word-type PI/O	Constant																																						
Direct word length	√	√	–																																						
Direct long length	–	√	–																																						
Indirect word length	–	Δ	Δ																																						
Indirect long length	–	Δ	Δ																																						

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: Source D: Destination	S (bit reversion) → R																																								
Flag configuration	The E will change. The others will become turned off.																																									
Remark																																										
Typical use	 																																									
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>		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	–	△	△
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	–	△	△																																							



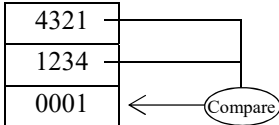

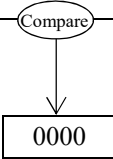
△ is an address. Parameter error if the number is odd.


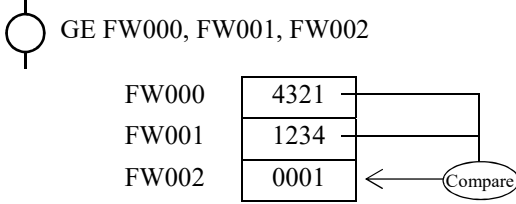
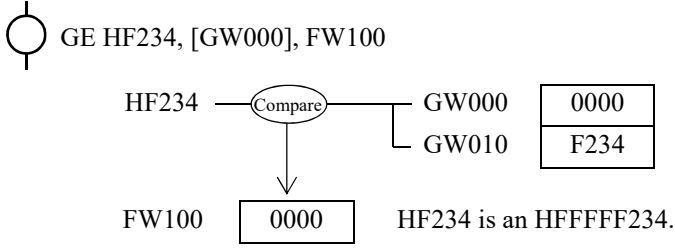
5. APPLIED INSTRUCTIONS

EQU	Compare to see if equal																																													
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	<div style="display: flex; align-items: center;"> <div style="margin-right: 20px;">  EQU S, D, R </div> <div style="border: 1px solid black; padding: 5px;"> $S = D \quad 1 \rightarrow R$ $S \neq D \quad 0 \rightarrow R$ </div> </div> <p>S: Source D: Destination R: Result</p>																																													
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	<div style="margin-bottom: 20px;">  EQU FW000, FW001, FW002 </div> <div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> <p>FW000</p> <p>FW001</p> <p>FW002</p> </div> <div style="border: 1px solid black; padding: 5px; margin-right: 10px;"> <table style="border-collapse: collapse; width: 100px;"> <tr><td style="text-align: center;">4321</td></tr> <tr><td style="text-align: center;">1234</td></tr> <tr><td style="text-align: center;">0000</td></tr> </table> </div> <div style="margin-left: 10px;">  </div> </div> <div style="margin-bottom: 20px;">  EQU HF234, [GW000], FW100 </div> <div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> <p>HF234</p> <p>FW100</p> </div> <div style="margin-right: 10px;">  </div> <div style="margin-right: 10px;"> <p>GW000</p> <p>GW010</p> </div> <div style="border: 1px solid black; padding: 5px;"> <table style="border-collapse: collapse; width: 100px;"> <tr><td style="text-align: center;">0000</td></tr> <tr><td style="text-align: center;">F234</td></tr> </table> </div> </div> <p>HF234 is an HFFFFFF234.</p>	4321	1234	0000	0000	F234																																								
4321																																														
1234																																														
0000																																														
0000																																														
F234																																														
Effective parameter	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="width: 25%;">S, D</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 15%;">Constant</th> <th style="width: 20%;"></th> <th style="width: 20%;">R</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 15%;">Constant</th> </tr> </thead> <tbody> <tr> <td>Direct word length</td> <td>√</td> <td>√</td> <td>√</td> <td></td> <td>Direct word length</td> <td>√</td> <td>√</td> <td>–</td> </tr> <tr> <td>Direct long length</td> <td>–</td> <td>√</td> <td>√</td> <td></td> <td>Direct long length</td> <td>–</td> <td>√</td> <td>–</td> </tr> <tr> <td>Indirect word length</td> <td>–</td> <td>△</td> <td>△</td> <td></td> <td>Indirect word length</td> <td>–</td> <td>△</td> <td>△</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>△</td> <td>△</td> <td></td> <td>Indirect long length</td> <td>–</td> <td>△</td> <td>△</td> </tr> </tbody> </table> <p>△ is an address. Parameter error if the number is odd.</p>	S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√		Direct word length	√	√	–	Direct long length	–	√	√		Direct long length	–	√	–	Indirect word length	–	△	△		Indirect word length	–	△	△	Indirect long length	–	△	△		Indirect long length	–	△	△
S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant																																						
Direct word length	√	√	√		Direct word length	√	√	–																																						
Direct long length	–	√	√		Direct long length	–	√	–																																						
Indirect word length	–	△	△		Indirect word length	–	△	△																																						
Indirect long length	–	△	△		Indirect long length	–	△	△																																						

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	<div style="display: flex; align-items: center;"> <div style="margin-right: 20px;">  <p>NEQ S, D, R</p> <p>S: Source D: Destination R: Result</p> </div> <div style="border: 1px solid black; padding: 5px;"> <p>$S \neq D \quad 1 \rightarrow R$ $S = D \quad 0 \rightarrow R$</p> </div> </div>																																													
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	<div style="margin-bottom: 20px;">  <p>NEQ FW000, FW001, FW002</p> <div style="display: flex; align-items: center; margin-top: 10px;"> <div style="margin-right: 10px;"> <p>FW000</p> <p>FW001</p> <p>FW002</p> </div> <div style="border: 1px solid black; padding: 5px; margin-right: 10px;"> <p>4321</p> <p>1234</p> <p>0001</p> </div> <div style="margin-right: 10px;"> <p>←</p> </div> <div style="border: 1px solid black; border-radius: 50%; padding: 2px 5px;"> <p>Compare</p> </div> </div> </div> <div>  <p>NEQ HF234, [GW000], FW100</p> <div style="display: flex; align-items: center; margin-top: 10px;"> <div style="margin-right: 10px;"> <p>HF234</p> </div> <div style="border: 1px solid black; border-radius: 50%; padding: 2px 5px; margin-right: 10px;"> <p>Compare</p> </div> <div style="margin-right: 10px;"> <p>→</p> </div> <div style="margin-right: 10px;"> <p>GW000</p> </div> <div style="border: 1px solid black; padding: 2px 5px; margin-right: 10px;"> <p>0000</p> </div> <div style="margin-right: 10px;"> <p>GW010</p> </div> <div style="border: 1px solid black; padding: 2px 5px; margin-right: 10px;"> <p>F234</p> </div> </div> <div style="margin-top: 10px; margin-left: 20px;"> <p>↓</p> </div> <div style="display: flex; align-items: center; margin-top: 10px;"> <div style="margin-right: 10px;"> <p>FW100</p> </div> <div style="border: 1px solid black; padding: 2px 5px; margin-right: 10px;"> <p>0001</p> </div> <div> <p>HF234 is an HFFFFFF234.</p> </div> </div> </div>																																													
Effective parameter	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="width: 25%;">S, D</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 15%;">Constant</th> <th style="width: 20%;"></th> <th style="width: 20%;">R</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 15%;">Constant</th> </tr> </thead> <tbody> <tr> <td>Direct word length</td> <td>√</td> <td>√</td> <td>√</td> <td></td> <td>Direct word length</td> <td>√</td> <td>√</td> <td>–</td> </tr> <tr> <td>Direct long length</td> <td>–</td> <td>√</td> <td>√</td> <td></td> <td>Direct long length</td> <td>–</td> <td>√</td> <td>–</td> </tr> <tr> <td>Indirect word length</td> <td>–</td> <td>△</td> <td>△</td> <td></td> <td>Indirect word length</td> <td>–</td> <td>△</td> <td>△</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>△</td> <td>△</td> <td></td> <td>Indirect long length</td> <td>–</td> <td>△</td> <td>△</td> </tr> </tbody> </table> <p style="font-size: small; margin-top: 10px;">△ is an address. Parameter error if the number is odd.</p>	S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√		Direct word length	√	√	–	Direct long length	–	√	√		Direct long length	–	√	–	Indirect word length	–	△	△		Indirect word length	–	△	△	Indirect long length	–	△	△		Indirect long length	–	△	△
S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant																																						
Direct word length	√	√	√		Direct word length	√	√	–																																						
Direct long length	–	√	√		Direct long length	–	√	–																																						
Indirect word length	–	△	△		Indirect word length	–	△	△																																						
Indirect long length	–	△	△		Indirect long length	–	△	△																																						

5. APPLIED INSTRUCTIONS

GT	Compare to see if larger																																													
Function description	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.																																													
Parameter and operation	 GT S, D, R <div style="float: right; border: 1px solid black; padding: 5px; margin-top: 10px;"> $S > D \quad 1 \rightarrow R$ $S \leq D \quad 0 \rightarrow R$ </div> <p>S: Source D: Destination R: Result</p>																																													
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	 GT FW000, FW001, FW002 <div style="margin-top: 10px;"> <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px;">FW000</td><td style="padding: 2px;">4321</td></tr> <tr><td style="padding: 2px;">FW001</td><td style="padding: 2px;">1234</td></tr> <tr><td style="padding: 2px;">FW002</td><td style="padding: 2px;">0001</td></tr> </table>  </div>  GT HF234, [GW000], FW100 <div style="margin-top: 10px;"> <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px;">HF234</td><td style="padding: 2px;">—</td><td style="padding: 2px;">Compare</td><td style="padding: 2px;">—</td><td style="padding: 2px;">GW000</td><td style="padding: 2px;">0000</td></tr> <tr><td></td><td></td><td></td><td style="padding: 2px;">—</td><td style="padding: 2px;">GW010</td><td style="padding: 2px;">F234</td></tr> </table>  <p>FW100 <table border="1" style="display: inline-table; border-collapse: collapse;"><tr><td style="padding: 2px;">0000</td></tr></table> HF234 is an HFFFFFF234.</p> </div>	FW000	4321	FW001	1234	FW002	0001	HF234	—	Compare	—	GW000	0000				—	GW010	F234	0000																										
FW000	4321																																													
FW001	1234																																													
FW002	0001																																													
HF234	—	Compare	—	GW000	0000																																									
			—	GW010	F234																																									
0000																																														
Effective parameter	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">S, D</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 10%;">Constant</th> <th style="width: 20%;"></th> <th style="width: 20%;">R</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 10%;">Constant</th> </tr> </thead> <tbody> <tr> <td>Direct word length</td> <td>√</td> <td>√</td> <td>√</td> <td></td> <td>Direct word length</td> <td>√</td> <td>√</td> <td>—</td> </tr> <tr> <td>Direct long length</td> <td>—</td> <td>√</td> <td>√</td> <td></td> <td>Direct long length</td> <td>—</td> <td>√</td> <td>—</td> </tr> <tr> <td>Indirect word length</td> <td>—</td> <td>△</td> <td>△</td> <td></td> <td>Indirect word length</td> <td>—</td> <td>△</td> <td>△</td> </tr> <tr> <td>Indirect long length</td> <td>—</td> <td>△</td> <td>△</td> <td></td> <td>Indirect long length</td> <td>—</td> <td>△</td> <td>△</td> </tr> </tbody> </table> <p>△ is an address. Parameter error if the number is odd.</p>	S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√		Direct word length	√	√	—	Direct long length	—	√	√		Direct long length	—	√	—	Indirect word length	—	△	△		Indirect word length	—	△	△	Indirect long length	—	△	△		Indirect long length	—	△	△
S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant																																						
Direct word length	√	√	√		Direct word length	√	√	—																																						
Direct long length	—	√	√		Direct long length	—	√	—																																						
Indirect word length	—	△	△		Indirect word length	—	△	△																																						
Indirect long length	—	△	△		Indirect long length	—	△	△																																						


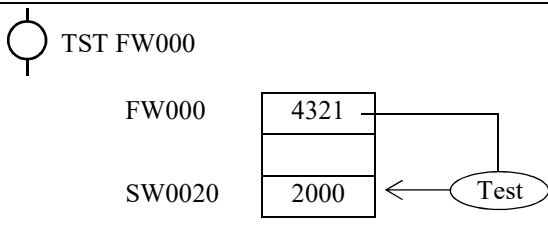
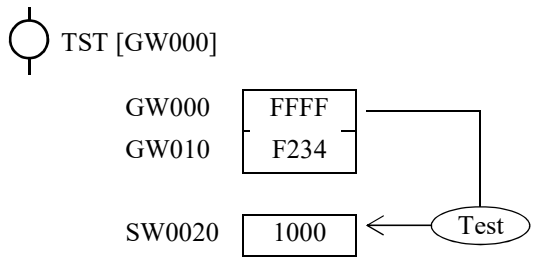
GE	Compare to see if equal or larger																																									
Function description	This function compares the source with the contents of the destination. If the source is equal or larger, this function stores a 1 in the result. If not, it stores a 0.																																									
Parameter and operation	 GE S, D, R S: Source D: Destination R: Result	$S \geq D \quad 1 \rightarrow R$ $S < D \quad 0 \rightarrow R$																																								
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	 																																									
Effective parameter	<p>Δ is an address. Parameter error if the number is odd.</p> <table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	Δ	Δ	Indirect long length	–	Δ	Δ	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	Δ	Δ	Indirect long length	–	Δ	Δ
S, D	Bit-type PI/O	Word-type PI/O	Constant																																							
Direct word length	√	√	√																																							
Direct long length	–	√	√																																							
Indirect word length	–	Δ	Δ																																							
Indirect long length	–	Δ	Δ																																							
R	Bit-type PI/O	Word-type PI/O	Constant																																							
Direct word length	√	√	–																																							
Direct long length	–	√	–																																							
Indirect word length	–	Δ	Δ																																							
Indirect long length	–	Δ	Δ																																							



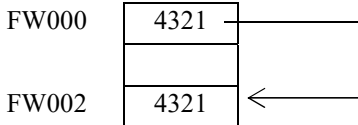

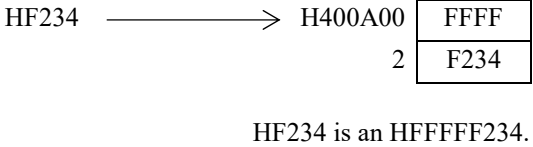
5. APPLIED INSTRUCTIONS

LT	Compare to see if smaller																																								
Function description	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.																																								
Parameter and operation	<div style="display: flex; justify-content: space-between; align-items: flex-start;"> <div style="width: 60%;"> <p>LT S, D, R</p> <p>S: Source D: Destination R: Result</p> </div> <div style="width: 35%; border: 1px solid black; padding: 5px;"> <p>$S < D \quad 1 \rightarrow R$ $S \geq D \quad 0 \rightarrow R$</p> </div> </div>																																								
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	<div style="display: flex; flex-direction: column; align-items: center;"> <div style="margin-bottom: 20px;"> <p>LT FW000, FW001, FW002</p> <table style="border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 5px;">FW000</td> <td style="border: 1px solid black; padding: 5px;">4321</td> <td rowspan="3" style="padding: 0 10px;">←</td> <td rowspan="3" style="border: 1px solid black; border-radius: 50%; padding: 5px;">Compare</td> </tr> <tr> <td style="padding: 5px;">FW001</td> <td style="border: 1px solid black; padding: 5px;">1234</td> </tr> <tr> <td style="padding: 5px;">FW002</td> <td style="border: 1px solid black; padding: 5px;">0000</td> </tr> </table> </div> <div> <p>LT HF234, [GW000], FW100</p> <table style="border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 5px;">HF234</td> <td style="border: 1px solid black; border-radius: 50%; padding: 5px;">Compare</td> <td style="padding: 0 10px;">→</td> <td style="padding: 5px;">GW000</td> <td style="border: 1px solid black; padding: 5px;">0000</td> </tr> <tr> <td></td> <td></td> <td style="padding: 0 10px;">→</td> <td style="padding: 5px;">GW010</td> <td style="border: 1px solid black; padding: 5px;">F234</td> </tr> <tr> <td style="padding: 5px;">FW100</td> <td style="border: 1px solid black; padding: 5px;">0001</td> <td style="padding: 0 10px;">↓</td> <td></td> <td></td> </tr> </table> <p>HF234 is an HFFFFFF234.</p> </div> </div>	FW000	4321	←	Compare	FW001	1234	FW002	0000	HF234	Compare	→	GW000	0000			→	GW010	F234	FW100	0001	↓																			
FW000	4321	←	Compare																																						
FW001	1234																																								
FW002	0000																																								
HF234	Compare	→	GW000	0000																																					
		→	GW010	F234																																					
FW100	0001	↓																																							
Effective parameter	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="width: 15%;">S, D</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 15%;">Constant</th> <th style="width: 15%;">R</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 15%;">Constant</th> </tr> </thead> <tbody> <tr> <td>Direct word length</td> <td>√</td> <td>√</td> <td>√</td> <td>Direct word length</td> <td>√</td> <td>√</td> <td>–</td> </tr> <tr> <td>Direct long length</td> <td>–</td> <td>√</td> <td>√</td> <td>Direct long length</td> <td>–</td> <td>√</td> <td>–</td> </tr> <tr> <td>Indirect word length</td> <td>–</td> <td>△</td> <td>△</td> <td>Indirect word length</td> <td>–</td> <td>△</td> <td>△</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>△</td> <td>△</td> <td>Indirect long length</td> <td>–</td> <td>△</td> <td>△</td> </tr> </tbody> </table> <p>△ is an address. Parameter error if the number is odd.</p>	S, D	Bit-type PI/O	Word-type PI/O	Constant	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct word length	√	√	–	Direct long length	–	√	√	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect word length	–	△	△	Indirect long length	–	△	△	Indirect long length	–	△	△
S, D	Bit-type PI/O	Word-type PI/O	Constant	R	Bit-type PI/O	Word-type PI/O	Constant																																		
Direct word length	√	√	√	Direct word length	√	√	–																																		
Direct long length	–	√	√	Direct long length	–	√	–																																		
Indirect word length	–	△	△	Indirect word length	–	△	△																																		
Indirect long length	–	△	△	Indirect long length	–	△	△																																		


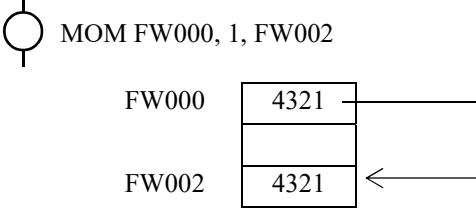
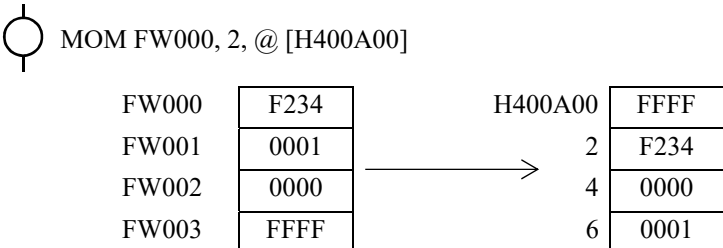
LE	Compare to see if equal or smaller																																								
Function description	This function compares the source with the contents of the destination. If the source is equal or smaller, this function stores a 1 in the result. If not, it stores a 0.																																								
Parameter and operation	<div style="display: flex; align-items: center;"> <div style="margin-right: 20px;"> LE S, D, R S: Source D: Destination R: Result </div> <div style="border: 1px solid black; padding: 5px;"> $S \leq D \quad 1 \rightarrow R$ $S > D \quad 0 \rightarrow R$ </div> </div>																																								
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	<div style="margin-bottom: 20px;"> LE FW000, FW001, FW002 </div> <div> LE HF234, [GW000], FW100 </div>																																								
Effective parameter	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Δ is an address. Parameter error if the number is odd.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table> </div> <div style="width: 45%;"> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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>Δ</td> <td>Δ</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>Δ</td> <td>Δ</td> </tr> </tbody> </table> </div> </div>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	Δ	Δ	Indirect long length	–	Δ	Δ	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	Δ	Δ	Indirect long length	–	Δ	Δ
S, D	Bit-type PI/O	Word-type PI/O	Constant																																						
Direct word length	√	√	√																																						
Direct long length	–	√	√																																						
Indirect word length	–	Δ	Δ																																						
Indirect long length	–	Δ	Δ																																						
R	Bit-type PI/O	Word-type PI/O	Constant																																						
Direct word length	√	√	–																																						
Direct long length	–	√	–																																						
Indirect word length	–	Δ	Δ																																						
Indirect long length	–	Δ	Δ																																						




5. APPLIED INSTRUCTIONS

Flag	Code test																						
Function description	This function tests the contents of the source and configures the flags P, Z, and N.																						
Parameter and operation	 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																					
Flag configuration	The E, P, Z and N will change. The others will become turned off.																						
Remark	Word-length data is code-extended to long length and tested.																						
Typical use	 																						
Effective parameter	<table border="1"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <p>△ is an address. Parameter error if the number is odd.</p>			S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△
S	Bit-type PI/O	Word-type PI/O	Constant																				
Direct word length	√	√	√																				
Direct long length	–	√	√																				
Indirect word length	–	△	△																				
Indirect long length	–	△	△																				


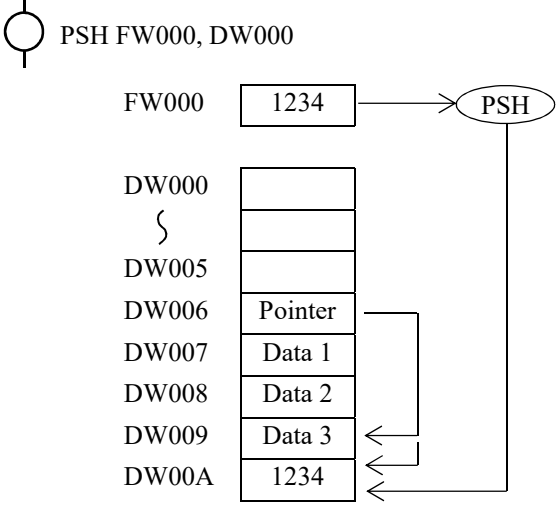
MOV	Transfer																																								
Function description	This function transfers the contents of the source to the destination.																																								
Parameter and operation	 MOV S, D <table border="1" style="float: right; margin-left: 20px;"> <tr> <td>S → D</td> </tr> </table> <p>S: Source D: Destination</p>	S → D																																							
S → D																																									
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, @[H400A00]  <p style="text-align: center;">HF234 is an HFFFFFF234.</p>																																								
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <p>△ is an address. Parameter error if the number is odd.</p>	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	–	△	△
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	–	△	△																																						

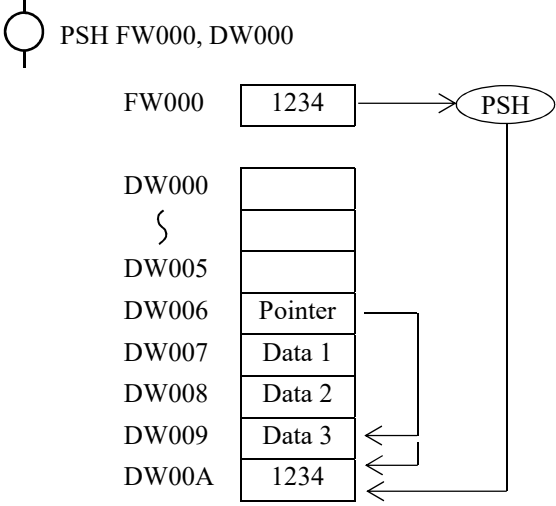
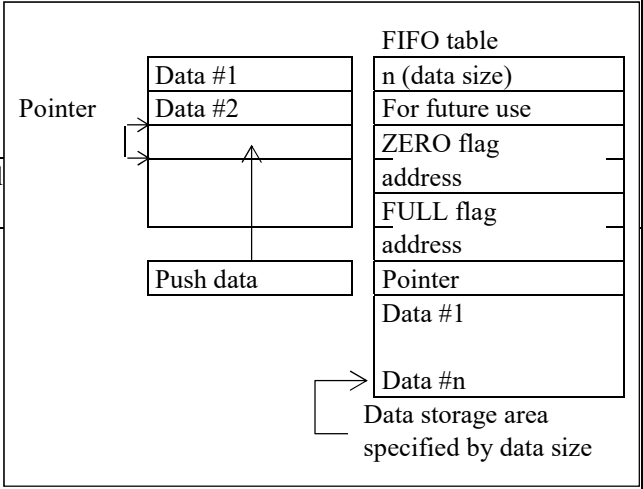
5. APPLIED INSTRUCTIONS


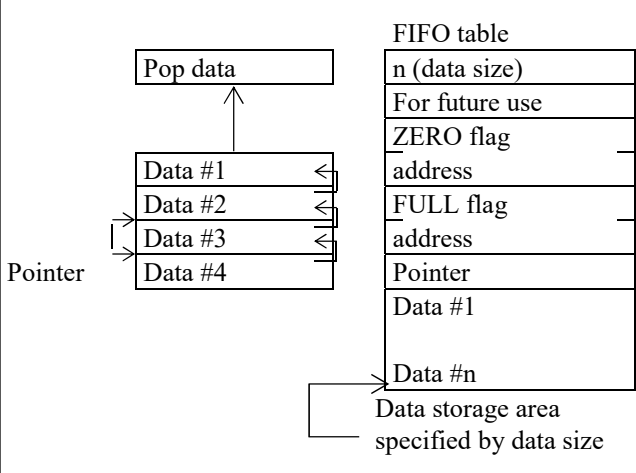
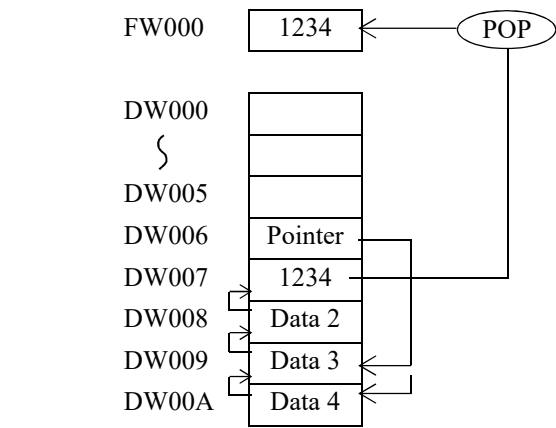
MOM	Collective transfer																																											
Function description	This function transfers the contents of the source to the destination by sending n elements worth (word and long) collectively.																																											
Parameter and operation	 MOM S, n, D S: Source D: Destination n: Number of elements to be transferred	$S_1 \rightarrow D_1$ $S_n \rightarrow D_n$																																										
Flag configuration	The E and V will change. The others will become turned off.																																											
Remark	The system will conduct no operation when $n \leq 0$ 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	  HF234 is an HFFFFFF234, H0001 is an H00000001.																																											
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>				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	–	△	△
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	–	△	△																																									

EXC	Replacement																											
Function description	This function replaces the contents of the source with the destination.																											
Parameter and operation	 EXC S, D <div style="float: right; border: 1px solid black; padding: 5px; margin-top: 10px;">S ↔ D</div> <p>S: Source D: Destination</p>																											
Flag configuration	The E and V will change. The others will become turned off.																											
Remark	If sizes differ in transfer, the system will convert their types and replace them.																											
Typical use	 EXC FW000, FW002 <div style="margin-top: 10px;"> <table style="border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">FW000</td> <td style="border: 1px solid black; padding: 2px 10px; text-align: center;">1234</td> <td style="padding: 0 5px;">←</td> <td rowspan="2" style="border-left: 1px solid black; border-right: 1px solid black; height: 20px; width: 10px;"></td> </tr> <tr> <td style="padding-right: 10px;">FW002</td> <td style="border: 1px solid black; padding: 2px 10px; text-align: center;">4321</td> <td style="padding: 0 5px;">←</td> </tr> </table> </div>  EXC @H400B00, @[H400A00] <div style="margin-top: 10px;"> <table style="border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">H400B00</td> <td style="border: 1px solid black; padding: 2px 10px; text-align: center;">F234</td> <td style="padding: 0 10px;">↔</td> <td style="padding-right: 10px;">H400A00</td> <td style="border: 1px solid black; padding: 2px 10px; text-align: center;">0010</td> </tr> <tr> <td></td> <td></td> <td></td> <td style="padding-right: 10px;">2</td> <td style="border: 1px solid black; padding: 2px 10px; text-align: center;">0001</td> </tr> </table> <p style="margin-top: 10px;">After replacement</p> <table style="border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">H400B00</td> <td style="border: 1px solid black; padding: 2px 10px; text-align: center;">7FFF</td> <td style="padding: 0 40px;"></td> <td style="padding-right: 10px;">H400A00</td> <td style="border: 1px solid black; padding: 2px 10px; text-align: center;">FFFF</td> </tr> <tr> <td></td> <td></td> <td></td> <td style="padding-right: 10px;">2</td> <td style="border: 1px solid black; padding: 2px 10px; text-align: center;">F234</td> </tr> </table> </div>	FW000	1234	←		FW002	4321	←	H400B00	F234	↔	H400A00	0010				2	0001	H400B00	7FFF		H400A00	FFFF				2	F234
FW000	1234	←																										
FW002	4321	←																										
H400B00	F234	↔	H400A00	0010																								
			2	0001																								
H400B00	7FFF		H400A00	FFFF																								
			2	F234																								
Effective parameter	<p>△ is an address. Parameter error if the number is odd.</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="width: 25%;">S, D</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 45%;">Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△							
S, D	Bit-type PI/O	Word-type PI/O	Constant																									
Direct word length	√	√	–																									
Direct long length	–	√	–																									
Indirect word length	–	△	△																									
Indirect long length	–	△	△																									


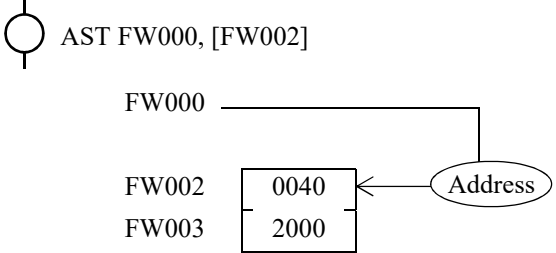
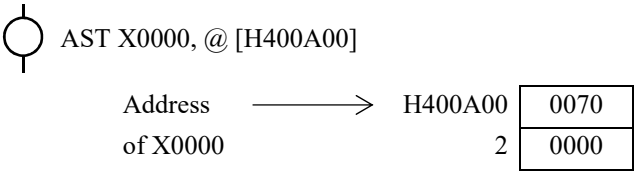
5. APPLIED INSTRUCTIONS


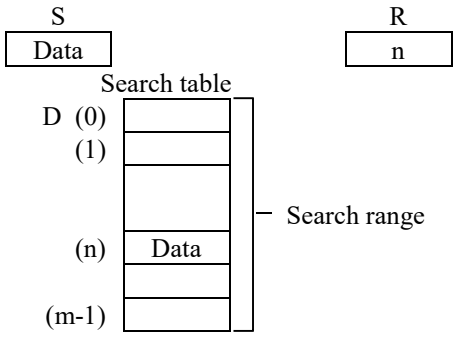

PSH	FIFO write																																								
Function description	This function pushes the contents of the source to the FIFO table. The data length of the FIFO table is word only.																																								
Parameter and operation	 PSH S, TB S: Source TB: Starting address of the FIFO table																																								
Flag configuration	The E and V will change. The others will become turned off.																																								
Remark	<p>The system will perform no operation when $n \leq 0$ and $n > 256$. It will perform no operation when the pointer < 0 or the data size $< \text{pointer}$. 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.</p> <p>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.</p>																																								
Typical use																																									
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>TB</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	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	-	△	△
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	-	△	△																																						




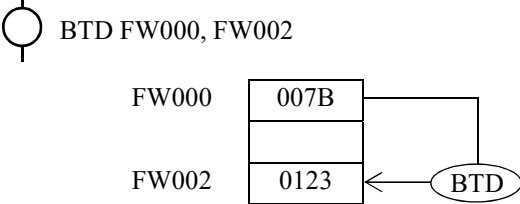
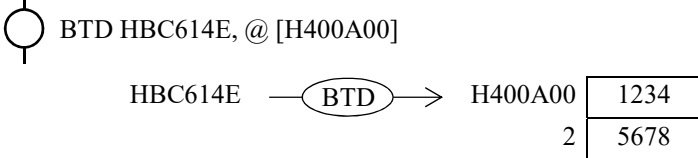
POP	FIFO read																																										
Function description	This function pops the FIFO table and stores pop data in the destination. The data length of the FIFO table is word only.																																										
Parameter and operation	 POP TB, D D: Destination TB: Starting 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 $n \leq 0$ 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. After the pop, and when 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.																																										
Typical use																																											
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>TB</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>			TB	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	-	△	△
TB	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	-	△	△																																								
△ is an address. Parameter error if the number is odd.																																											


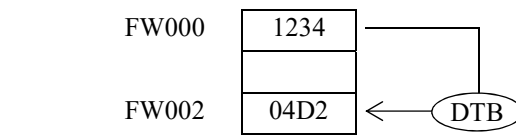
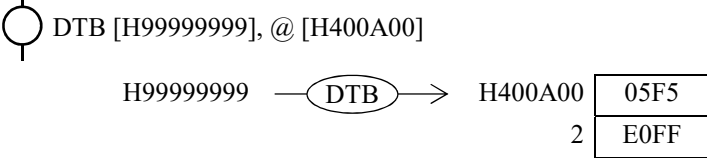
5. APPLIED INSTRUCTIONS

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 S: Source D: Destination	Address of S → D																																									
Flag configuration	The E will change. The others will become turned off.																																										
Remark																																											
Typical use	 																																										
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <tr> <td>Direct word length</td> <td style="text-align: center;">√</td> <td style="text-align: center;">√</td> <td style="text-align: center;">-</td> </tr> <tr> <td>Direct long length</td> <td style="text-align: center;">-</td> <td style="text-align: center;">√</td> <td style="text-align: center;">-</td> </tr> <tr> <td>Indirect word length</td> <td style="text-align: center;">-</td> <td style="text-align: center;">-</td> <td style="text-align: center;">-</td> </tr> <tr> <td>Indirect long length</td> <td style="text-align: center;">-</td> <td style="text-align: center;">-</td> <td style="text-align: center;">-</td> </tr> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <tr> <td>Direct word length</td> <td style="text-align: center;">-</td> <td style="text-align: center;">-</td> <td style="text-align: center;">-</td> </tr> <tr> <td>Direct long length</td> <td style="text-align: center;">-</td> <td style="text-align: center;">√</td> <td style="text-align: center;">-</td> </tr> <tr> <td>Indirect word length</td> <td style="text-align: center;">-</td> <td style="text-align: center;">-</td> <td style="text-align: center;">-</td> </tr> <tr> <td>Indirect long length</td> <td style="text-align: center;">-</td> <td style="text-align: center;">△</td> <td style="text-align: center;">△</td> </tr> </tbody> </table>			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	-	△	△
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	-	△	△																																								
△ is an address. Parameter error if the number is odd.																																											


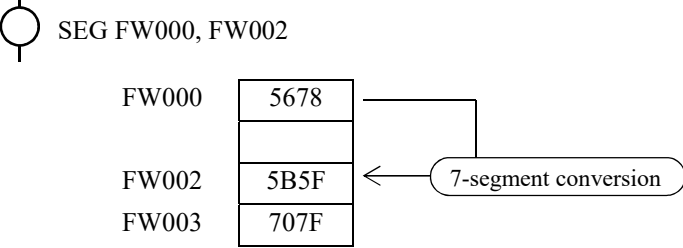
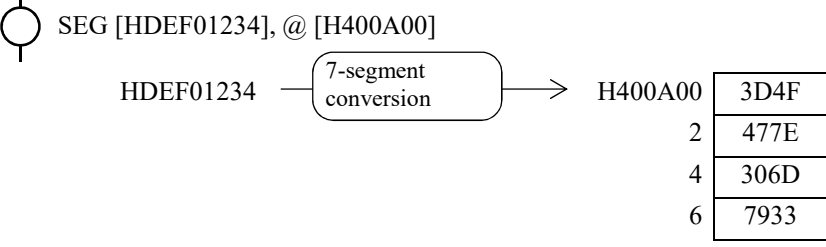
SCH	Search																																										
Function description	This function searches the range of a specified distance (in meters) from the destination for the contents of the source and stores in the result the number (n) of steps from the destination.																																										
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 $m \leq 0$ 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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																										
Typical use	 SCH DW000, FW000, 5, FW005 DW000 1234 → Search FW000 0000 FW001 1234 FW002 0000 FW003 1234 FW004 0000 FW005 0001 ← First																																										
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S, m</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>D, R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>			S, m	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	-	√	√	Indirect word length	-	△	△	Indirect long length	-	△	△	D, R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	-	Direct long length	-	√	-	Indirect word length	-	△	△	Indirect long length	-	△	△
S, m	Bit-type PI/O	Word-type PI/O	Constant																																								
Direct word length	√	√	√																																								
Direct long length	-	√	√																																								
Indirect word length	-	△	△																																								
Indirect long length	-	△	△																																								
D, R	Bit-type PI/O	Word-type PI/O	Constant																																								
Direct word length	√	√	-																																								
Direct long length	-	√	-																																								
Indirect word length	-	△	△																																								
Indirect long length	-	△	△																																								
	△ is an address. Parameter error if the number is odd.																																										


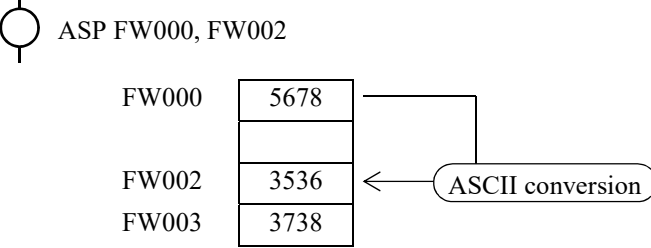
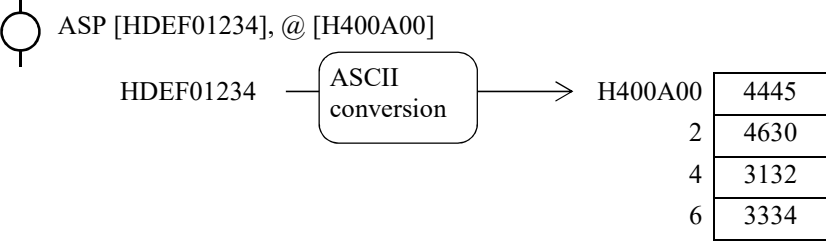
5. APPLIED INSTRUCTIONS

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	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> S (binary) → R (BCD) </div>																																									
	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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																										
Typical use	 																																										
Effective parameter	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">S</th> <th style="width: 12.5%;">Bit-type PI/O</th> <th style="width: 12.5%;">Word-type PI/O</th> <th style="width: 12.5%;">Constant</th> <th style="width: 25%;">R</th> <th style="width: 12.5%;">Bit-type PI/O</th> <th style="width: 12.5%;">Word-type PI/O</th> <th style="width: 12.5%;">Constant</th> </tr> </thead> <tbody> <tr> <td>Direct word length</td> <td>√</td> <td>√</td> <td>√</td> <td>Direct word length</td> <td>√</td> <td>√</td> <td>–</td> </tr> <tr> <td>Direct long length</td> <td>–</td> <td>√</td> <td>√</td> <td>Direct long length</td> <td>–</td> <td>√</td> <td>–</td> </tr> <tr> <td>Indirect word length</td> <td>–</td> <td>△</td> <td>△</td> <td>Indirect word length</td> <td>–</td> <td>△</td> <td>△</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>△</td> <td>△</td> <td>Indirect long length</td> <td>–</td> <td>△</td> <td>△</td> </tr> </tbody> </table>			S	Bit-type PI/O	Word-type PI/O	Constant	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct word length	√	√	–	Direct long length	–	√	√	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect word length	–	△	△	Indirect long length	–	△	△	Indirect long length	–	△	△
S	Bit-type PI/O	Word-type PI/O	Constant	R	Bit-type PI/O	Word-type PI/O	Constant																																				
Direct word length	√	√	√	Direct word length	√	√	–																																				
Direct long length	–	√	√	Direct long length	–	√	–																																				
Indirect word length	–	△	△	Indirect word length	–	△	△																																				
Indirect long length	–	△	△	Indirect long length	–	△	△																																				
	△ is an address. Parameter error if the number is odd.																																										






DTB	BCD → binary conversion																																									
Function description	This function converts the contents of the source from BCD to binary and stores them in the result.																																									
Parameter and operation	 DTB S, R S: Source R: Result	S (BCD) → R (binary)																																								
Flag configuration	The E and V will change. The others will become turned off.																																									
Remark	When anything between A and F is used in S, the E flag will be turned on and the system will perform no operation. Do not specify bit-type PI/O for R because the result will not be set correctly.																																									
Typical use	 																																									
Effective parameter	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																							


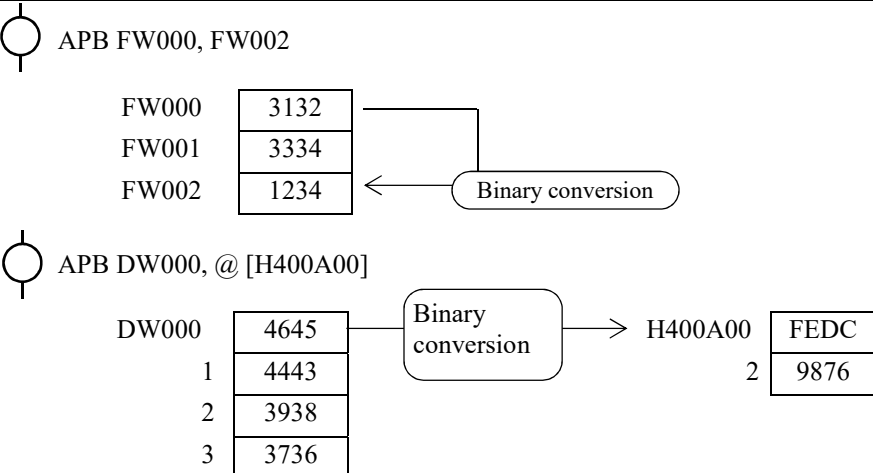


5. APPLIED INSTRUCTIONS

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: Source R: Result	S (binary) → R (7-segment data)																																								
Flag configuration	The E will change. The others will become turned off.																																									
Remark	The size × 2 of the S is written in the R. Do not specify bit-type PI/O for R because the result will not be set correctly.																																									
Typical use	  <p>[7-segment table]</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>No.</th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> <th>6</th> <th>7</th> <th>8</th> <th>9</th> <th>A</th> <th>B</th> <th>C</th> <th>D</th> <th>E</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>Data</td> <td>7E</td> <td>30</td> <td>6D</td> <td>79</td> <td>33</td> <td>5B</td> <td>5F</td> <td>70</td> <td>7F</td> <td>7B</td> <td>77</td> <td>1F</td> <td>4E</td> <td>3D</td> <td>4F</td> <td>47</td> </tr> </tbody> </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						
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	<p>△ is an address. Parameter error if the number is odd.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																							


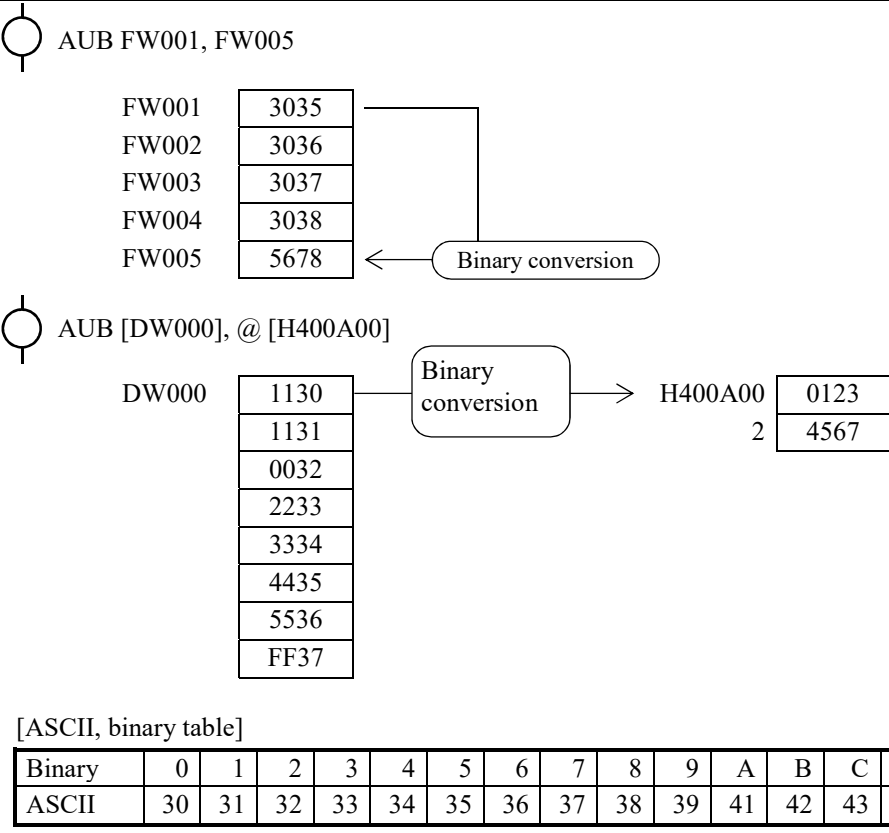


ASP	Binary → ASCII conversion pack mode																																										
Function description	This function converts the contents of the source from binary to ASCII data and stores them in the result in the pack mode.																																										
Parameter and operation	 ASP S, 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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																										
Typical use	  [ASCII, binary table] <table border="1" data-bbox="375 1220 1412 1299"> <thead> <tr> <th>Binary</th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> <th>6</th> <th>7</th> <th>8</th> <th>9</th> <th>A</th> <th>B</th> <th>C</th> <th>D</th> <th>E</th> <th>F</th> </tr> </thead> <tbody> <tr> <th>ASCII</th> <td>30</td> <td>31</td> <td>32</td> <td>33</td> <td>34</td> <td>35</td> <td>36</td> <td>37</td> <td>38</td> <td>39</td> <td>41</td> <td>42</td> <td>43</td> <td>44</td> <td>45</td> <td>46</td> </tr> </tbody> </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							
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.	<table border="1" data-bbox="359 1377 869 1792"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1" data-bbox="917 1377 1428 1792"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																								


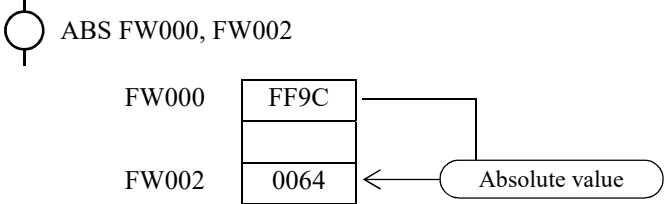
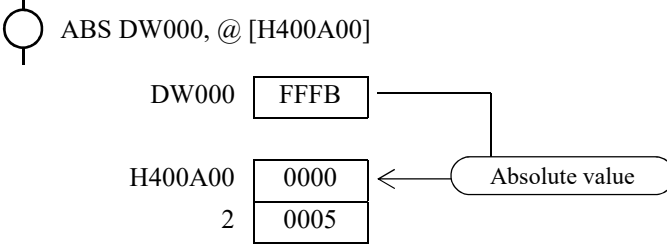
5. APPLIED INSTRUCTIONS

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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																																																								
Typical use	 ASU FW001, FW002 <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>FW001</td><td>5678</td></tr> <tr><td>FW002</td><td>3035</td></tr> <tr><td>FW003</td><td>3036</td></tr> <tr><td>FW004</td><td>3037</td></tr> <tr><td>FW005</td><td>3038</td></tr> </table>   ASU [HDEF01234], @[H400A00] <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>HDEF01234</td><td>→ ASCII conversion →</td><td>H400A00</td></tr> <tr><td></td><td></td><td>3044</td></tr> <tr><td></td><td></td><td>2 3045</td></tr> <tr><td></td><td></td><td>4 3046</td></tr> <tr><td></td><td></td><td>6 3030</td></tr> <tr><td></td><td></td><td>8 3031</td></tr> <tr><td></td><td></td><td>A 3032</td></tr> <tr><td></td><td></td><td>C 3033</td></tr> <tr><td></td><td></td><td>E 3034</td></tr> </table> [ASCII, binary table] <table border="1" style="width: 100%; text-align: center;"> <tr> <td>Binary</td> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td> </tr> <tr> <td>ASCII</td> <td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td> </tr> </table>		FW001	5678	FW002	3035	FW003	3036	FW004	3037	FW005	3038	HDEF01234	→ ASCII conversion →	H400A00			3044			2 3045			4 3046			6 3030			8 3031			A 3032			C 3033			E 3034	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
FW001	5678																																																																								
FW002	3035																																																																								
FW003	3036																																																																								
FW004	3037																																																																								
FW005	3038																																																																								
HDEF01234	→ ASCII conversion →	H400A00																																																																							
		3044																																																																							
		2 3045																																																																							
		4 3046																																																																							
		6 3030																																																																							
		8 3031																																																																							
		A 3032																																																																							
		C 3033																																																																							
		E 3034																																																																							
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.	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△																														
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	–	△	△																																																																						



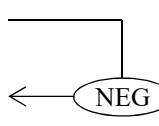

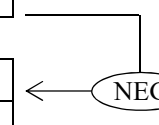
APB	ASCII → binary conversion pack mode																																									
Function description	This function converts the contents of the source from ASCII data (pack mode) to binary and stores them in the result.																																									
Parameter and operation	 APB S, R S: Source 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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																									
Typical use	 <p>  APB FW000, FW002 FW000 3132 FW001 3334 FW002 1234 Binary conversion </p> <p>  APB DW000, @ [H400A00] DW000 4645 1 4443 2 3938 3 3736 Binary conversion → H400A00 2 FEDC 9876 </p> <p>[ASCII, binary table]</p> <table border="1"> <thead> <tr> <th>Binary</th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> <th>6</th> <th>7</th> <th>8</th> <th>9</th> <th>A</th> <th>B</th> <th>C</th> <th>D</th> <th>E</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>ASCII</td> <td>30</td> <td>31</td> <td>32</td> <td>33</td> <td>34</td> <td>35</td> <td>36</td> <td>37</td> <td>38</td> <td>39</td> <td>41</td> <td>42</td> <td>43</td> <td>44</td> <td>45</td> <td>46</td> </tr> </tbody> </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						
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	<p>△ is an address. Parameter error if the number is odd.</p> <table border="1"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	–	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																							


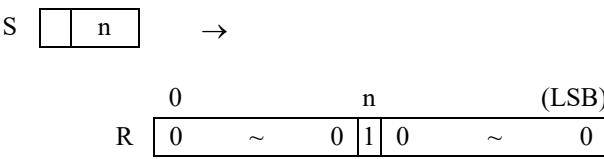
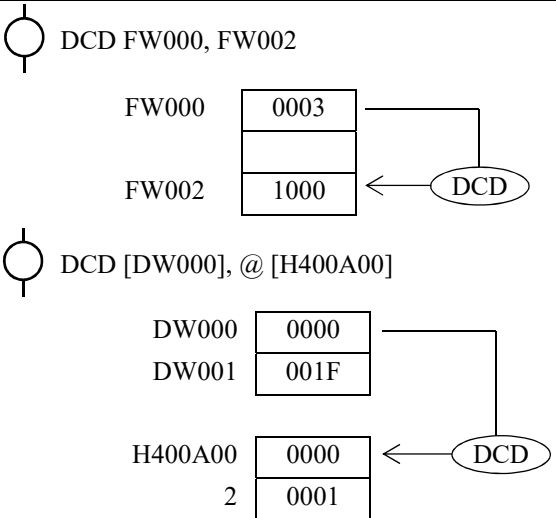
5. APPLIED INSTRUCTIONS

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. If a 16-bit value is specified for S, the eight low-level bits are effective. Do not specify bit-type PI/O for R because the result will not be set correctly.																																									
Typical use	 <p>  AUB FW001, FW005 FW001 3035 FW002 3036 FW003 3037 FW004 3038 FW005 5678 Binary conversion </p> <p>  AUB [DW000], @ [H400A00] DW000 1130 1131 0032 2233 3334 4435 5536 FF37 Binary conversion → H400A00 0123 2 4567 </p> <p>[ASCII, binary table]</p> <table border="1"> <tr> <td>Binary</td> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td> </tr> <tr> <td>ASCII</td> <td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td> </tr> </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						
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	<p>△ is an address. Parameter error if the number is odd.</p> <table border="1"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	–	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																							

ABS	Absolute value																																										
Function description	This function stores the absolute values in the source in the result.																																										
Parameter and operation	 ABS S, R S: Source R: Result	$ S \rightarrow R$																																									
Flag configuration	The E and V will change. The others will become turned off.																																										
Remark	When overflowed, the system will set the result to H7FFFFFFF. Do not specify bit-type PI/O for R because the result will not be set correctly.																																										
Typical use	 																																										
Effective parameter	<p>△ is an address. Parameter error if the number is odd.</p> <table border="1"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>		R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																								

5. APPLIED INSTRUCTIONS

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 <div style="float: right; border: 1px solid black; padding: 5px; margin-left: 20px;">-S → R</div> <p>S: Source R: Result</p>																																								
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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																								
Typical use	 NEG FW000, FW002 <div style="margin-left: 40px;"> <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>FW000</td><td>1000</td></tr> <tr><td> </td><td> </td></tr> <tr><td>FW002</td><td>F000</td></tr> </table>  </div>  NEG DW000, @[H400A00] <div style="margin-left: 40px;"> <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>DW000</td><td>1234</td></tr> </table> <table border="1" style="display: inline-table;"> <tr><td>H400A00</td><td>FFFF</td></tr> <tr><td>2</td><td>EDCC</td></tr> </table>  </div>	FW000	1000			FW002	F000	DW000	1234	H400A00	FFFF	2	EDCC																												
FW000	1000																																								
FW002	F000																																								
DW000	1234																																								
H400A00	FFFF																																								
2	EDCC																																								
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	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	–	△	△
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	–	△	△																																						


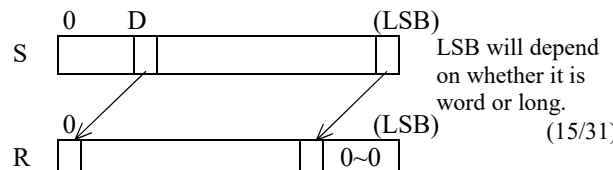
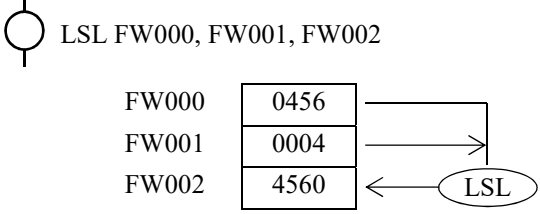
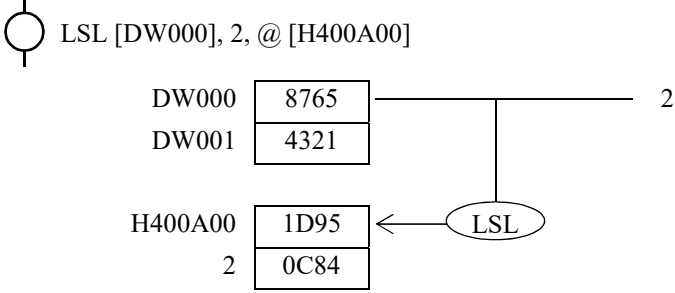
DCD	Decode																																										
Function description	This function decodes the contents of the source and stores the finding in the result.																																										
Parameter and operation	 DCD S, R S: Source R: Result	 <p>With n specified in S, the system will turn on the bit of the bit number n as counted from the MSB of R (counted from 0 onwards).</p>																																									
Flag configuration	The E will change. The others will become turned off.																																										
Remark	The effective bit of S will be four low-level bits when R is word length and five low-level bits when it is long length. Do not specify bit-type PI/O for R because the result will not be set correctly.																																										
Typical use																																											
Effective parameter	<table border="1"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△	
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	–	△	△																																								


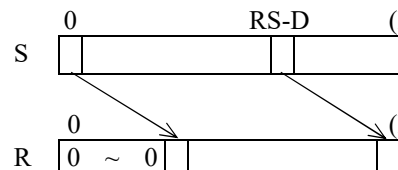


5. APPLIED INSTRUCTIONS

ECD	Encode																																								
Function description	This function encodes the contents of the source and stores the finding in the result.																																								
Parameter and operation	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 20px;"> <p>⊙ ECD S, R</p> <p>S: Source R: Result</p> </div> <div style="flex-grow: 1;"> <p style="text-align: right;">(LSB)</p> <p>S <table border="1" style="display: inline-table; border-collapse: collapse; text-align: center;"> <tr> <td style="width: 10%; padding: 2px 5px;">0</td> <td style="width: 10%; padding: 2px 5px;">~</td> <td style="width: 10%; padding: 2px 5px;">0</td> <td style="width: 10%; padding: 2px 5px;">1</td> <td style="width: 10%; padding: 2px 5px;">?</td> <td style="width: 10%; padding: 2px 5px;">~</td> <td style="width: 10%; padding: 2px 5px;">?</td> </tr> </table></p> <p style="text-align: center;">→ R <table border="1" style="display: inline-table; border-collapse: collapse; text-align: center;"> <tr> <td style="width: 10%; padding: 2px 5px;"></td> <td style="width: 10%; padding: 2px 5px;">n</td> </tr> </table></p> <p>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.</p> </div> </div>	0	~	0	1	?	~	?		n																															
0	~	0	1	?	~	?																																			
	n																																								
Flag configuration	The E will change. The others will become turned off.																																								
Remark	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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																								
Typical use	<div style="margin-bottom: 20px;"> <p>⊙ ECD FW000, FW002</p> <div style="display: flex; align-items: center; margin-left: 40px;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px 5px;">FW000</td><td style="padding: 2px 5px;">0456</td></tr> <tr><td style="padding: 2px 5px;">FW002</td><td style="padding: 2px 5px;">0005</td></tr> </table> <div style="margin-left: 10px;"> <p>← ⊙ ECD</p> </div> </div> </div> <div> <p>⊙ ECD [DW000], @[H400A00]</p> <div style="display: flex; align-items: center; margin-left: 40px;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px 5px;">DW000</td><td style="padding: 2px 5px;">0000</td></tr> <tr><td style="padding: 2px 5px;">DW001</td><td style="padding: 2px 5px;">0080</td></tr> </table> <div style="margin-left: 10px;"> <p>← ⊙ ECD</p> </div> </div> <div style="margin-left: 40px; margin-top: 10px;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px 5px;">H400A00</td><td style="padding: 2px 5px;">0000</td></tr> <tr><td style="padding: 2px 5px;">2</td><td style="padding: 2px 5px;">0018</td></tr> </table> <p>← ⊙ ECD</p> </div> </div>	FW000	0456	FW002	0005	DW000	0000	DW001	0080	H400A00	0000	2	0018																												
FW000	0456																																								
FW002	0005																																								
DW000	0000																																								
DW001	0080																																								
H400A00	0000																																								
2	0018																																								
Effective parameter	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="width: 15%;">S</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 15%;">Constant</th> <th style="width: 15%;">R</th> <th style="width: 15%;">Bit-type PI/O</th> <th style="width: 15%;">Word-type PI/O</th> <th style="width: 15%;">Constant</th> </tr> </thead> <tbody> <tr> <td>Direct word length</td> <td>√</td> <td>√</td> <td>√</td> <td>Direct word length</td> <td>√</td> <td>√</td> <td>–</td> </tr> <tr> <td>Direct long length</td> <td>–</td> <td>√</td> <td>√</td> <td>Direct long length</td> <td>–</td> <td>√</td> <td>–</td> </tr> <tr> <td>Indirect word length</td> <td>–</td> <td>△</td> <td>△</td> <td>Indirect word length</td> <td>–</td> <td>△</td> <td>△</td> </tr> <tr> <td>Indirect long length</td> <td>–</td> <td>△</td> <td>△</td> <td>Indirect long length</td> <td>–</td> <td>△</td> <td>△</td> </tr> </tbody> </table> <p>△ is an address. Parameter error if the number is odd.</p>	S	Bit-type PI/O	Word-type PI/O	Constant	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct word length	√	√	–	Direct long length	–	√	√	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect word length	–	△	△	Indirect long length	–	△	△	Indirect long length	–	△	△
S	Bit-type PI/O	Word-type PI/O	Constant	R	Bit-type PI/O	Word-type PI/O	Constant																																		
Direct word length	√	√	√	Direct word length	√	√	–																																		
Direct long length	–	√	√	Direct long length	–	√	–																																		
Indirect word length	–	△	△	Indirect word length	–	△	△																																		
Indirect long length	–	△	△	Indirect long length	–	△	△																																		


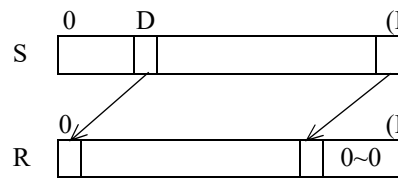
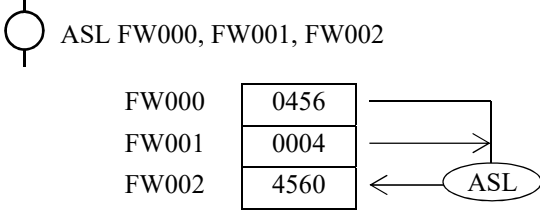
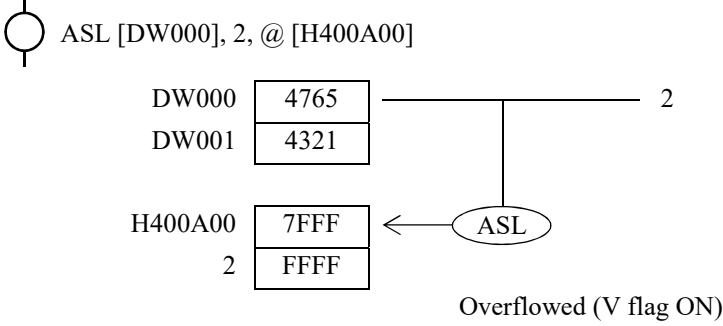
LSR	Logic right-shift																																									
Function description	This function right-shifts the contents of the source with the contents of the destination and stores the finding in the result.																																									
Parameter and operation	<p>⊙ LSR S, D, R</p> <p>S: Source R: Result D: Destination</p>	<p>RS will depend on whether it is word or long. (15/31)</p>																																								
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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																									
Typical use	<p>⊙ LSR FW000, FW001, FW002</p> <table border="1" style="margin-left: 20px;"> <tr><td>FW000</td><td>0456</td></tr> <tr><td>FW001</td><td>0004</td></tr> <tr><td>FW002</td><td>0045</td></tr> </table> <p style="margin-left: 20px;">↳ LSR ←</p> <p>⊙ LSR [DW000], 2, @ [H400A00]</p> <table border="1" style="margin-left: 20px;"> <tr><td>DW000</td><td>8765</td></tr> <tr><td>DW001</td><td>4321</td></tr> </table> <p style="margin-left: 20px;">————— 2</p> <table border="1" style="margin-left: 20px;"> <tr><td>H400A00</td><td>21D9</td></tr> <tr><td>2</td><td>50C8</td></tr> </table> <p style="margin-left: 20px;">↳ LSR ←</p>			FW000	0456	FW001	0004	FW002	0045	DW000	8765	DW001	4321	H400A00	21D9	2	50C8																									
FW000	0456																																									
FW001	0004																																									
FW002	0045																																									
DW000	8765																																									
DW001	4321																																									
H400A00	21D9																																									
2	50C8																																									
Effective parameter	<p>△ is an address. Parameter error if the number is odd.</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	—	√	√	Indirect word length	—	△	△	Indirect long length	—	△	△	<table border="1" style="margin-left: 20px;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	—	Direct long length	—	√	—	Indirect word length	—	△	△	Indirect long length	—	△	△
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	—	△	△																																							


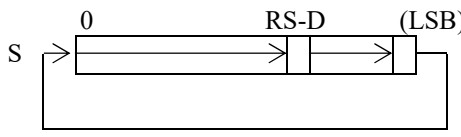
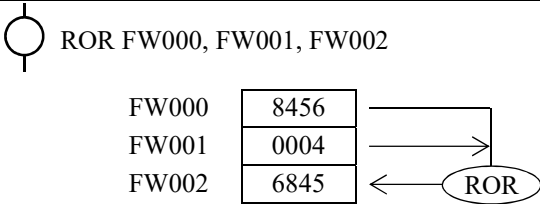
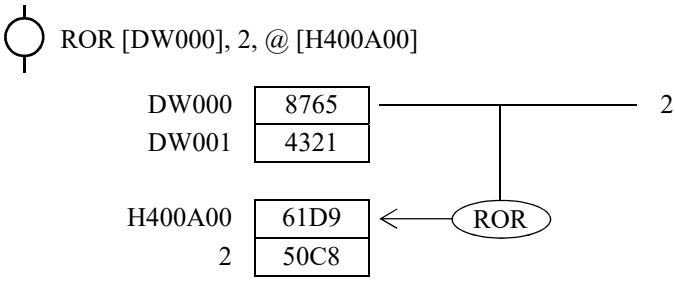
5. APPLIED INSTRUCTIONS

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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																										
Typical use	 																																										
Effective parameter	<table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>		R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																								


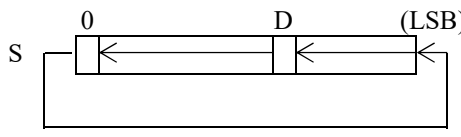

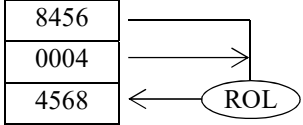

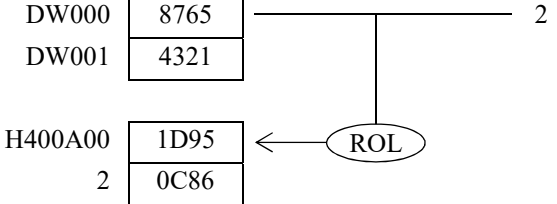
ASR	Arithmetic right-shift																																											
Function description	This function right-shifts (holds the code bit) the contents of the source with the contents of the destination and stores the finding in the result.																																											
Parameter and operation	 ASR S, D, R S: Source R: Result D: Destination	 <p>RS will depend on whether it is word or long. (15/31)</p>																																										
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 it is long length. Do not specify bit-type PI/O for R because the result will not be set correctly.																																											
Typical use	 ASR FW000, FW001, FW002 FW000: 8456 FW001: 0004 FW002: F845 ASR result: F845																																											
	 ASR [DW000], 2, @ [H400A00] DW000: 8765 DW001: 4321 H400A00: E1D9 2: 0246 ASR result: E1D9																																											
Effective parameter	<table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>		S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>		R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																									
	△ is an address. Parameter error if the number is odd.																																											


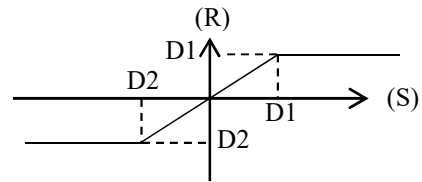
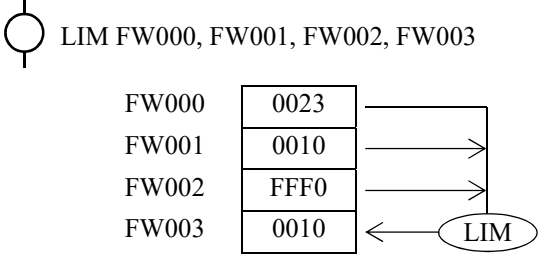
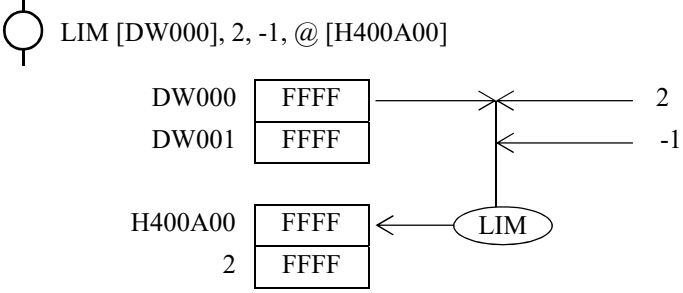
5. APPLIED INSTRUCTIONS

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		(LSB) LSB will depend on whether it is word or long. (15/31)																																								
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 it is long length. Do not specify bit-type PI/O for R because the result will not be set correctly.																																										
Typical use	  Overflowed (V flag ON)																																										
Effective parameter	<table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△	<p>△ is an address. Parameter error if the number is odd.</p>
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	–	△	△																																								


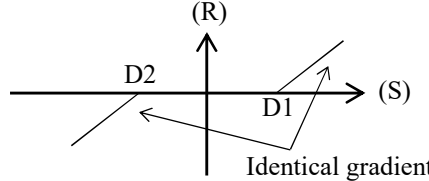
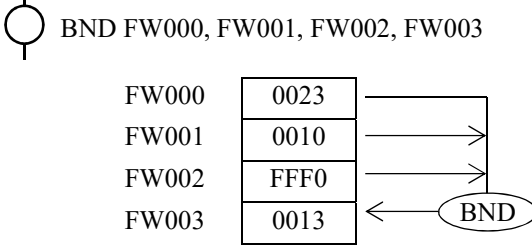
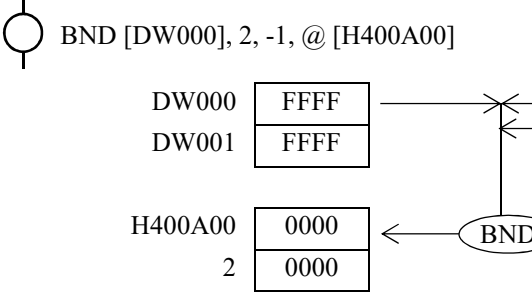
ROR	CW rotation																																										
Function description	This function rotates the contents of the source clockwise with the contents of the destination and stores the finding in the result.																																										
Parameter and operation	 ROR S, D, R S: Source R: Result D: Destination	 <p>RS will depend on whether it is word or long. (15/31)</p>																																									
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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																										
Typical use	 																																										
Effective parameter	<p>△ is an address. Parameter error if the number is odd.</p> <table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	-	√	√	Indirect word length	-	△	△	Indirect long length	-	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>		R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	-	Direct long length	-	√	-	Indirect word length	-	△	△	Indirect long length	-	△	△
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	-	△	△																																								


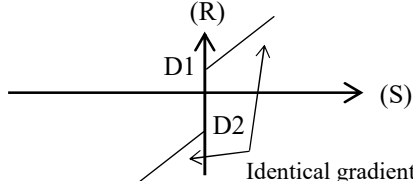

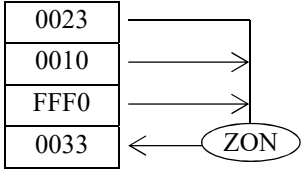

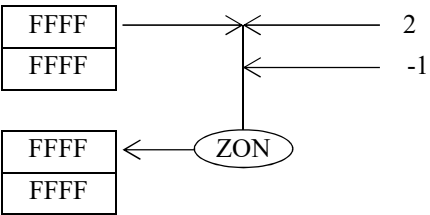
5. APPLIED INSTRUCTIONS

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 it is long length. Do not specify bit-type PI/O for R because the result will not be set correctly.																																									
Typical use	 ROL FW000, FW001, FW002 <table border="1" style="margin-left: 20px;"> <tr><td>FW000</td><td>8456</td></tr> <tr><td>FW001</td><td>0004</td></tr> <tr><td>FW002</td><td>4568</td></tr> </table>   ROL [DW000], 2, @[H400A00] <table border="1" style="margin-left: 20px;"> <tr><td>DW000</td><td>8765</td></tr> <tr><td>DW001</td><td>4321</td></tr> </table> <table border="1" style="margin-left: 20px;"> <tr><td>H400A00</td><td>1D95</td></tr> <tr><td>2</td><td>0C86</td></tr> </table> 			FW000	8456	FW001	0004	FW002	4568	DW000	8765	DW001	4321	H400A00	1D95	2	0C86																									
FW000	8456																																									
FW001	0004																																									
FW002	4568																																									
DW000	8765																																									
DW001	4321																																									
H400A00	1D95																																									
2	0C86																																									
Effective parameter	<table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																							


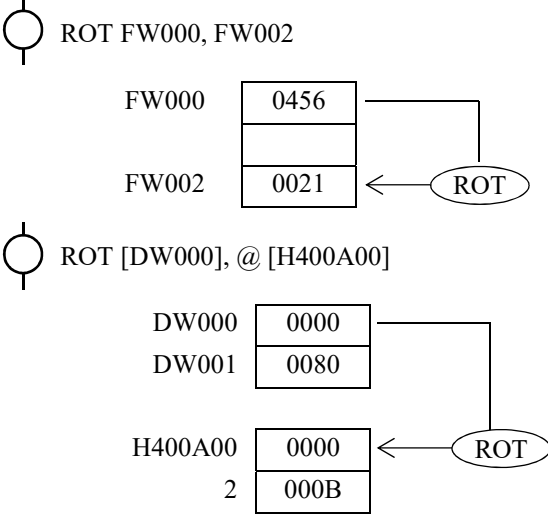
LIM	Limiter																																									
Function description	This function compares the contents of the source with the contents of the boundary values (destinations D1 and D2) and stores the finding in the result.																																									
Parameter and operation	 LIM S, 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	 																																									
Effective parameter	<table border="1"> <thead> <tr> <th>S, D1, D2</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D1, D2	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																							


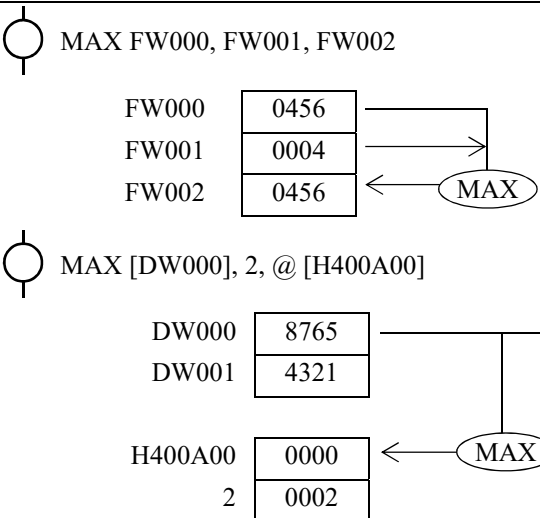
5. APPLIED INSTRUCTIONS

BND	Dead band																																										
Function description	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).																																										
Parameter and operation	 BND 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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																										
Typical use	 																																										
Effective parameter	<table border="1"> <thead> <tr> <th>S, D1, D2</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D1, D2	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>		R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																								


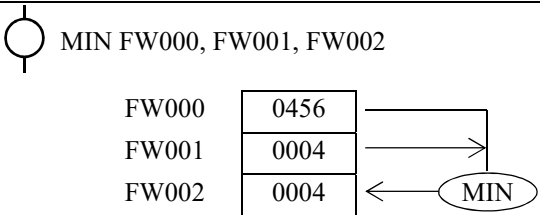
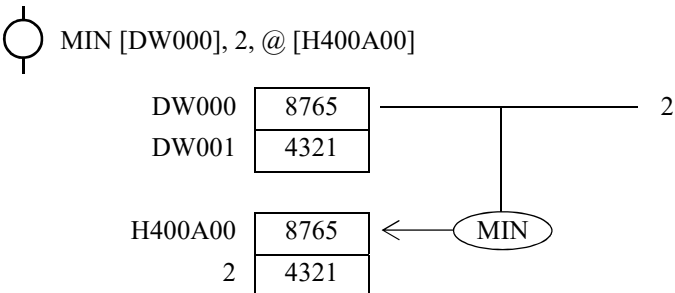
ZON	Dead zone																																									
Function description	This function adds a bias (destinations D1 and D2) to the contents of the source depending on whether it is positive or negative and stores the finding in the result.																																									
Parameter and operation	 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. Do not specify bit-type PI/O for R because the result will not be set correctly.																																									
Typical use	 ZON FW000, FW001, FW002, FW003 <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>FW000</td><td>0023</td></tr> <tr><td>FW001</td><td>0010</td></tr> <tr><td>FW002</td><td>FFF0</td></tr> <tr><td>FW003</td><td>0033</td></tr> </table>   ZON [DW000], 2, -1, @ [H400A00] <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>DW000</td><td>FFFF</td></tr> <tr><td>DW001</td><td>FFFF</td></tr> </table>  <table border="1" style="display: inline-table;"> <tr><td>H400A00</td><td>FFFF</td></tr> <tr><td>2</td><td>FFFF</td></tr> </table>			FW000	0023	FW001	0010	FW002	FFF0	FW003	0033	DW000	FFFF	DW001	FFFF	H400A00	FFFF	2	FFFF																							
FW000	0023																																									
FW001	0010																																									
FW002	FFF0																																									
FW003	0033																																									
DW000	FFFF																																									
DW001	FFFF																																									
H400A00	FFFF																																									
2	FFFF																																									
Effective parameter	<table border="1"> <thead> <tr> <th>S, D1, D2</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D1, D2	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△
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	–	△	△																																							




5. APPLIED INSTRUCTIONS

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 $S \geq 0$, Square root of S \rightarrow R When $S < 0$ 0 \rightarrow R																																										
Flag configuration	The E and V will change. The others will become turned off.																																											
Remark	Do not specify bit-type PI/O for R because the result will not be set correctly.																																											
Typical use																																												
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>				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	–	△	△
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 $S \geq D$, $S \rightarrow R$ When $S < D$ $D \rightarrow R$																																										
Flag configuration	The E and V will change. The others will become turned off.																																											
Remark																																												
Typical use																																												
Effective parameter	<table border="1"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	S, D	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	√	Direct long length	–	√	√	Indirect word length	–	△	△	Indirect long length	–	△	△	<table border="1"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>	R	Bit-type PI/O	Word-type PI/O	Constant	Direct word length	√	√	–	Direct long length	–	√	–	Indirect word length	–	△	△	Indirect long length	–	△	△	<p>△ is an address. Parameter error if the number is odd.</p>	
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	–	△	△																																									

5. APPLIED INSTRUCTIONS

MIN	Minimum value																																											
Function description	This function compares the contents of the source with those of the destination in size and stores the smaller value in the result.																																											
Parameter and operation	 MIN S, D, R S: Source R: Result D: Destination	When $S \leq D$, $S \rightarrow R$ When $S > D$ $D \rightarrow R$																																										
Flag configuration	The E and V will change. The others will become turned off.																																											
Remark																																												
Typical use	 																																											
Effective parameter	<table border="1" style="display: inline-table; margin-right: 20px;"> <thead> <tr> <th>S, D</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" style="display: inline-table;"> <thead> <tr> <th>R</th> <th>Bit-type PI/O</th> <th>Word-type PI/O</th> <th>Constant</th> </tr> </thead> <tbody> <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> </tbody> </table>				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	–	△	△
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	–	△	△																																									

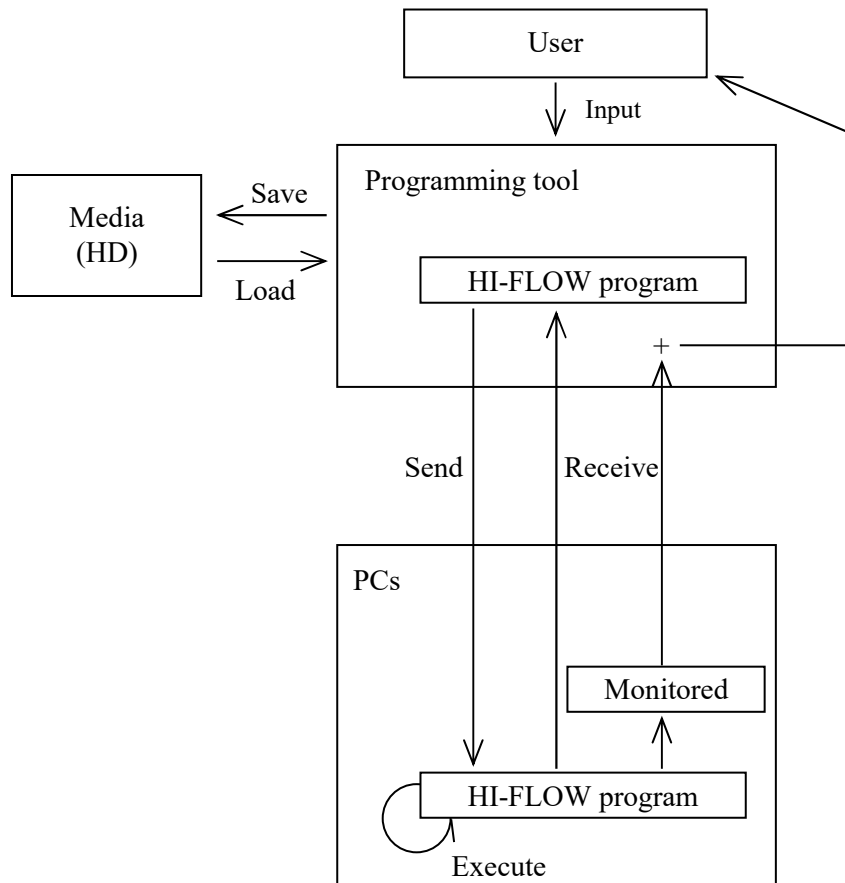
CLR	Clear																														
Function description	This function clears the specified I/O area. The TCLR, UCLR, and CCLR will clear the discrete value area as well.																														
Parameter and operation	 Name S Name: Name of each CLR instruction S: Source (Unused. Specify 0.)																														
Flag configuration	The system will turn off all flags.																														
Remark																															
Description	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Name</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>XCLR</td> <td>Clears X0000 through XFFFF.</td> </tr> <tr> <td>YCLR</td> <td>Clears Y0000 through YFFFF.</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 E0000 through EFFFF.</td> </tr> <tr> <td>FCLR</td> <td>Clears S0020 through S002F.</td> </tr> <tr> <td>JCLR</td> <td>Clears J000 through JFFF.</td> </tr> <tr> <td>QCLR</td> <td>Clears Q0000 through QFFFF.</td> </tr> <tr> <td>HHCLR</td> <td>Clears HH000 through HH1FF.</td> </tr> </tbody> </table>	Name	Function	XCLR	Clears X0000 through XFFFF.	YCLR	Clears Y0000 through YFFFF.	GCLR	Clears G000 through GFFF.	RCLR	Clears R000 through RFFF.	KCLR	Clears K000 through KFFF.	TCLR	Clears T000 through T3FF. Clears the measurement of T.	UCLR	Clears U000 through U3FF. Clears the measurement of U.	CCLR	Clears C000 through C3FF. Clears the measurement of C.	VCLR	Clears V000 through VFFF.	ECLR	Clears E0000 through EFFFF.	FCLR	Clears S0020 through S002F.	JCLR	Clears J000 through JFFF.	QCLR	Clears Q0000 through QFFFF.	HHCLR	Clears HH000 through HH1FF.
Name	Function																														
XCLR	Clears X0000 through XFFFF.																														
YCLR	Clears Y0000 through YFFFF.																														
GCLR	Clears G000 through GFFF.																														
RCLR	Clears R000 through RFFF.																														
KCLR	Clears K000 through KFFF.																														
TCLR	Clears T000 through T3FF. Clears the measurement of T.																														
UCLR	Clears U000 through U3FF. Clears the measurement of U.																														
CCLR	Clears C000 through C3FF. Clears the measurement of C.																														
VCLR	Clears V000 through VFFF.																														
ECLR	Clears E0000 through EFFFF.																														
FCLR	Clears S0020 through S002F.																														
JCLR	Clears J000 through JFFF.																														
QCLR	Clears Q0000 through QFFFF.																														
HHCLR	Clears HH000 through HH1FF.																														
Typical use	 XCLR 0  HHCLR 0																														

This Page Intentionally Left Blank

SUPPLEMENT A 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.

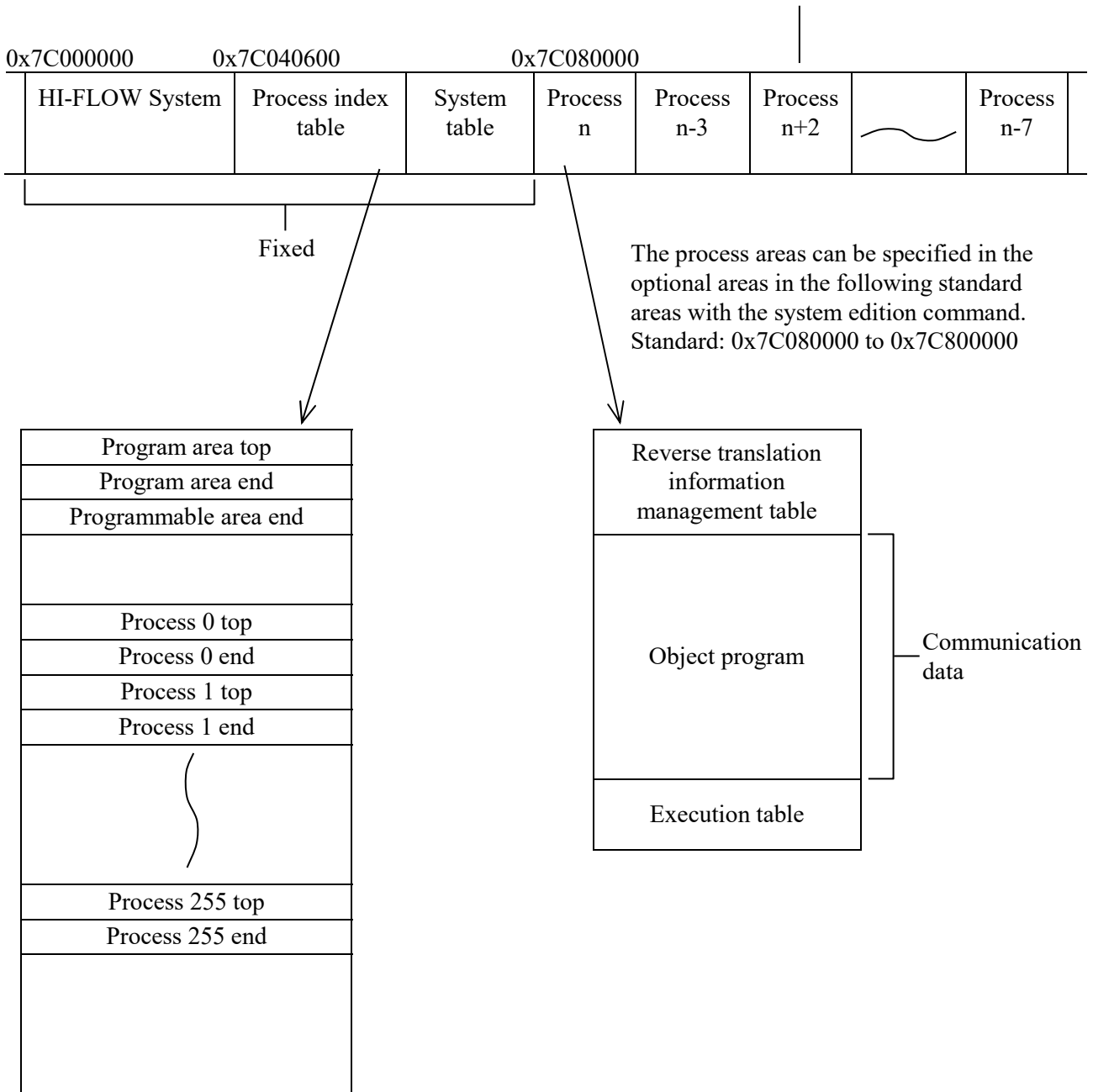
The system also gives and receives data with different media (HD) to save and load created programs.



SUPPLEMENT B PCs MEMORY

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

The processes are not necessarily in ascending order.



SUPPLEMENT C ONLINE MODE

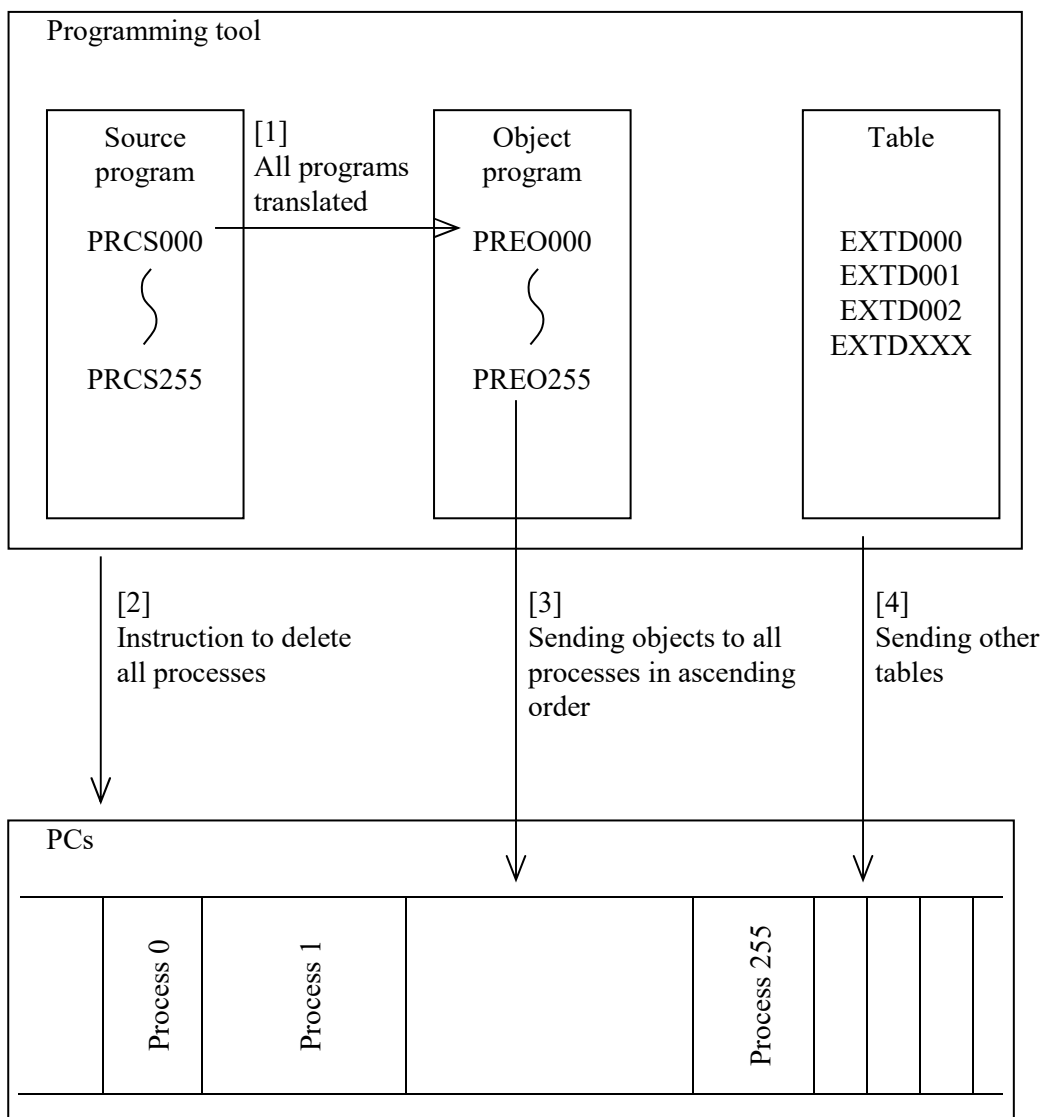
“Offline” means the mode in which an object to be edited is turned into a program as part of the programming tool regardless of the contents of the PCs memory.

“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

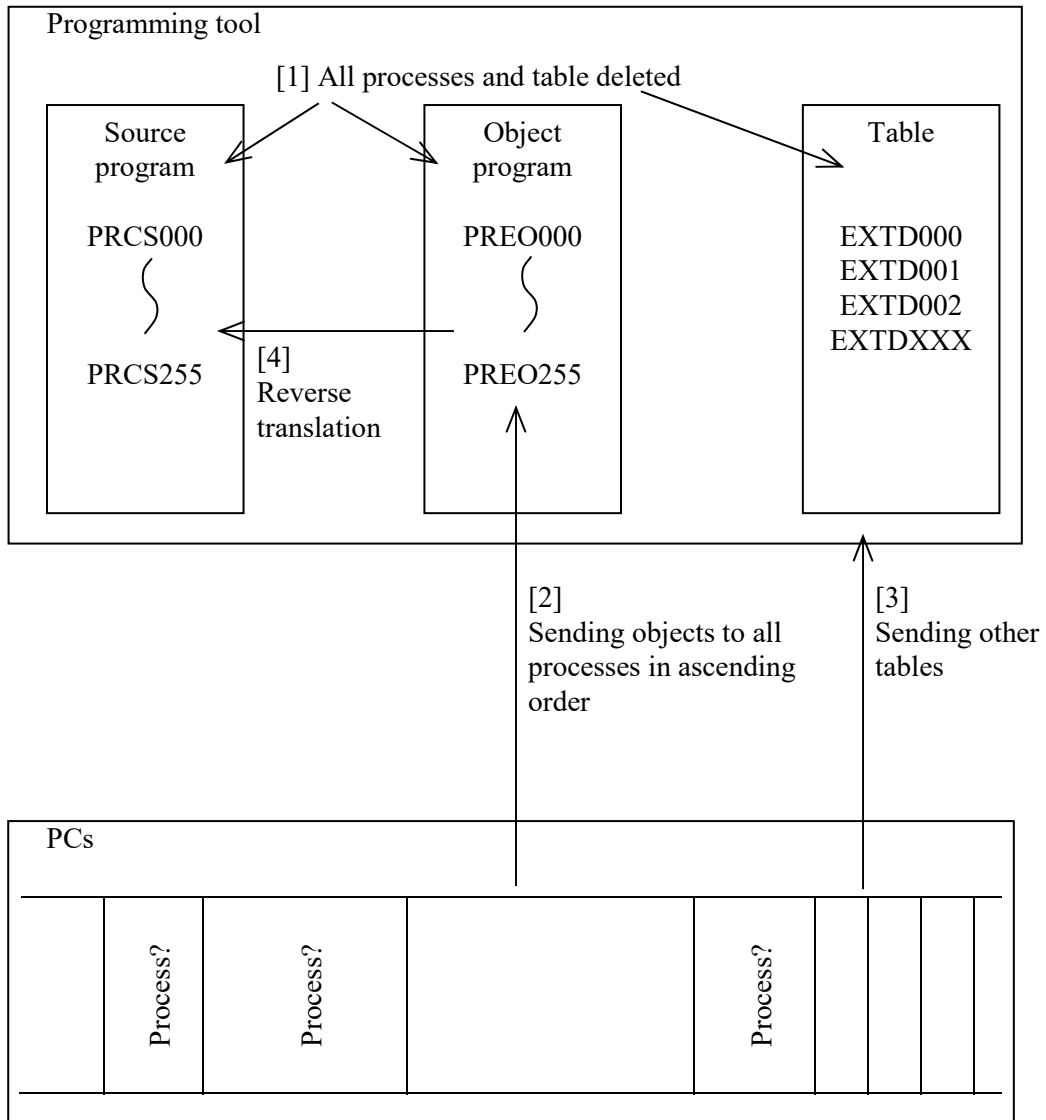
Here is the flow of data when all HI-FLOW programs existing on the tool are sent to the PCs.



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

(2) Receiving all processes

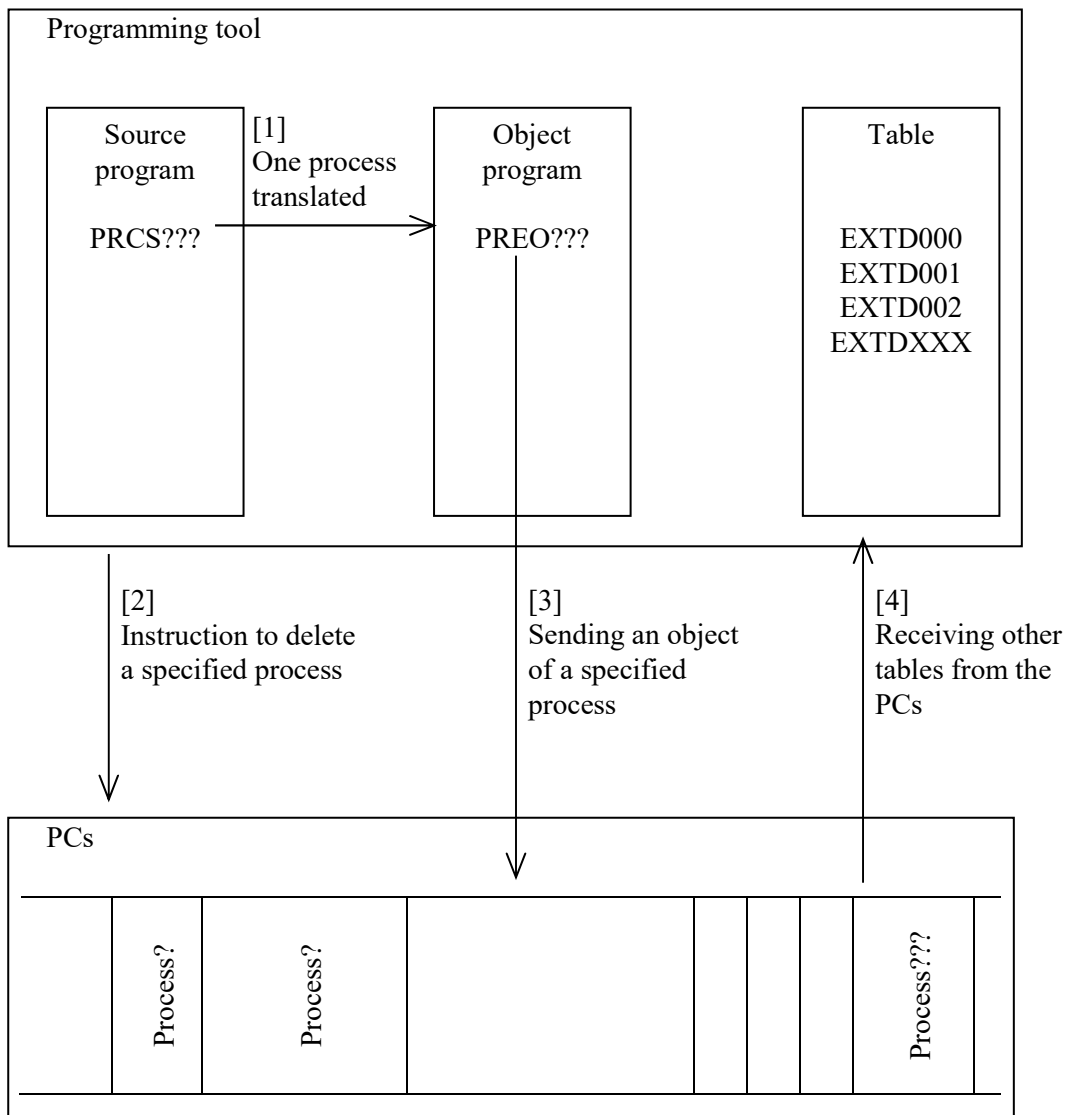
Here is the flow of data when all HI-FLOW programs existing on the PCs are sent to the tool.



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

(3) Sending one process

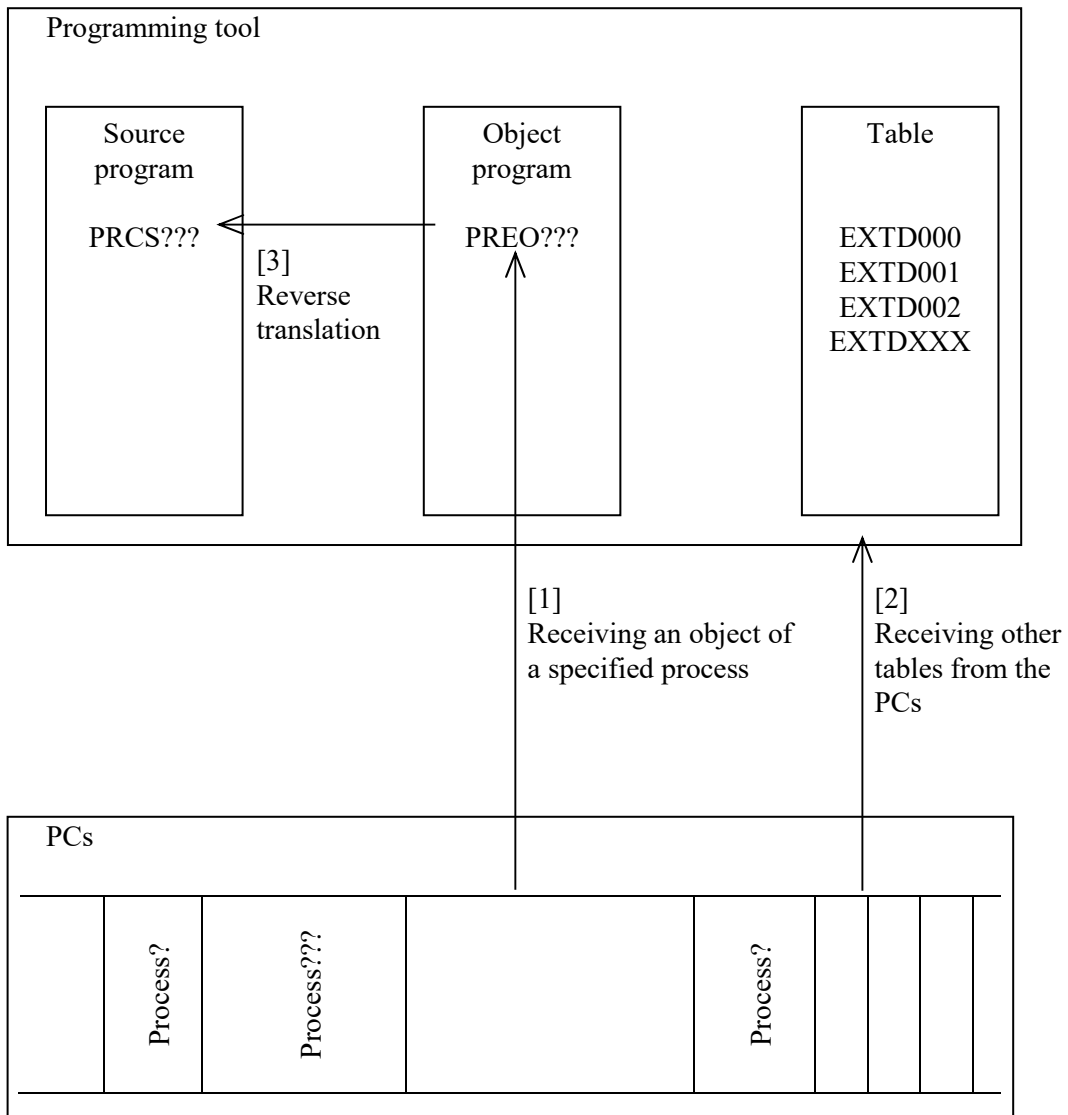
Here is the flow of data when a single particular HI-FLOW program existing on the tool is sent to the PCs.



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 waded in ascending order when received.

SUPPLEMENT D 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.

SUPPLEMENT D PROGRESS CHECK

Item	Description
Selection	<p>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.</p> <p>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.</p>
Wait for the conditions	<p>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.</p>
Figure without delay	<p>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.</p>
Non-synchronous merging	<p>If a para start is made, then a check must be made to see if the next step is still in progress.</p> <p>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.</p>

SUPPLEMENT E HI-FLOW PROGRAM AND CPU LOAD

A HI-FLOW program runs as part of the HP on the CPU module of the PCs. Increasing the number of HI-FLOW programs therefore increases the HP load on the CPU 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.

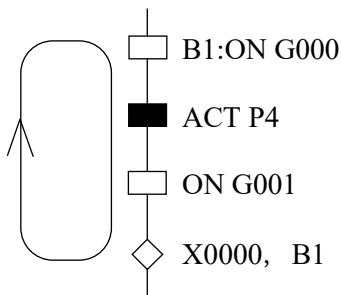
<How to create a HI-FLOW program effectively>

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

Any program that executes all of a large number of incorporated split processes or routes will have a heavier load. The vertical (route) length of the HI-FLOW program is not significant.

- [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 X0000 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.

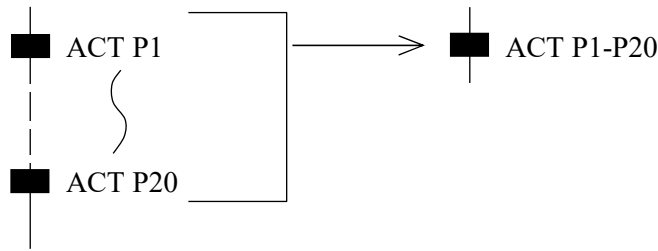
Wait timers on the same route are never executed simultaneously. Assign an identical timer number and avoid using the rear timer number whenever possible.

[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.

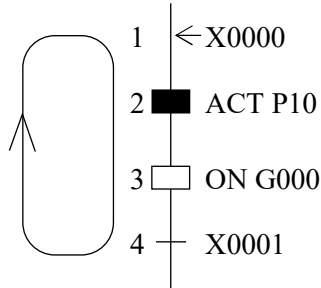
The system control bit needs a check for each sequence cycle and for each step execution. It is then considerably loaded. Set a necessary minimum.

[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 X0001 does not hold and X0000 remains on in the left-hand program, the system will continue to execute 1 through 4 for each sequence. To prevent this, change X0000 to an edge condition (to execute only when X0000 changes from OFF to ON).

[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.**

Complex conditional expressions of HI-FLOW take considerable time to analyze compared with the ladder. Complicated conditions will become lighter in load when handed over to HI-FLOW after being received by the ladder.

This Page Intentionally Left Blank