

2a S10mini

SERIES

SOFTWARE MANUAL PROGRAMMING HI-FLOW For Windows®

Applicable to : HITACHI-S10/2 α HITACHI-S10/2 α E HITACHI-S10/2 α H HITACHI-S10/2 α Hf S10mini model S S10mini model H S10mini model F S10mini model D

NESP-S25E NESP-2 α E NESP-2 α H NESP-2 α Hf





<u>NOTE</u>

All information in this manual is based on the latest product information available at the time of printing. Hitachi has reviewed the accuracy of this manual, but assumes no responsibility for any omissions or errors which may appear. The design of the product is under constant review and, while every effort is made to keep this manual up to date, the right is reserved to change specifications and equipment at any time without prior notice.

PROHIBITION

These products should not be used for medical, power supply, nuclear, water supply, drainage plants, traffic control, military, space, nor disaster prevention equipment.

Diversion and/or resale of these products without this manual is prohibited.

Reproduction of the contents of this manual in whole or in part, without written permission of Hitachi, is prohibited.

TRADEMARKS

HITACHI-S10/2 α , S10/4 α and PSE α are registered trademarks of Hitachi, Ltd.

FIRST EDITION, JUNE, 1997, SAE - 3 - 122 (A) (out of print) SECOND EDITION, OCTOBER, 2000, SAE - 3 - 122 (C) THIRD EDITION, SEPTEMBER, 2003, SAE - 3 - 122 (D) All Rights Reserved, Copyright © 1997, 2003, Hitachi, Ltd.

BI-KB-TN<IC-NS> (FL-MW20)

LIMITED WARRANTY

Hitachi, Ltd., warrants its products to be manufactured in accordance with published specifications and free from defects in materials and/or workmanship.

Hitachi, Ltd., warrants its products against defects in parts and workmanship for one full year from date of purchase.

HITACHI, LTD., MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED EXCEPT AS PROVIDED HEREIN, INCLUDING WITHOUT LIMITATION THEREOF, WARRANTIES AS TO MARKETABILITY FOR A PARTICULAR PURPOSE OF USE, OR AGAINST INFRINGEMENT OF ANY PATENT. IN NO EVENT SHALL HITACHI BE LIABLE FOR ANY DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY NATURE, OR COSTS, CHARGES, LOSSES OR EXPENSES RESULTING FROM ANY DEFECTIVE PRODUCT OR THE USE OF ANY PRODUCT.

SOFTWARE UP-TO DATE POLICY

Hitachi, Ltd., constantly reviews its software so as to incorporate the latest technology. Hitachi reserves the right to make changes to any software to improve reliability, function, or design. Hitachi cannot be held responsible for any errors in its software.

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 the 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.

"RUN/STOP" SWITCH CAUTION

The "RUN/STOP" switch only stops execution of the ladder logic program or HI-FLOW program. Digital and analog outputs are left in the active state when execution stops, unless the optional rungs described in the CPU manual have been added. The "RUN/STOP" switch does not affect the operation of C-language or FA-BASIC language programs. Outputs can still be produced in response to C-language or FA-BASIC programs, or by the action of programmers typing in commands in these languages, while the "RUN/STOP" switch is in the "STOP" position.

DO NOT DEPEND ON THE STOP SWITCH TO STOP MOVING PARTS OR TO PREVENT UNEXPECTED MOTION OR ENERGIZATION. USE HARDWIRED SAFETY DISCONNECT AND LOCK OUT POWER AND CONTROL VOLTAGES BEFORE WORKING ON ELECTRICAL CIRCUITS OR PARTS THAT CAN MOVE.

PREFACE

Flowchart type programming language HI-FLOW was developed to allow the user to easily code programs for the programmable controller.

This manual describes instructions for programming in HI-FLOW. For ladder programs, refer to the following manual.

<Related manual>

SOFTWARE MANUAL PROGRAMMING LADDER CHART For Windows® (Manual number SAE-3-121)

See the following list when you use the NESP (Nissan Electronic Sequence Processor) series.					
[HITACHI-S10a series]		[NESP series]			
HITACHI-S10/2α		NESP-S25E			
HITACHI-S10/2aE		NESP- $2\alpha E$			
HITACHI-S10/2αH		$NESP-2\alpha H$			
HITACHI-S10/2αHf		$NESP-2\alpha Hf$			

<Trademarks>

 Microsoft® Windows® operating system, Microsoft® Windows® 95 operating system, Microsoft® Windows® 98 operating system, Microsoft® Windows® 2000 operating system, Microsoft® Windows® XP operating system are registered trademarks of Microsoft Corporation in the United States and/or other countries.

• Ethernet is a registered trademark of Xerox Corp.

Other product names written in this manual are the trademarks of each manufacturer.

CONTENTS

1 (CONFIGURATION OF HI-FLOW PROGRAMS	1
2 H	HOW TO USE THIS MANUAL	3
2.1	Overview	4
2.2	Outline of the Syntax	5
2.3	Application Instructions	6
3 F	PROCESSES	9
3.1	About the Process	10
3.2	Program	15
3.3	Process Information	28
4 E	EXPLANATION OF SYNTAXES	29
4.1	Process Start and Process End	30
4.2	Route Start and Route End	34
4.3	Wait	35
4.4	Box	37
4.5	Control Box	42
4.6	Repeat Start and Repeat End	46
4.7	If	47
4.8	Jump	49
4.9	Escape	50
4.10	Parallel Start and Parallel End	51
4.11	Select, Wait in Selective Branching and Select End	52
4.12	2 Multi-entry	53
4.13	3 Call	54
4.14	4 Function	55
4.15	5 Wait with Previous State Cleared	55
5 A	APPLICATION INSTRUCTIONS	57
5.1	Overview	58
5.2	Usage	58
5.3	Parameters	58
5.4	Type Conversion for Operation	60
5.5	System Error Flags	61

5.6 Explana	tion of Functions	62
SUPPLEMEN'	Γ	113
Supplement 1	Work Flow Based on HI-FLOW Program	114
Supplement 2	PCs Memory	115
Supplement 3	Online Mode	116
Supplement 4	Check for Progress	120
Supplement 5	Relationships between a HI-FLOW Program	
	and the CPU Load	122
INDEX		127

THIS PAGE INTENTIONALLY LEFT BLANK.

1 CONFIGURATION OF HI-FLOW PROGRAMS

1 CONFIGURATION OF HI-FLOW PROGRAMS

This manual describes the specifications of the new HI-FLOW language. When creating actual programs, refer to this manual at the necessary times.

HI-FLOW programs you create consist of the following components:

locess				From	n process 0	to process 2	255
– Progra	am ———				•		
	ute ———						
	Step ———					limit on the	2
[Step numbers	1 to	999	1 to 255	1 to 20 horizontally	, number of	
[Symbols		19	Vertically (Y axis).	(X axis).	routes.	
	Labels	B1 to	B255] ^{Up} to 999	limit on the		
$ $ $ $ $_{r}$	— Syntax ———			1	number of		
	Reserved wore	ds 18	21				
	Constants	3					
	Variables	5					
	Operators	15					
╽╽┝	Step comment	Lin to 70 chara	acters]			
╎┝╧]			
Fre	ee label	U	o to 6 c	characters			
Fre	ee comment	Up	to 70 c	characters			
- Proces	ss information —]	
Nam	e		Up t	o 16 chara	cters Se	ttable	
Com	ment		L In to	132 chara	cters for	each	

2 HOW TO USE THIS MANUAL

2.1 Overview

This manual is prepared according to the configuration shown in Chapter 1. The following table shows relationships between individual items and their corresponding chapters or sections and pages.

Item	Chapter or section	Page
• Process	3	9
Program	3.2	15
• Route		15
• Step		19
• Step number		20
• Symbol		20
• Label		23
• Syntax		23
 Reserved word 		24
• Constant		24
Variable		24
 Operator 		26
• Step comment		26
• Free label		27
• Free comment		27
 Process information 	3.3	28
• Name		28
• Comment		28

2.2 Outline of the Syntax

After "2.1 Overview," this manual describes details of the syntax for each function. The following table shows relationships between functions and their corresponding chapters or sections and pages.

Item	Symbol	Chapter or section	Page
Explanation of syntax		4	29
Process Start		4.1	30
Process End			
• STP	Ŧ		30
• RST			31
• CLR			32
• ACT	_		32
Route Start		4.2	34
Route End	⊥_		
• Wait		4.3	35
 Condition expression 			35
• Timer	+		35
Output bit			35
• Wait timer			35
• Box		4.4	37
 Assignment expression 			37
• ON			38
• OFF	\Box		39
Parallel timer			39
• TUP			40
• TRS			41
Control Box		4.5	42
• ACT			42
• RST	I		43
• STP			44
• CLR			44
Repeat Start	\rightarrow	4.6	46
Repeat End	\forall		
• If	\diamond	4.7	47
• Jump	Ļ	4.8	49
• Escape	\times	4.9	50
Parallel Start	Ħ	4.10	51
Parallel End	\mathbb{V}		
• Select	 	4.11	52
Wait in Selective Branching	≡≢≡		
Select End	K	4.12	53
• Multi-entry			
• Call		4.13	54
• Function		4.14	55
Wait with Previous State Cleared		4.15	55

2.3 Application Instructions

HI-FLOW supports application instructions of functions as well as ladder diagrams. The following table shows application instructions and their functions.

Major class	Class	Symbol	Function	Page
Arithmetic	Addition	ADD	$S+D \rightarrow R$	63
operation	Subtraction	SUB	$S - D \rightarrow R$	64
instructions	+1	INC	$S+1 \rightarrow S$	65
	-1	DEC	$S - 1 \rightarrow S$	66
	Multiplication	MUL	$S*D \rightarrow R$	67
	Division	DIV	$S/D \rightarrow R$	68
	Residue	MOD	Residual of S/D \rightarrow R	69
	Scale conversion	SCL	$S*D1/D2 \rightarrow R$	70
Logical	Logical Product	AND	$S AND D \rightarrow R$	71
operation	Logical Sum	OR	$S \text{ OR } D \rightarrow R$	72
instructions	Exclusive Logical Sum	EOR	$S EOR D \rightarrow R$	73
	Not	NOT	NOT S \rightarrow R	74
Compare	=	EQU	True or false of $S = D \rightarrow R$	75
operation	<>	NEQ	True or false of $S <> D \rightarrow R$	76
instructions	>	GT	True or false of $S > D \rightarrow R$	77
	>=	GE	True or false of S \ge D \rightarrow R	78
	<	LT	True or false of $S < D \rightarrow R$	79
	<=	LE	True or false of S $\leq D \rightarrow R$	80
	Test	TST	Sign of $S \rightarrow R$	81
Data move	Move	MOV	$S \rightarrow D$	82
instructions	Block Move	MOM	$S \sim Sn \rightarrow D \sim Dn$	83
	Exchange	EXC	$S \leftrightarrow D$	84
	FIFO Write	PSH	$S \rightarrow D$ (FIFO table)	85
	FIFO Read	POP	S (FIFO table) \rightarrow D	86
	Address Set	AST	S address \rightarrow D	87
	Search	SCH	$S = D(n) \rightarrow n \text{ is set in } R.$	88
Data	BIN-BCD	BTD	$BIN \rightarrow BCD$	89
conversion			$S \longrightarrow R$	
instructions	BCD-BIN	DTB	$BCD \rightarrow BIN$	90
	DDI 70EC		$S \longrightarrow K$	01
	BIN-/SEG	SEG	$\begin{array}{c} \text{BIN} \rightarrow \text{/ segments} \\ \text{S} \longrightarrow \text{P} \end{array}$	91
			л — — к	

Major class	Class	Symbol	Function	Page
Data	BIN-ASC	ASP	BIN \rightarrow ASCII (packed, unpacked)	92
conversion		ASU	$S \rightarrow (R, R+1), (R, R+1, R+2, R+3)$	93
instructions	ASC-BIN	APB	ASCII (packed, unpacked) \rightarrow BIN	94
		AUB	$(S, S+1), (S, S+1, S+2, S+3) \rightarrow R$	95
	Absolute Value	ABS	$ S \rightarrow R$	96
	+/-	NEG	$_S \rightarrow R$	97
	Decode	DCD	2^{11} to 2^{15} of S \rightarrow Bit 2^{n} of R	98
			is turned on.	
	Encode	ECD	Number of the first turned-on bit of S \rightarrow 2^11 to 2^15 of R	99
Shift	Logical Right Shift	LSR	S Logical Right Shift D \rightarrow R	100
instructions	Logical Left Shift	LSL	S Logical Left Shift D \rightarrow R	101
	Arithmetic Right Shift	ASR	S Arithmetic Right Shift D \rightarrow R	102
	Arithmetic Left Shift	ASL	S Arithmetic Left Shift D \rightarrow R	103
Rotate	Right Rotate	ROR	S Right Rotate R	104
instructions	Left Rotate	ROL	S Lift Rotate R	105
Function	Limiter	LIM		106
process	Dead Band	BND		107
instructions	Dead Zone	ZON		108
	Root	ROT		109
	MAX	MAX		110
	MIN	MIN		111
Special	Clear	XCLR		112
instructions		YCLR		
		GCLR		
		RCLR		
		KCLR		
		CCLR		
		VCLR		
		FCLR		
		FCLR		
		JCLR		
		QCLR		
		HHCLR		

THIS PAGE INTENTIONALLY LEFT BLANK.

3.1 About the Process

A process is the largest configuration unit in a HI-FLOW program. It starts with Process Start (ϕ) and ends with Process End (ϕ). A process consists of programs having at least one route and process information. You can control production lines with one or more processes created by function.

A process (P0 to P255) is recognized with "P+process number (in decimal)." P0 is called the initial process. It is reserved for activation by the HI-FLOW operating system that controls execution of the HI-FLOW program when the PCs is turned on. After the initial process is activated, processes P1 to P255 can be controlled.

During process execution, the specified PI/O register is turned on, enabling process execution to be monitored. (See the description of standard QF00 to QFFF and the system bit assignment command.)



Process States

Each process in the PCs is in one of nine states.

State	Description			
Not found	There is no HI-FLOW process.			
Executable	A HI-FLOW process can be operated when it is activated.			
Executing	A HI-FLOW process was with ACT by another process and is being executed.			
Stop	A HI-FLOW process is being stopped in the middle because some conditions were satisfied. The process information and PI/O values are held. For the timer, you can specify whether you hold the timer value or continue measurement.			
Reset	Execution of a HI-FLOW process was canceled because some conditions were satisfied, and it is stopping at the process start point. The process information is initialized. The PI/O values are held. For the timer, you can specify whether you reset the timer at predetermined reset time or continue measurement.			
Clear	When some conditions are satisfied while a HI-FLOW process is being stopped, reset, call-stopped, or callreset, the bit-type PI/O (in the ON statement or parallel timer) being used in the process is cleared to 0.			
Call execution	A HI-FLOW process was subroutine-called by another process and is being executed.			
Call stop	A HI-FLOW process is being stopped at a point in it because some conditions were satisfied during execution of a call. The process information and PI/O values are held. For the timer, you can specify whether you hold the timer value or continue measurement.			
Call reset	Execution of a HI-FLOW process was canceled because some conditions were satisfied during execution of a call, and it is stopping at the process start point. The process information is initialized. The PI/O values are held. For the timer, you can specify whether you reset the timer at reset time or continue measurement.			

The process enters into the stop and reset state once conditions are satisfied. Even if these conditions are released, the process continues the states. The clear state is entered each time the conditions are satisfied.

Process states change

There are nine process states. The following figure shows how these states are changed by what (circled numbers in the figure).

[State change]



- ① Control box ACT (
- (2) Escape (\times)
- (3) Process Start STP (ϕ), Control Box STP (μ)
- (4) Process Start ACT ($\mathbf{\Phi}$), Control Box ACT ($\mathbf{\mu}$)
- (5) Process Start RST ($\mathbf{\Phi}$), Control Box RST ($\mathbf{\mu}$)
- ⑥ Process Start ACT (●), Control Box ACT (■)
- 7 Process Start RST (ϕ), Control Box RST (ϕ)
- (8) Process Start CLR ($\mathbf{\Phi}$), Control Box CLR ($\mathbf{\mu}$)
- (9) Process Start CLR ($\mathbf{\Phi}$), Control Box CLR ($\mathbf{\mu}$)
- 1 Process Call (
- ① Process Call (
- IProcess End (●), Escape (×)
 Control Box RST to calling process (■)
 Process Start RST to calling process (●)
- IB Process Start STP (●)
 Control Box STP to calling process (■)
 Process Start STP to calling process (●)
- Image: Process Start ACT (●)
 Control Box ACT to calling process (■)
 Process Start ACT to calling process (●)
- ① Process Start RST (●)
- (16) Process Start ACT (igoplus)
- (17) Process Start RST (igoplus)
- Image: Process Start CLR (●)
 Control Box CLR to calling process (■)
 Process Start CLR to calling process (●)
- 19 Process Start CLR (igoplus)
- Control Box RST to calling process (
 Process Start RST to calling process (
)
- (21) Control Box RST to calling process (➡)
 Process Start RST to calling process (●)

When a process is changed to the executing or call execution state, you can specify either of two activation methods: master reset or zone. When you omit specification, zone activation is assumed.

When a process is changed to the process end (\spadesuit), escape (\times), or executable state, the handing of the PI/O values (whether they are held or cleared to 0) and the timer (whether the timer is forcibly timed out or reset or measurement is continued) depends on the activation method.

Relationships between the setting of the key switch on the PCs and the states of processes

Below is an explanation of how processes in the PCs change according to the setting of the key switch on the PCs and when the PCs are recovered from a power failure. HI-FLOW does not recognize the difference in the setting of the key switch between RUN and SIM RUN. With the key switch set to RUN or SIM RUN, the states of processes depend on the operation of the PCs.

- (a) When the PCs are recovered from a power failure (when the key switch is reset)
 When the PCs are recovered from a power failure, all processes in the PCs are initialized.
 <At initialization>
 - The processes become executable.
 - The timers are stopped.

• PI/Os are turned off. (The states of DW, FW, K, and KW remain unchanged.) Process 0 (initial process) is reserved for activation. It is executed when the key switch on the PCs is set to RUN next time.

(b) When the key switch is set to STOP

With the key switch on the PCs set to STOP, the states of the processes remain unchanged even when the state of a PI/O or timer in the PCs changes.

- (c) When the key switch is set to RUN or SIM RUN With the key switch on the PCs set to RUN or SIM RUN, the states of the processes change accordingly when the state of a PI/O and/or timer in the PCs changes.
- (d) When the key switch is changed from STOP to RUN or SIM RUN When the key switch on the PCs is changed from STOP to RUN or SIM RUN, the processes change from the state in (b) to that in (c). If the PCs are in the state immediately after power is recovered, process 0 is executed. You can also set the processes in the state in (c) when the PCs are not in the state immediately after power is recovered. To do this, specify the same effect for HI-FLOW as after the PCs are recovered from a power failure. (See the system edition commands.)
- (e) When key switch is changed from RUN or SIM RUN to STOP When the key switch on the PCs is changed from RUN or SIM RUN to STOP, the processes changed from the state in (c) to that in (b). At the same time, timers WT and PT stop measurement.

3.2 Program

A process consists of programs and process information. Programs actually control production lines. A program consists of one or more routes.

Route

A route which flows vertically starts with Process Start (ϕ) or Route Start (\neg) and ends with Process End (ϕ) or Route end (\perp). A route is the minimum unit of a process program. With multiple routes, a process can be subject to synchronous or selective processing. A branch occurs at a main route. Branching routes are called subroutes. A subroute branches at Parallel Start (\models) or Select (\vdash). Subroutines are joined at Parallel End ($\not\models$) or Select End ($\not\models$).

You do not need to recognize routes with route numbers, so these numbers are managed only by the system.



Synchronous routes do not always need to be joined. When they are not joined, the effect of the main route is just to activate routes. A selection route must be joined with another route even if there is an unconditional branch.

(1) Use of both the synchronization syntax and selection syntax

When the synchronization syntax and selection syntax are programmed independently, there is no problem. When using them together, however, care must be taken.

(a) When the same route is used as a route at which a branch starts and a route at which routes are joined

For both synchronization and selection routes, all possible patterns are allowed.



(b) When different routes are used as a route at which a branch starts and a routed at which routes are joined

When the synchronization syntax and selection syntax are programmed independently, operation is possible. Otherwise, a program can be created but it cannot be operated correctly.

[The program runs correctly.]



[The program does not run correctly.]



Step

A step is an instruction unit in HI-FLOW. As with a free label and free comment, a step is a unit of a route. A step consists of a step number, symbol, label, syntax, and step comment.



* Note that, as regards the combined use of syntax, label, and comment strings, the total length of the strings that is acceptable is up to 70 characters. Note, also, that a logical operator within a syntax string is considered to be two characters long in counting although it is regarded as a string of only one character in editing.



Step number

A step number is a unique number in the process. It is automatically assigned by the system when the program is created. (A process number is from 1 to 999. This means that up to 999 steps can be created in one process.)



Symbol

A symbol outlines a condition, branch, or other control. A symbol is always required when a step is created. Some steps consist of only a symbol. Other steps consist of a symbol and syntax.



There are 19 symbols (listed below). Symbols themselves have meanings.

* Note that, as regards the combined use of syntax, label, and comment strings, the total length of the strings that is acceptable is up to 70 characters. Note, also, that a logical operator within a syntax string is considered to be two characters long in counting although it is regarded as a string of only one character in editing.

[Symbols usable in HI-FLOW]

No.	Symbol	Name	Function	Syntax	Remarks
1	•	Process Start	Starts a process.	Yes	
2	-	Process End	Stops the process.	No	
3	T	Route Start	Starts a subroute.	No	
4		Route End	Stops the subroute.	No	
5	+	Repeat Start	Starts a repetitive process.	Yes	End is decided with >=.
6	+	Repeat End	Stops the repetitive process.	No	
7	¢	If	Conditionally branches program control.	Yes	A branch to another route can be made.
8	L.	Jump	Unconditionally branches program control.	No	A branch to another route can be made.
9	×	Escape	Forrcibly stops the local process.	No	A subprocess returns to the main process in the same scan.
10		Parallel Start	Branches to a synchronous subroute.	No	
11	Ш	Parallel End	Waits for the end of a synchronous subroute.	No	When program control is returned from the selectively branched subroutine, control is passed to the next step in the same scan. (With the previous model, a delay of one scan occurred.)
12		Select	Branches to a selectively branched subroutine.	No	
13	≡≢≡	Wait in Selective Branching	Provides a route selection condition in selective branching.	Yes	Use Route Start and Select as a pair.
14	¥	Select End	Returns program control form the selectively branched subroutine.	No	The subroute may not join the main route. (This is not possible in the previous model.) Control is passed to the next step with no scan delay. (With the previous model, a delay of one scan was required before the next step was executed.)

(2/2)

No.	Symbol	Name	Function	Syntax	Remarks
14	¥	Multi-entry	Starts reexecution from this step when the set conditions are satisfied.	Yes	
15	+	Wait	Waits for the shift condition to be satisfied.	Yes	
			Wait for the specified time to expire.	Yes	It is possible to monitor that the condition for consecutive PI/Os are satisfied.
16	Ļ	Box	PI/O Output.	Yes	With interlocked Y- output.
			Assignment expression	Yes	
			Sends PI/O waveforms.	Yes	
			Resets timers.	Yes	Up to 7 timers can be reset. This function is equivalent to the conventional forcible timeout function.
			Causes a forcible timeout.	Yes	New function added to this model.
17		Control Box	Controls the state of another process.	Yes	The master reset function is provided. The function to specify a step is added. The function to specify STP timer measurement is added. The function to forcibly timeout or reset the RST timer is added.
			Controls a task.	Yes	
18		Call	Call a subroute of another process.	Yes	The master reset function is provided. The function to specify a step is added.
19	¢	Function	Application instruction	Yes	New function added to this model.
20	+*	Condition with previous State Cleared	Clears PI/O when a condition is changed.	Yes	New function added to this model. Use this function together with the wait function.
		Wait Timer with previous State Cleared	Clears PI/O when a forcible timeout occurs.	Yes	

Label

A label represents a destination to which a symbol branches. A label is represented by a number from B1 to B255 followed by a colon (:). (These numbers can be created for each process. A branch to another process cannot be specified.) Labels can be assigned only to steps.



Syntax

A syntax clarified the contents of a condition expression, assignment expression, or control statement. Some symbols have no syntax. A syntax consists of a reserved word and an expression which consists of a constant, variable and operator.



* Note that, as regards the combined use of syntax, label, and comment strings, the total length of the strings that is acceptable is up to 70 characters. Note, also, that a logical operator within a syntax string is considered to be two characters long in counting although it is regarded as a string of only one character in editing.

Reserved word

Each reserved word has a specific meaning provided by the system. You cannot use reserved words as symbol names.

Reserved words
ACT, CLR, MRST, ON, OFF, RST, STP, TASK
TUP, TRS, TCNT, CNxxx, PTxxx, WTxxx,
Bxxx, Pxxx, H???????
Application instruction names (See "5 APPLICATION INSTRUCTIONS.")
xxx: 3-digit decimal constant ????????: 8-digit hexadecimal constant

Constant

Long-word constants can be specified in HI-FLOW.



Variable

You can use physical PI/O registers such as X and Y.

In application instructions, (a) can be prefixed to PI/O for indirect specification of a variable, and a variable can be enclosed by brackets ([]) to handle it as long word.

The physical PI/O registers usable in HI-FLOW are listed below.

Variables ____ Bit type _____ Simple One-dimensional array such as X000 (5) Word type ____ Simple One-dimensional array such as XW000 (FW000) Long type ____ Simple Usable only in application instructions. When [FW000] is specified, FW000 and FW001 are handled as long words. [PI/O registers]

Item	Symbol		Range	Туре	Remarks
Registers	Extemal input registers	Х	000 to FFF	Bit	
		XW	000 to FF0	Word	
	Extemal output registers	Y	000 to FFF	Bit	
		YW	000 to FF0	Word	
	Communication link registers	G	000 to FFF	Bit	
		GW	000 to FF0	Word	
		А	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	
		М	000 to FFF	Bit	
		MW	000 to FF0	Word	
		Е	000 to FFF	Bit	
		EW	000 to FF0	Word	
		Ζ	000 to 3FF	Bit	
		ZW	000 to 3F0	Word	
		S	000 to BFF	Bit	
		SW	000 to BF0	Word	
	Other registers	J	000 to FFF	Bit	These registers establishes a link
		JW	000 to FF0	Word	to the ladder program.
		Q	000 to FFF	Bit	
		QW	000 to FF0	Word	
		HH	000 to 1FF	Bit	These registers establishes a link
					to another process.
		DW	000 to FFF	Word	
		FW	000 to BFF	Word	
	Timers	WT	000 to 255		Decimal notation
		PT	000 to 255		
	Counter	CN	000 to 127		Decimal notation
	Label	В	001 to 255 (You can		Decimal notation. Specifiable
			set a label consisting		for each process.
			of six or less		
			charactera.)		

Operator

As with the previous model, there are four operators: parentheses, arithmetic operators, relational operators and logical operators.

Item		Description	Priority
Operators	Logical	& (AND) (OR) ~ (NOT) ∧ (Exclusive OR)	5
	Arithmetic	* /	2
		+ -	3
	Relational	=, <>, <, >, >=, <=	4
	Parentheses	Parentheses can be nested at up to 7 levels.	1

Step comment

A step comment is a character string consisting of a combination of letters, numbers and special characters. You can write a statement comment as long as it fits in one line (74 columns). However, you do not always need to write a statement comment to the full length of a line.



* Note that, as regards the combined use of syntax, label, and comment strings, the total length of the strings that is acceptable is up to 70 characters. Note, also, that a logical operator within a syntax string is considered to be two characters long in counting although it is regarded as a string of only one character in editing.

Free label

You can create, in other than steps, labels to which symbols branch. These labels are called free labels. They are optional. A free label can be assigned any name (other than reserved words) beginning with a letter consisting of six or less characters and ending with a colon (:). You can use free labels only in other than steps.

LABEL:	Free label (up to 6 characters)
Joining point	Free comment (up to 70 characters)

Free comment

You can create comments in other than steps. These comments are called free comments. They are optional. A free comment is a character string consisting of a combination of letters, numbers and special characters. You can write a free comment as long as it fits in one line. You can add free comments to more identifiable points.

LABEL:Free label (up to 6 characters)Joining pointFree comment (up to 70 characters)

* As regards the combined use of a free label and free comment, the total length of the strings that is acceptable is up to 70 characters (including the colon ":" for a free label).

3.3 Process Information

A process consists of programs and process information.

In process information, additional information on processes is defined. Process information consists of five elements that you can change freely with process information commands.



Name

Name used in process information. You can assign unique name to the process with up to 16 characters.

Comment

Comment used in process information. You can assign comments to the process with up to 132 characters.
This chapter explains syntaxes consisting of symbols and destination labels with typical examples. In the following syntaxes, optional items are enclosed in brackets ([]).

Of the items enclosed in braces ($\{ \}$), select a desired one. The items followed by ~ are repeated.

4.1 Process Start and Process End

Process Start starts a process. Process End ends the process. Symbols for them are automatically added. You do not need to enter them.

Process Start sets a condition under which the process is stopped, reset or restarted, or PI/O is initialized. (See the descriptions of STP, RST, ACT and CLR.)

Process End performs processing if all routes except the local route have been ended. If not, Process End waits for them to be ended. At activation, Process End clear to 0 bit type PI/Os to be turned on by the local process if master reset is specified. (See the ON statement and parallel timer.)

The timers being used by the local process are handled according to the activation method. If the timer was activated with the TUP option specified, it is forcibly timed out. If the TRS option was specified for activation, the timer is reset to 0 after it expires. When no option was specified, the timer continues measurement.

[Syntax]



STP

• If the condition expression is satisfied during process execution, execution of the local process is stopped under the current execution state. (The process enters the stop state.)

- When the condition expression for STP is satisfied, the value of the timer and the values of bittype PI/Os to be turned on by the local process are retained. (See the ON statement and parallel timer.) However, note that it is unavoidable that the local process is turned on or off by other processes.
- When the condition expression for STP is satisfied, PI/O bits with option specification are turned on or off as specified. (If the condition expression is not satisfied, PI/O bits are turned on or off at each scan, contrary to the specification.)
- With the [,TCNT] option specified, the timer continues measurement even when the process enters the stop state. If the option is not specified, the timer value is retained.
- When the condition expression for STP is satisfied, the called process enters the stop state as with the calling process. However, the processes for which calls are ended or calls are not made are not affected.



- If a parallel timer is performing measurement when the condition expression for STP is satisfied, the timer is not affected. (This is because a call has been ended.)

If the condition expression for STP is satisfied during a call of this process, the called process also enters the stop state. If the [,TCNT] option is not specified, the timer value is retained.



- If the condition expression is satisfied while the process is being executed or stopped, the execution of the local process is stopped and the process enters the wait state with Process Start. (The process enters the reset state.)
- When the condition expression for RST is satisfied, the values of bit-type PI/Os to be turned on or off by the local process are retained. (See the ON and OFF statement and parallel timer.) However, note that it is unavoidable that the local process is turned on or off by other processes.
- When the condition expression for RST is satisfied, PI/O bits with option specification are turned on or off as specified. (If the condition expression is not satisfied, PI/O bits are turned on or off at each scan, contrary to the specification.)
- With the [,TUP] option specified, the value of the timer is set to the specified value and the timer is forcibly timed out. If the option is not specified, the timer is cleared to 0 and measurement is canceled.

• When the condition expression for RST is satisfied, the called process enters the executable state. In this case, PI/Os and timers are handled according to the activation method. The processes for which calls are ended or not made are not affected.



If a parallel timer is performing measurement when the condition expression for RST is satisfied, the timer is not affected. (This is because a call has been ended.)

If the condition expression for RST is satisfied during a call of this process, the called process enters the executable state. If master reset activation is specified, PI/Os are cleared. If zone activation is specified, PI/Os are retained. If a call is made with the [,TUP] option specified, the timer is forcibly timed out. If a call is made with the [,TRS] option specified, the timer is reset to 0 after it expires. When no option was specified, the timer continues measurement.

CLR

If the condition expression is satisfied in the stop or reset state, bit-type PI/Os which are used in the ON statement or parallel timer and are to be turned on by the local process are cleared to 0.

ACT

When the condition expression for STP or RST is not satisfied in the stop or reset state, process execution is resumed after the condition expression is satisfied. (The process enters the execution state.)

[Sample programs containing Process Start (ϕ)] (~ : Indicated that the same line is repeated.)

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

When X000 is on, the process enters the stop state. (The timer value is retained.) When X001 is on, the process is reset. (The timer is reset to 0 after it expires.) When X002 is on in the stop or reset state, the bit-type PI/Os in the ON statement or parallel timer used in the process are cleared to 0. When X000 and X001 are off and X003 is on, the process is executed.

2. • STP G000 & X002, TCNT [ON J000: OFF J001] ~, RST Q000, TUP

When both G000 and X020 are on, the process is stopped. (The timer continues measurement.)When the process is stopped, J000 is turned on and J001 is turned off.When Q000 is on, the process is reset. (The timer is forcibly timed out.)During process execution, J000 is turned off and J001 is turned on, at each scan.

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

When FW000 becomes smaller than DW000, the process is reset. (The timer is reset to 0 after it expires.)

When the process is reset, G100 is turned off.

When FW000 is greater than or equal to DW000 and FW001 is 0 in the stop or reset state, the process is executed.

G100 is turned on at each scan during process execution.

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

When Q001 is on, the process is reset. (The timer is reset to 0 after it expires.)

When the process is reset, J001 and G200 are turned on. When X200 is on in the stop or reset state, the ON statement and bit-type PI/Os in the parallel timer used in the process are cleared to 0.

J001 and G200 are turned off at each scan during process execution.

STP, RST, CLR, and ACT can be specified in any order in Process Start.

4.2 Route Start and Route End

- → No syntax
- ⊥ No syntax

Route Start starts a subroute. Route End ends the subroute. Be sure to use Route Start and Route End as a pair. Creation of subroutes enables synchronous syntaxes and selective branch syntaxes to be implemented. For operation after Route End is executed, see the chapter describing the synchronous/selective syntax execution order.

[Sample programs containing Route Start (op) and Route End (op)]

1.



4.3 Wait

The program waits at this step until the condition to proceed to the next step is satisfied. The condition is specified by a condition expression or the wait timer that makes the program wait until the specified time is reached.

[Syntax]

+ { condition-expression[, timer, output-bit] } { WTxxx (expression[, condition-expression]) }

Condition expression

A condition expression consists of bit-type or word-type numbers and operators.

Timer (usable on HI-FLOW system version 07-00 or later only)

- Monitoring timer that checks whether the condition expression yields an output ("true") within the specified period of time. A timer setting is possible in 100-millisecond units.
- Enter a constant in decimal notation.
- The acceptable range of settings is from 0 to 32767. If you set the range of -32768 through -1, the program runs as if 32768 through 65535 were set.
- The maximum number of timers that can be monitored simultaneously is 64. Do not put more than 64 timers under monitoring at the same time.

Output bit (usable on HI-FLOW system version 07-00 or later only)

- This bit is set if the condition is not met within the time period specified by the timer.
- The following registers can be designated as the output bit:
- $(\mathbf{Y}, \mathbf{G}, \mathbf{A}, \mathbf{R}, \mathbf{K}, \mathbf{M}, \mathbf{E}, \mathbf{Z}, \mathbf{S}, \mathbf{J}, \mathbf{Q})$
- The output bit is automatically reset unconditionally at the beginning of the monitoring process.
- Switching to the next step does not take place unless the condition is met within the timerspecified period of time.
- Even if the condition is met after output bit ON (set), the output bit is not turned OFF (reset).

Wait timer

- A wait timer delays execution for the specified time at the desired step. WT0 to WT255 (numbers are in decimals) can be used. A delay can be specified in 100 ms increments in the range from 0 to 32767 in decimals. If you set the range of -32768 through -1, the program runs as if 32768 through 65535 were set.
- When wait timers identified by the same number make the program wait at multiple steps, the step that occupies the timer first is awaited as specified. The other steps turn on the specified PI/O (by default, HH1FA) and wait until the step releases the timer. Therefore, the other steps are awaited longer than the specified time.
- A condition expression can be specified for a wait timer. In this case, execution is awaited until the condition expression is satisfied continuously for the specified time.

[Sample programs containing Wait (+)]

1. + X000

When X000 is on, the program proceeds to the next step.

2. + GW000<H2000

When GW000 becomes smaller than H2000, the program proceeds to the next step.

3. + X001 (FW000)

When the X register having the value of FW000 as a subscript is turned on during condition check, the program proceeds to the next step. (The condition check may vary at every time.)

4. + WT000 (100)

The program proceeds to the next step 10 seconds after the program reaches this step first.

5. + WT255 (10, X01F)

The program proceeds to the next step after it reaches this step then X01F is turned on continuously for one second.

6. + GW000>H2000, 100, Y000

Switching to the next step takes place when the GW000 value is greater than the H2000 value. If the GW000 value remains smaller than the H2000 value for a period of 10 seconds or longer, the Y000 is turned ON (set).

Even if the GW000 value is greater than the H2000 value after Y000 ON, the Y000 is not turned OFF (reset).

4.4 Box

Box controls PI/O output, data processing, and timers. Multiple Boxes separated by semicolons (;) can be specified.

[Syntax]



Assignment expression

An assignment expression assigns the result of a logical or arithmetic calculation to a variable. A one-dimensional array is allowed for expressions. Array subscripts are allowed only for word-type variables. The usable variables and operators are shown below.



Operands and results are assumed to be unsigned.

A multiplier and multiplicand must be both one word long.

Multipliers and multiplicands that are too long truncated to one word. The result is also one word long.

A divisor and dividend must be both one word long. A divisor and dividend that are too long are truncated to one word. The result is also one word long. If division by 0 is performed, the answer remains unchanged.

There is no answer back for the operation result state (such as normal termination or overflow). If answer back is required, use application instructions.

[Sample programs containing the assignment statement (山)]

1. 🛱 FW000=FW001+FW002

The current value of FW001 and that of FW002 are added and the result is assigned to FW000. Then the program proceeds to the next step.

2. 📥 YW000 (DW001)=HFFFF

/FFFF is assigned to the array of YW000 having the current value of DW001 as a subscript.

ON statement

The ON statement turns on the specified PI/O output bit (Y, G, A, R, K, M, E, Z, J, Q, or HH). When multiple bits separated by commas (,) are specified, multiple outputs can be obtained for PI/O. A one-dimensional array is allowed for PI/O output bits. Array subscripts are allowed only for word-type variables.

[Sample programs containing the ON statement (山)]

1. 🛱 ON Y000, Y00F:OFF Y001

Y000 and Y00F are turned on, and Y001 is turned off. Then the program proceeds to the next step.

2. 🛱 ON G000 (GW010)

The bit separated by the value of current GW010 from G000 is turned on. Then the program proceeds to the next step.

OFF statement

The OFF statement turns off the specified PI/O output bit (Y, G, A, R, K, M, E, Z, J, Q, or HH). When multiple bits separated by commas (,) are specified, multiple outputs can be obtained for PI/O. A one-dimensional array is allowed for PI/O output bits. Array subscripts are allowed only for word-type variables.

[Sample programs containing the OFF statement (_)]

1. 🗘 OFF Y000, Y001

Y000 and Y001 are turned off, and Y001 is turned off. Then the program proceeds to the next step.

2. 🗘 OFF G000 (GW010)

The bit separated by the value of current GW010 from G000 is turned off. Then the program proceeds to the next step.

Parallel timer

The parallel timer sends a waveform to a desired PI/O. t1 is a rising time and t2 is a falling time. When t1 is 0, PI/O for which ON is specified is just turned off after t2 elapses and PI/O for which OFF is specified is just turned on after t2 elapses. When t2 is 0 or is omitted, PI/O for which ON is specified is just turned on after t1 elapses and PI/O for which OFF is specified is just turned off after t2 elapses. After an instruction for waveform output is issued, the program proceeds to the next step soon.

Parallel timers can be specified in the range from PT000 to PT255. In t1 and t2 each, a time can be specified in 100 ms increments in the range from 0 to 32767. If you set the range of -32768 through -1, the program runs as if 32768 through 65535 were set.

If the specified timer is already in use at timer activation, the specified PI/O (by default, HH1F9) is turned on and the program waits until the timer is released.

Multiple PI/Os separated by commas (,) can be coded. Multiple statements separated by colons (:) can be coded. A one-dimensional array is also allowed. Usable bit PI/Os are Y, G, A, R, K, M, E, Z, J, Q and HH.



[Sample programs containing parallel timers (山)]

1. 🛱 PT000 (10, 10, ON Y000 : OFF Y001)

	When this step is passed, the program proceeds to the next step soon.	1 second later	2 seconds later
Y000	?→OFF	→ON	→OFF
Y001	?→ON	→OFF	→ON

2. 🛱 PT010 (20, ON G000 : OFF G001)

	When this step is passed, the program proceeds to the next step soon.	2 seconds later	
G000	?→OFF	→ON	\rightarrow
G001	?→ON	→OFF	\rightarrow

3. 🗘 PT255 (0, 30, ON J100 : OFF J101)

	When this step is passed, the program proceeds to the next step soon.	3 seconds later	
J100	?→ON	→OFF	\rightarrow
J101	?→OFF	→ON	\rightarrow

TUP (Timer Up)

TUP forcibly times out the timers that are performing measurement. When a wait timer is performing measurement, the timer value is reset to the specified value. As a result, the wait state is released and the program waiting for a forcible time-out proceeds to the next step. For a parallel timer, the timer value is set to t2 (t1 when t2 is omitted). As a result, the timer provides PI/O output earlier than the specified time. For a loop counter, the timer value is set to the end value. As a result, the program exits at the next loop check.

[Sample programs containing TUP (口)]

1. 🗘 TUP WT001, WT002, PT001, CN001

Wait timers 1 and 2, parallel timer 1, and counter 1 are forcibly timed out.

TRS (Timer Reset)

TRS resets the timers that are performing measurement. For wait timers and loop counters, TRS provides the same effect as TUP. For parallel timers, however, TRS resets t1 and t2. The state of the specified PI/O is the same as when a timer reset is indicated.

[Sample programs containing TRS (亡)]

1. 🛱 TRS WT001, WT002, PT001, CN001

Wait timers 1 and 2, parallel timer 1, and counter 1 are reset.

4.5 Control Box

Control Box activates (reactivates), stops or resets other processes, or clears PI/O.

[Syntax]

ACT

	Item	Description
1	Function outline	ACT activates the specified processes. Specifiable processes are P0 to P255. A range of processes can be specified with a hyphen (-). When a step number is omitted, execution starts with step 1. The specified step may not be a main route. After activation, the program proceeds to the next step soon.
2	Behavior of the activated process	When Process End finishes execution, the activated process is executed again from the process start point at the next scan. (This is also true when a step is specified.)
3	Activating a process being executed	The ACT bit that represents the Control Box result is turned on. Then the program proceeds to the next step (by default, HH1FF).
4	Activating a nonexisting process	
5	Activating a stopped process	The stopped process is activated and execution is resumed.
6	Activating a reset process	The reset process is activated and execution is resumed from the process start point.
7	Instruction for timer states	When Process End or Escape is executed with the ,TUP option specified or the executable state is entered, the parallel timers occupied by the local process are forcibly timeout. When Process End or Escape is executed with the ,TRS option specified or the executable state is entered, the parallel timers occupied by the local process are reset.
8	Activation with master reset specified	When Process End or Escape is executed with the ,MRST option specified or the executable state is entered, bit-type PI/Os turned on by the local process are cleared to 0. (ON statement, parallel timer)
9	Activating a CPMS task	Specify the ,TASK,factor-number options and specify a CPMS task (1 to 127) with Pxxx. Then issue the RLEAS and QUEUE macros.

RST

	Item	Description
1	Function outline	RST resets the specified processes. Specifiable processes are P0 to P255. A range of processes can be specified with a hyphen (-). After Control Box with RST specified is issued, the program proceeds to the next step soon.
2	Behavior of the reset process	Execution of the specified process is canceled. The process enters the reset state and waits to be reexecuted with Process Start. (When the process is activated by another process with ACT or the ACT condition is satisfied to start the local process, the process is reexecuted.)
3	Instruction for timer states	The parallel timers occupied by the process with the ,TUP option specified are forcibly timed out. When the ,TUP option is not specified, the timers are reset. The option is valid only for the specified process. The called processes are not affected.
4	PI/O for the reset process	When the process has been activated with master reset specified, bit-type PI/Os turned on or off by the local process are cleared to 0.
5	Resetting a stopped process	Execution of the specified process is canceled. The process enters the reset state and waits to be reexecuted with Process Start. (When the process is activated by another process with ACT or the ACT condition is satisfied to start the local process, the process is reexecuted.)
6	Resetting a nonexisting process	The RST bit that represents the Control Box result is turned on. Then the program proceeds to the next step (by default, HH1FD).
7	Resetting the local process	Specify the local process number with the parameter.
8	Stopping a CPMS task	Specify the ,TASK option and specify a CPMS task with Pxxx. Then issue the ABORT macro.

STP

	Item	Description
1	Function outline	RST stops the specified processes. Specifiable processes are P0 to P255. A range of processes can be specified with a hyphen (-). After Control Box with STP specified is issued, the program proceeds to the next step soon.
2	Behavior of the stopped process	Execution of the specified process is stopped. The process enters the stop state and waits to be reexecuted at the current execution point.
3	Conditions for reexecution	The process is reexecuted when it is activated by another process with ACT or the ACT condition is satisfied to start the local process, the process is reexecuted.
4	Instruction for timer states	The parallel timers occupied by the process with the ,TCNT option specified continue measurement. When the ,TCNT option is not specified, the timers are stopped. The option is valid for all processes linked by the specified process.
5	PI/O for the stopped process	When the process has been activated with master reset specified, bit-type PI/Os turned on or off by the local process are cleared to 0.
6	Stopping a nonexisting process	The STP bit that represents the Control Box result is turned on. Then the program proceeds to the next step (by default, HH1FE).
7	Resetting a reset process	
8	Stopping the local process	Specify the local process number with the parameter.

CLR

	Item	Description		
1	Function outline	Bit-type PI/Os turned on or off by the specified processes are cleared to 0. Specifiable processes are P0 to P255. After Control Box with CLR specified is issued, the program proceeds to the next step soon. This function is valid only when the specified processes are stopped or reset. PI/Os used by other processes are cleared without checking their usage states. A range of processes can be specified with a hyphen (-).		
2	Clearing a nonexisting process	The CLR bit that represents the Control Box result is turned on. Then the program proceeds to the next step (by default, HH1FC).		
3	Clearing a process being executed			
4	Clearing a process not activated			

[Sample programs containing Control Box (

1. 🔳 ACT P1-P5, MRST

Processes 1 to 5 are activated in master reset mode, starting from step 1. Then the program proceeds to the next step. When Process End or Escape is executed or the process enters the executable state, the parallel timers continue measurement.

2. 📫 ACT P100, 5, TUP

Process 100 is activated in zone mode, starting from step 5. Then the program proceeds to the next step. When Process End or Escape is executed or the process enters the executable state, the parallel timers are forcibly timed out.



When the RLEAS macro is issued to CPMS task 80 and the QUEUE macro is issued with factor 3, the program proceeds to the next step.

4. 🔳 RST P10

Process 10 is reset then the program proceeds to the next step. The parallel timers are reset, which were performing measurement when Control Box with RST specified was issued.

5. 🗰 RST P11, TUP

Process 11 is reset then the program proceeds to the next step. The parallel timers are forcibly timed out, which were performing measurement when Control Box with RST specified was issued.

6. 🔳 RST P12, TASK

After the ABORT macro is issued to CPMS task 12, the program proceeds to the next step.



Process 50 is stopped then the program proceeds to the next step. The parallel timers and/or wait timers are stopped, which were performing measurement when Control Box with STP specified was issued.

8. 🔳 STP P51, TCNT

Process 51 is stopped then the program proceeds to the next step. The parallel timers and/or wait timers continue measurement without being stopped, which were performing measurement when Control Box with STP specified was issued.

9. 🔳 CLR P40

Bit-type PI/Os used by process 40 are cleared to 0 then the program proceeds to the next step.

4.6 Repeat Start and Repeat End

Repeat Start and Repeat Stop bracket the steps to be executed repeatedly. If Repeat Start steps and Repeat Stop steps are not paired correctly within the same loop, a syntax error is detected. An increment is added to the initial value after each repetition. Repetition continues until the result of addition exceeds the end value. If the initial value is greater than the end value, the program proceeds to the next step without executing the steps between Repeat Start and Repeat Stop. When an increment is omitted, 1 is assumed. When 0 is specified as an increment, the program enters an infinite loop.

The setting range of the initial value, the end value, and increment is 0 through 32767. If you set the range of -32768 through -1, the program runs as if 32768 through 65535 were set.

[Syntax]

/+ CNxxx (initial-value, end-value {, increment})

(xxx: Decimal number from 000 to 127)

└┼╯ No syntax

[Sample programs containing Repeat Start (,+,) and Repeat End (,+)]



3.

CN127 (1, 5, 2)

The program executes the steps between Repeat Start and Repeat End repeatedly 10 times, then proceeds to the next step after Repeat End. Repeat Start is executed immediately after Repeat End is executed.

The program execute

The program executes the steps between Repeat Start and Repeat End repeatedly three times, then proceeds to the next step after Repeat End.

CN001 (FW000, FW001, FW002)

The values of FW000, FW001, and FW002 when Repeat Start is passed first are the initial value, end value and increment, respectively.

4.7 If

If judges whether the specified condition expression is true or false and performs processing accordingly. When the condition expression is satisfied (true), If executes the steps between the comma (,) and semicolon (;). When the condition expression is not satisfied (false), If executes the steps after the semicolon. When the condition expression is not satisfied but there are no steps after the semicolon, the program proceeds to the next step. When a label is specified after the comma or semicolon, the program branches to the label.

[Syntax]

\diamondsuit condition-expression	{ destination-label (Bxxx) { free-label }	}	
	{ON/OFF-statement}	-{: ON/OFF-statement}	
	{assignment-statement}	{: assignment-statement}	
	{ACT-statement}	{: ACT-statement}	
	{STP-statement}	<pre>{: STP-statement}</pre>	
	{RST-statement}	{: RST-statement}	~
	{CLR-statement}	{: CLR-statement}	
	{TUP-statement}	<pre>{: TUP-statement}</pre>	
	{TRS-statement}	{: TRS-statement}	
	{PT-statement}	_{: PT-statement}	
	 {; destination-label (Bxxx) {; free-label } {; ON/OFF-statement} {; assignment-statement}) } {: ON/OFF-statement} {: assignment-statement} {: ACT-statement}	
	{; STP-statement} {; RST-statement}	{: STP-statement} {: RST-statement}	~
	{; CLR-statement}	{: CLR-statement}	
	{, TRS-statement}	{ TRS-statement}	
	{; PT-statement}	{: PT-statement}	

(xxx: Decimal number from 1 to 255)

<NOTICE>

A branch to another process cannot be made but a branch to another route can be made. In actual execution, however, the following branches may not be handled correctly:

- Branch to the inside of a loop between Loop Start and Loop End
- Branch from the inside of parallel processing
- Branch no the inside of parallel processing
- Branch to a route being executed

[Sample programs containing If (\diamondsuit)]

1. 🔷 X000, B1 ; LABEL

When X000 is on, the program jumps to the step having label B1. When X000 is off, the program jumps to the next step after the step having label LABEL.

2. 💠 H0 < > (YW000 & H3000), ON Q005

When the logical product of YW000 and H3000 is not 0, Q005 is turned on. If the logical product is 0, the program proceeds to the next step without doing nothing.

3. 🔆 Q000, FW100=FW100+1 ; ACT P10

When Q000 is on, 1 is added to FW001. Then the program proceeds to the next step. When Q000 is off, process 10 is activated with ACT. Then the program proceeds to the next step.

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

When GW000 is 4, process 6 is stopped and process 7 is reset. Then the program proceeds to the next step.

When GW000 is not 4, EW000 is set to 8 and J000 is turned on. Then the program proceeds to the next step.

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

When X010 is on, J000, J001, J002, and J003 are turned on. Then the program proceeds to the next step. When X010 is off, the program jumps to the next step after the step having label ERRLB.

4.8 Jump

Jump causes the program to unconditionally jump to the specified label within the process. Label B1 to B255 are specifiable within one process. In HI-FLOW, free labels can be specified. (You can assign any names consisting of up to six characters to free labels. Free labels can be set in other than steps.)

[Syntax]

{ destination-label (Bxxx) }
 { free-label }

<NOTICE>

A branch to another process cannot be made but a branch to another route can be made. In actual execution, however, the following branches may not be handled correctly:

- Branch to the inside of a loop between Loop Start and Loop End
- Branch from the inside of parallel processing
- Branch no the inside of parallel processing
- Branch to a route being executed

[Sample programs containing Jump (∟)]

1. L, B1

The program jumps to the step having label B1 then starts execution from the step soon.

2. Ļ ERRBLK

The program jumps to the next step after the step having label LABEL then starts execution from the step soon.

4.9 Escape

Escape forcibly terminates the local process. When the local process is a main process, Escape forcibly terminates all routes and the process enters the executable state. The processes being called, if any, are all escaped. The timers being used by the local process are handled according to the activation mode. (See the TUP and TRS options.) When the local process is a subprocess, Escape functions in the same way as when the local process

is a main process is a susprocess, Escape functions in the same way as when the focal process is a main process are except that execution points are returned to the main process in the same scan. When the local process was activated in master reset mode, the bit-type PI/Os to be turned on by the local process are cleared to 0. (See the ON statement and parallel timer.)

[Syntax]

imes No syntax

[Sample programs containing Escape (\times)]



4.10 Parallel Start and Parallel End

A Parallel Start and Parallel End pair represents a portion to be synchronously processed. After the synchronized subroute is activated, Parallel Start causes the program to proceed to the next step after the local route. After all joined routes are terminated, Parallel End instructs execution of the next step after the local route.

In the previous models, Parallel End monitored termination of the joined subroute (i.e., the main route was being executed), execution of the next step was delayed one scan. In this model, however, each of Parallel End and Route End checks whether it was joined last. If Parallel End or Route End was joined last, it instructs execution of the next step joining the main route. If not, it instructs termination of the local route (the main route is not always being executed). This eliminates a delay of one scan.

[Syntax]

⊨ No syntax ⊨ No syntax

[Sample programs containing Parallel Start (⊨) and Parallel End (⊨)]



4.11 Select, Wait in Selective Branching and Select End

A set of Select, Wait in Selective Branching and Select End represents a portion to be processed for selective branching.

After the selective branching route is activated, Select causes the program to proceed to Wait in Selective Branching in the local route. (Select and Wait in Selective Branching or Route Start and Wait in Selective Branching must be consecutive.)

When the condition expression for the local route is satisfied, Wait in Selective Branching terminates execution of the other routes. The program proceeds to the next step in the local route. In the previous models, the main route was always being executed. (Both the main route and the selected route were being executed.) In this model, however, selection of a subroutine terminates the main route. (Only the selected route is executed.)

Condition expressions are checked sequentially from the leftmost route on the screen. When multiple conditions are met in the same scan, therefore, the leftmost route is selected.

In the previous models, Select End monitored termination of the joined subroutes (i.e., the main route was being executed), execution of the next step was delayed one scan.

In this model, however, when a subroute is selected, Route End of the route activates the main route and instructs execution of the next step joining the main route. This eliminates a delay of one scan.

Also in the previous models, Select End and Select had to be present in the same route. In this model, however, they may not need to be in the same route (they may not need to join the branch source route).

[Syntax]

- ⊢ No syntax
- EFE condition-expression [, timer, output-bit]
- \ltimes No syntax
- Timer (usable on HI-FLOW system version 07-00 or later only)
- Output bit (usable on HI-FLOW system version 07-00 or later only)
- * For the timer and output bit, see "4.3 Wait."

[Sample programs containing Select (\vdash), Wait in Selective Branching ($\equiv \ddagger \Rightarrow$ and Select End (\ltimes)]



4.12 Multi-entry

When a condition expression is set in the symbol for Select End, it is handled as Multi-entry. When the condition expression is satisfied during process execution, execution is resumed from the step containing Multi-entry. (Even during execution of the first process, execution starts from the first process when the condition expression is satisfied.) The condition expression is checked at the beginning of scanning. This may delay execution a maximum of one scan. Steps are checked against the condition sequentially from the one having the smallest step number. If several conditions are satisfied in the same scan, therefore, execution is resumed from the step

Multi-entry can also be set in subroutes.

having the smallest step number.

When the condition is satisfied and the process is executed, all routes except those containing parallel timers, wait timers, counters, a process being called and Multi-entry are initialized but the PI/O values remain unchanged.

[Syntax]

 \ltimes condition-expression

<NOTICE>

- If Multi-entry is set between Loop Start and Loop End, the program may not run correctly.
- Multi-entry cannot be set in a subroute having a synchronous syntax.

[Sample programs containing Multi-entry (⇐)]

1. ⊨ X000

When X000 is on, reexecution starts from this step.

2. ∣≤ GW000<H2000

When GW000 becomes smaller than H2000, reexecution starts from this step.

4.13 Call

Call makes a subroutine call for the process specified by one of P0 to P255. Execution starts from the step specified by the [,step-number] option. (When a step number is omitted, execution starts from Process Start.)

When the specified process or step is not found or the local process is called, the CALL bit that represents the Control Box result is turned on then the program proceeds to the next step. If the specified process is already being executed, the program waits until it can call the process (until the process shifts to the executable state). However, a process which was called with ACT and is reset cab be called.

A subprocess can further call another process. Up to 16 nesting levels are allowed.

To make a master reset call, specify the [,MRST] option. When a master reset call is made, bittype PI/Os turned on by the local process are cleared to 0 on the following timing: the called process is terminated; Escape is executed; or the executable state is entered.

When the [,TUP] option is specified, the parallel timers occupied by the local process are forcibly timed out at execution of Process End or Escape or at shift to the executable state.

When the [,TRS] option is specified, the parallel timers occupied by the local process are reset at execution of Process End or Escape or at shift to the executable state.

When the option is not specified, the parallel timers continue measurement even after the process is terminated.

[Syntax]

```
□ Pxxx [, step-number] [, MRST] { [, TUP] }
{ [, TRS] }
```

[Sample programs containing Call (

```
1. 🛱 P1
```

Process 1 is called in zone mode, starting from step 1. The parallel timers occupied by the called process continue measurement when Process End or Escape is executed or the process executable state is entered.

2. 🛱 P2, 5, MERST

Process 2 is called in master reset mode, starting from step 5. The parallel timers occupied by the called process continue measurement when Process End or Escape is executed or the process executable state is entered.

3. 🛱 P3, TUP

Process 3 is called in zone mode, starting from step 1. The parallel timers occupied by the called process are forcibly timed out when Process End or Escape is executed or the process executable state is entered.

4.14 Function

Function supplements the operation and data processing functions that are supported by Box. For details, see the chapter explaining application instructions.

[Syntax]

♦ application-instruction-name parameter [, parameter] ~

4.15 Wait with Previous State Cleared

Wait with Previous State Cleared provides the same effect as Wait before the shift conditions are met. After these conditions are met, it functions differently. When the previous step is an ON statement or process call, Wait with Precious State Cleared turns off its PI/Os before making the program proceed to the next step. When the previous step is neither an ON statement nor a process call, Wait with Previous State Cleared is the same as Wait. (The program proceeds to the next step without PI/Os being turned off.)

When starting execution from this step through a branch, note that the previous state of the branch source is not cleared.

This step is conform to the SFC standard.

[Syntax]

+* {condition-expression}
{WTxxx (expression [, SB] [, condition-expression]) }

THIS PAGE INTENTIONALLY LEFT BLANK.

5 APPLICATION INSTRUCTIONS

5.1 Overview

HI-FLOW syntaxes support only assignment for word-length data, arithmetic operations, and logical operations as operation and data processing functions. To supplement this, HI-FLOW supports application instructions equivalent to ladder diagrams.

5.2 Usage

Program application instructions in the following format:

o application-instruction parameter [, parameter] ~

5.3 Parameters

Unlike ladder operation functions, HI-FLOW accepts difference in type between application instructions and parameters specified in them.

(Ladder)



FW000+FW001→FW002 All parameters are word-length parameters.

(HI-FLOW)



In general, three types of parameters are used: source (S), destination (D), and result (R). Three types of parameters, bit-type PI/O, word-type PI/O and constant, are provided. In HI-FLOW application instructions, the following four types of addressing modes can be specified for parameters:

- 1. Direct word-length specification: Specify parameters as they are.
- 2. Direct long-length specification: Enclose parameters in brackets ([]).
- 3. Indirect word-length specification: Prefix @ to the description in 1. above.
- 4. Indirect long-length specification: Prefix *(a)* to the description in 2. above.

Addressing mode			Parameters			
		Bit-type PI/O	Word-type PI/O	Constant		
		X000 Data 1 X001 Data 2	FW000Data 3FW001Data 4	XXXX YYYYYYYY		
		Data 1 and data 2 Data a Data b	Data 3 and data 4 Data c Data d	XXXX Data e Data f YYYYYYYY Data g Data h		
1. Direct word-le	ength	Result obtained by	Data 3	XXXX.		
		ANDing data 1 with data 1		For long-length data		
				YYYYYYYY, only the low-order word is valid.		
	Examples	X000	FW000	1230		
				H20000000		
2. Direct long-length		Result obtained by	Data 3 and data 4	XXXX, YYYYYYYY.		
		ANDing data 1 and data 2		XXXX is handled as long-		
		with data 1		length data.		
	Examples	[X000]	[FW000]	[H1234] [H2000000]		
2 In dias of second	lou oth	Donomoton omon	Data a	[H20000000]		
5. Indirect word-	length	Parameter error	When data 3 and data 4 are	For YYYYYYY data g		
			odd numbers, an error is	is valid. When XXXX		
			detected.	and YYYYYYYY are odd		
				numbers, an error is		
				detected.		
	Examples		@ FW000	@ HFFF0		
				@ H180000		
4. Indirect long-length		Parameter error	Data c and data d. When data 3 and data 4 are	For XXXX, data e and data f are valid. For VVVVVVV data g and		
			detected	data h are valid When		
				XXXX and YYYYYYYY		
				are odd numbers, an error		
				is detected.		
	Examples		@ [FW000]	@ [HFFF0]		
				@ [H180000]		

5 APPLICATION INSTRUCTIONS



5.5 System Error Flags

Flags are set in SW020, according to the execution result of the HI-FLOW application instruction.

Flags



Each flag is set according to the flag setting condition determined for each application instruction. When the following conditions hold, the pertinent flags are set for all application instructions.

- Error flag: This flag is set when (1) the number of parameters used in the application instruction is invalid, (2) CPU memory is protected, (3) the address or PI/O specified by the result parameter (R) points to an address in a protected area, or (4) the specified PI/O is in error (for example, it cannot be used).
- Overflow flag: This flag is set when the value of the operation result exceeds the range (word or long) specified by the result parameter (R). The limit of the size is set in the operation result.

Word-length	Positive overflow: 7FFF
	Negative overflow: 8000
Long-length	Positive overflow: 7FFFFFFF
	Negative overflow: 8000000

5 APPLICATION INSTRUCTIONS

Explanation of Functions 5.6

This section details individual application instructions in the following format:

Application	instruction name	Function name
1		

		iume i	unction	nume		. r	- Processing of the		
ADD	ADD ———						application instruction is		
Function	Adds the conte area specified b	ie contents of the source to those of the destination and stores the result in the ecified by the result parameter.					outlined.		
Parameters and	¢ add s, d,	, R		S	$+D \rightarrow R$ —		- Processing is illustrated.		
processing	S: Source D: Destination R: Result						- A parameter list is shown.		
Flags	The settings of	the E and V	flags change	e. The othe	r flags are turned off. —		- The flags whose contents		
Processing time	0.41 ms						change after execution of		
Remarks							the application instruction		
Examples	Q ADD FW0	00, FW001	, FW002				are shown.		
	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$					 The time required for processing by the S10/2α is shown. Notes to be followed are described. 			
Valid	FW100 7FFF The V flag is turned on.						Main examples are shown.		
parameters	S, D, R	S, D, R Bit-type Word-type PI/O PI/O Constant		- Parameters specifiable as					
\triangle : A parameter error is detected when an odd-	Direct word-length	0	0	0			(D), and result (R)		
numbered	Direct long-length	0	0	0			parameters are shown.		
	Indirect word-length	×	\bigtriangleup	\bigtriangleup			○: Can be specified		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup			△: Can be specified on conditions		
							\times : Cannot be specified		

ADD	ADD									
Function	Adds the contents of the source to those of the destination and stores the result in									
-	the area specified by the result parameter.									
Parameters and	\bigcirc ADD S, D, R						$S+D \rightarrow R$			
processing	S: Source									
	D: Destination									
	R: Result									
Flags	The settings of the E and V flags change. The other flags are turned off.									
Processing time	0.41 ms									
Remarks										
Examples O ADD FW000, FW001, FW002										
	FW000 0001									
		FW001	00FF			-				
		FW002	0100	_	(Ð				
🔷 ADD H1234, [GW000], FW100										
	H1234 — 🕀 — GW000 🛛 0010									
	GW001 0011									
	FW100 7FFF The V flag is turned on.									
Valid										
parameters △: A parameter error is detected when an odd- numbered address is used.	S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant	
	Direct word-length	0	0	0	Direct word-lengt	ct l-length	0	0	×	
	Direct long-length	\bigcirc	0	\bigcirc	Direction Direct	ct ·length	\bigcirc	\bigcirc	\times	
	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indir word	ect -length	×	\bigtriangleup	\bigtriangleup	
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indir long-	ect ·length	×	\bigtriangleup	\bigtriangleup	

SUB	SUBTRACT											
Function	Subtracts the contents of the destination from those of the source and stores the result in the area specified by the result parameter.											
Parameters and	$\bigcirc SUBS, D, R \qquad S-D \rightarrow R$											
processing	S: Source											
	D: Destination											
	R: Result											
Flags	The settings of the E and V flags change. The other flags are turned off.											
Processing time	0.41 ms											
Remarks												
Examples	 \$\begin{pmatrix} SUB FW000, FW001, FW002 & \$\begin{pmatrix} FW000 & \$\begin{pmatrix} 0100 & \$\\expact{FW001} & \$\begin{pmatrix} 00FF & \$\\expact{D001} & \$\expact{D001} & \$\expact{D001} & \$\expact{D001} & \$\end{pmatrix}\$											
Valid												
parameters	S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant			
\triangle : A parameter error is detected when an odd- numbered address is used.	Direct word-length	0	0	0		Direct word-length	\bigcirc	0	×			
	Direct long-length	0	0	\bigcirc		Direct long-length	\bigcirc	0	×			
	Indirect word-length	×	\bigtriangleup	\bigtriangleup		Indirect word-length	×	\bigtriangleup	\bigtriangleup			
	Indirect long-length	×	\bigtriangleup	\bigtriangleup		Indirect long-length	×	\bigtriangleup	\bigtriangleup			
INC	+1 (INCREMENT)											
---------------------------	-------------------------	---	------------------	------------------	--------	----------------------------	--	--	--	--	--	--
Function	Increments	the cont	ents of the	e source b	y one.							
Parameters and		3				$S+1 \rightarrow S$						
processing	S: Source	S: Source										
Flags	The setting	The settings of the E and V flags change. The other flags are turned off.										
Processing time	0.29 ms	0.29 ms										
Remarks												
Examples	↓ INC F											
	F	FW000 +1										
	0 INC [GW000]											
	GW000→ (+1)											
		GW001										
		GW000	and GW	001 are i	ncreme	nted by one, assuming that						
		tney are	long-len	gth varia	bles.							
Valid		Ditter	\A/and tura									
parameters	S	PI/O	PI/O	Constant								
\triangle : A parameter	Direct word-length	\bigcirc	0	×								
when an odd-	Direct long-length	\bigcirc	0	×								
numbered	Indirect word-length	×	\bigtriangleup	\bigtriangleup								
uuu1055 15 useu.	Indirect long-length	×	\bigtriangleup	\bigtriangleup								

DEC	-1 (DECREMENT)											
Function	Decrement	s the con	tents of th	e source l	by one.							
Parameters and	0 dec	S				$S-1 \rightarrow S$						
processing	S: Source											
Flags	The setting	s of the H	E and V fl	ags chang	e. The	e other flags are turned off.						
Processing time	0.29 ms	0.29 ms										
Remarks												
Examples	0 dec	O DEC FW000										
		FW000		∼	→ (-1							
	4											
		GW0001										
		GW001										
		GW000	and GW	/001 are	decrem	ented by one, assuming that						
		they are	e long-ler	ngth varia	ables.							
Valid					l							
parameters	S	Bit-type PI/O	Word-type PI/O	Constant								
\triangle : A parameter	Direct word-length	\bigcirc	0	×								
error is detected when an odd-	Direct long-length	0	0	×								
numbered	Indirect word-length	×	\bigtriangleup	\bigtriangleup								
uuuross 15 useu.	Indirect long-length	×	\bigtriangleup	\bigtriangleup								

MULTIPL	Y									
Multiplies result in the	the conte	nts of the ecified by	source by the result	those param	of the oneter.	destinatio	n and stor	res the		
MULS: SourceD: DestinatR: Result										
The settings of the E and V flags change. The other flags are turned off.										
0.49 ms (w 1.35 ms (w	0.49 ms (when both S and D are word-length parameters) 1.35 ms (when either S or D is a long-length parameter)									
 MUL FW000, FW001, FW002 FW000 0100 FW001 00FF FW002 FF00 MUL H22, [GW000], FW100 H0022 S GW001 0010 GW001 0011 FW100 7FFF The V flag is turned on. 										
S, D	Bit-type PI/O	Word-type PI/O	Constant	Γ	R	Bit-type PI/O	Word-type PI/O	Constant		
Direct word-length	\bigcirc	0	0	Dire wor	ect d-length	\bigcirc	0	\times		
Direct long-length	0	0	0	Dire long	ect g-length	0	0	×		
Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indi wor	irect d-length	×	\bigtriangleup	\bigtriangleup		
Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indi long	\bigtriangleup					
	MULTIPLY Multiplies result in the C MUL S: Source D: Destinar R: Result The setting 0.49 ms (w 1.35 ms (w C MUL C MUL C MUL C MUL C MUL	MULTIPLY Multiplies the conteresult in the area spector MUL S, D, R S: Source D: Destination R: Result The settings of the E 0.49 ms (when both 1.35 ms (when eithe 0.49 ms (when both 1.35 ms (when eithe 0.49 ms (when oth 1.35 ms (when eithe 0.49 ms (when both 1.35 ms (when eithe 0.49 ms (when eithe 0.49 ms (when both 1.35 ms (when eithe 0.49 ms (w	MULTIPLY Multiplies the contents of the result in the area specified by	MULTIPLY Multiplies the contents of the source by result in the area specified by the result \diamondsuit MUL S, D, R S: Source D: Destination R: Result The settings of the E and V flags change 0.49 ms (when both S and D are word-1 1.35 ms (when either S or D is a long-left) \diamondsuit MUL FW000, FW001, FW002 FW001 0100 FW002 FF00 \clubsuit MUL H22, [GW000], FW100 H0022 \longrightarrow GW FW100 7FFF The S, D Bit-type PI/O Constant Direct O Order-length O Direct O Indirect X Indirect X Indirect X Indirect X Indirect X	MULTIPLY Multiplies the contents of the source by those result in the area specified by the result param \Phi MUL S, D, R S: Source D: Destination R: Result The settings of the E and V flags change. 0.49 ms (when both S and D are word-length 1.35 ms (when either S or D is a long-length parameter for the formation of the formation	MULTIPLY Multiplies the contents of the source by those of the result in the area specified by the result parameter.	MULTIPLY Multiplies the contents of the source by those of the destination result in the area specified by the result parameter. \bigcirc MUL S, D, R $S \times D \to R$ S: Source $D: Destination$ $S \times D \to R$ D: Destination $S \times D \to R$ $S \times D \to R$ S: Source $D: Destination$ $S \times D \to R$ The settings of the E and V flags change. The other flags are 0.49 ms (when both S and D are word-length parameters) 1.35 ms (when either S or D is a long-length parameter) 1.35 ms (when either S or D is a long-length parameter) MUL FW000, FW001, FW002 $FW000$ $FW000$ 0100 $FW000$ 0100 $FW000$ 0100 $FW000$ 0100 $FW100$ $FW100$ $FW100$ $TFFF$ $Bit-type$ O $Direct$ O	MULTIPLY Multiplies the contents of the source by those of the destination and stor result in the area specified by the result parameter. \diamondsuit MUL S, D, R $S \times D \rightarrow R$ S: Source D D: Destination R: Result The settings of the E and V flags change. The other flags are turned or 0.49 ms (when both S and D are word-length parameters) 1.35 ms (when either S or D is a long-length parameters) 1.35 ms (when either S or D is a long-length parameter) \bigcirc MUL FW000, FW001, FW002 FW000 \bigcirc 0100 FW001 \bigcirc 00FF FW002 \bigcirc FF00 \bigcirc MUL H22, [GW000], FW100 H0022 — \bigotimes \bigcirc GW000 \bigcirc MUL H22, [GW000], FW100 H0022 — \bigotimes \bigcirc GW001 \bigcirc FW100 $7FFF$ The V flag is turned on. $ S, D Bit-type Word-type Constant \\ Direct \bigcirc \bigcirc \\ Direct $		

| DIVIDE | |
 | DIVIDE

 |

 | | |
 | | | | |
|--|--
--
--
--
--
--
---|--
---|--|---|--|--|--|
| Divides the quotient in | e contents
the area | s of the so
specified
 | urce by th
by the res

 | ose o
ult pa

 | of the des
arameter. | stination a | and stores
 | the | | | |
| ф div s | 6, D, R |
 |

 |

 | S÷D | $\rightarrow R$ | | |
 | | | | |
| S: Source
D: Destina
R: Result | tion |
 |

 |

 | | | | |
 | | | | |
| The setting | The settings of the E and V flags change. The other flags are turned off. |
 |

 |

 | | | | |
 | | | | |
| 0.49 ms (w
1.35 ms (w | 0.49 ms (when both S and D are word-length parameters)
1.35 ms (when either S or D is a long-length parameter) |
 |

 |

 | | |
 | | | | |
| When D = | 0, the E f | lag is turr
 | ned on an

 | d notl

 | hing is pe | erformed. | | |
 | | | | |
| DIV FW000, FW001, FW002 FW000 0100 FW001 0010 W002 0010 W100 H0022 GW000], FW100 H0022 GW001 GW001 O011 | |
 |

 |

 | | |
 | | | | |
| | Rit-type | Word-type
 |

 |

 | | Rit_type | Word-type
 | | | | |
| S, D | PI/O | PI/O
 | Constant

 |

 | R | PI/O | PI/O
 | Constant | | | |
| Direct
word-length | \bigcirc | 0
 | 0

 | D
w

 | virect
ord-length | \bigcirc | 0
 | × | | | |
| Direct
long-length | \bigcirc | 0
 | 0

 | D
lo

 | virect
ong-length | \bigcirc | 0
 | × | | | |
| Indirect
word-length | × | \bigtriangleup
 | \bigtriangleup

 | In
w

 | ndirect
ord-length | × | \bigtriangleup
 | \bigtriangleup | | | |
| Indirect
long-length | × | \bigtriangleup
 | \bigtriangleup

 | In
lo

 | ndirect
ong-length | × | \bigtriangleup
 | \bigtriangleup | | | |
| | DIVIDE
Divides the
quotient in
\bigcirc DIV S
S: Source
D: Destina
R: Result
The setting
0.49 ms (w
1.35 ms (w
When D =
\bigcirc DIV F
F
F
F
F
F
F
F
F
F
F
F
F
F | DIVIDEDivides the contentsquotient in the area \diamondsuit DIV S, D, RS: SourceD: DestinationR: ResultThe settings of the F0.49 ms (when both1.35 ms (when both1.35 ms (when eitherWhen D = 0, the E f \bigcirc DIV FW000, FFW000FW000FW000FW000FW000FW000FW000FW000FW000FW000FW000FW000FW000FW000FW000FW000FW100DirectODirectODirectODirectODirectODirectODirectODirectODirectODirectODirectOIndirectXIndirect <td colspan<="" td=""><td>DIVIDEDivides the contents of the so
quotient in the area specified\bigcirc DIV S, D, RS: SourceD: DestinationR: ResultThe settings of the E and V fl0.49 ms (when both S and D a
1.35 ms (when either S or D iWhen D = 0, the E flag is turn\bigcirc DIV FW000, FW001, FW001\bigcirc DIV FW000, FW001, FW001\bigcirc DIV FW002\bigcirc DIV H22, [GW000], FW
0010\bigcirc DIV H22, [GW000], FW
H0022 — $\bigcirc$$\bigcirc$ DIV H22, [GW000], FW
H002 — $\bigcirc$$\bigcirc$ DIV H22, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H22, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H23, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H24, [GW000], FW
H002 = $\bigcirc$$\bigcirc$ DIV H25, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H26, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H27, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H28, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H29, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H29, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H20, [GW000], FW
H00 = \bigcirc<!--</td--><td>DIVIDE
Divides the contents of the source by the quotient in the area specified by the rest
\bigcirc DIV S, D, R
S: Source
D: Destination
R: Result
The settings of the E and V flags change
0.49 ms (when both S and D are word-1
1.35 ms (when either S or D is a long-le
When D = 0, the E flag is turned on and
\bigcirc DIV FW000, FW001, FW002
FW000 0100
FW001 0010
FW002 0010
\downarrow DIV H22, [GW000], FW100
H0022 \bigcirc GW0
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW0
\downarrow DIV H22, [GW000], FW100
H0022 \bigcirc O
\bigcirc O
\bigcirc O
\bigcirc O</br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></td><td>DIVIDE
Divides the contents of the source by those of
quotient in the area specified by the result pro-
\bigcirc DIV S, D, R
S: Source
D: Destination
R: Result
The settings of the E and V flags change.
0.49 ms (when both S and D are word-length
1.35 ms (when either S or D is a long-length
When D = 0, the E flag is turned on and not
\bigcirc DIV FW000, FW001, FW002
FW001 0010
FW001 0010
FW002 0010
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
\bigcirc GW001
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
\bigcirc DIV H22, [GW000], FW100
\bigcirc OI10
\bigcirc OI10</td><td>DIVIDE Divides the contents of the source by those of the desquotient in the area specified by the result parameter. \bigcirc DIV S, D, R S÷D S: Source D: Destination R: Result The settings of the E and V flags change. The other 0.49 ms (when both S and D are word-length paramet 1.35 ms (when either S or D is a long-length paramet When D = 0, the E flag is turned on and nothing is paramet Image: Content of the settings of the E flag is turned on and nothing is paramet \bigcirc DIV FW000, FW001, FW002 FW001 0100 Image: Content of the settings of the E flag is turned on and nothing is paramet \bigcirc DIV FW000, FW001, FW002 FW001 0010 Image: Content of the settings of the E flag is turned on and nothing is parameter \bigcirc DIV FW000, FW001, FW002 FW001 0010 Image: Content of the settings of the E flag is turned on and nothing is parameter \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the E flag is turned on and nothing is parameter. Image: Constant of the settings of the E flag is turned on and nothing is parameter. \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the E flag is turned on and nothing is parameter. Image: Constant of the settings of the E flag is turned on and nothing is parameter. \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the the settings of the settin</td><td>DIVIDE Divides the contents of the source by those of the destination a quotient in the area specified by the result parameter.</td><td>DIVIDE Divides the contents of the source by those of the destination and stores quotient in the area specified by the result parameter. \bigcirc DIV S, D, R S÷D \rightarrow R S: Source D: Destination R: Result The settings of the E and V flags change. The other flags are turned or 0.49 ms (when both S and D are word-length parameters) 1.35 ms (when either S or D is a long-length parameter) When D = 0, the E flag is turned on and nothing is performed. \diamondsuit DIV FW000, FW001, FW002 FW001 0010 FW002 0010 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW000 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H20 \bigcirc DIV H20 \bigcirc GW001 \bigcirc GW001 \bigcirc GW001<</td></td></td> | <td>DIVIDEDivides the contents of the so
quotient in the area specified\bigcirc DIV S, D, RS: SourceD: DestinationR: ResultThe settings of the E and V fl0.49 ms (when both S and D a
1.35 ms (when either S or D iWhen D = 0, the E flag is turn\bigcirc DIV FW000, FW001, FW001\bigcirc DIV FW000, FW001, FW001\bigcirc DIV FW002\bigcirc DIV H22, [GW000], FW
0010\bigcirc DIV H22, [GW000], FW
H0022 — $\bigcirc$$\bigcirc$ DIV H22, [GW000], FW
H002 — $\bigcirc$$\bigcirc$ DIV H22, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H22, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H23, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H24, [GW000], FW
H002 = $\bigcirc$$\bigcirc$ DIV H25, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H26, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H27, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H28, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H29, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H29, [GW000], FW
H00 = $\bigcirc$$\bigcirc$ DIV H20, [GW000], FW
H00 = \bigcirc<!--</td--><td>DIVIDE
Divides the contents of the source by the quotient in the area specified by the rest
\bigcirc DIV S, D, R
S: Source
D: Destination
R: Result
The settings of the E and V flags change
0.49 ms (when both S and D are word-1
1.35 ms (when either S or D is a long-le
When D = 0, the E flag is turned on and
\bigcirc DIV FW000, FW001, FW002
FW000 0100
FW001 0010
FW002 0010
\downarrow DIV H22, [GW000], FW100
H0022 \bigcirc GW0
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW0
\downarrow DIV H22, [GW000], FW100
H0022 \bigcirc O
\bigcirc O
\bigcirc O
\bigcirc O</br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></td><td>DIVIDE
Divides the contents of the source by those of
quotient in the area specified by the result pro-
\bigcirc DIV S, D, R
S: Source
D: Destination
R: Result
The settings of the E and V flags change.
0.49 ms (when both S and D are word-length
1.35 ms (when either S or D is a long-length
When D = 0, the E flag is turned on and not
\bigcirc DIV FW000, FW001, FW002
FW001 0010
FW001 0010
FW002 0010
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
\bigcirc GW001
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
\bigcirc DIV H22, [GW000], FW100
\bigcirc OI10
\bigcirc OI10</td><td>DIVIDE Divides the contents of the source by those of the desquotient in the area specified by the result parameter. \bigcirc DIV S, D, R S÷D S: Source D: Destination R: Result The settings of the E and V flags change. The other 0.49 ms (when both S and D are word-length paramet 1.35 ms (when either S or D is a long-length paramet When D = 0, the E flag is turned on and nothing is paramet Image: Content of the settings of the E flag is turned on and nothing is paramet \bigcirc DIV FW000, FW001, FW002 FW001 0100 Image: Content of the settings of the E flag is turned on and nothing is paramet \bigcirc DIV FW000, FW001, FW002 FW001 0010 Image: Content of the settings of the E flag is turned on and nothing is parameter \bigcirc DIV FW000, FW001, FW002 FW001 0010 Image: Content of the settings of the E flag is turned on and nothing is parameter \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the E flag is turned on and nothing is parameter. Image: Constant of the settings of the E flag is turned on and nothing is parameter. \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the E flag is turned on and nothing is parameter. Image: Constant of the settings of the E flag is turned on and nothing is parameter. \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the the settings of the settin</td><td>DIVIDE Divides the contents of the source by those of the destination a quotient in the area specified by the result parameter.</td><td>DIVIDE Divides the contents of the source by those of the destination and stores quotient in the area specified by the result parameter. \bigcirc DIV S, D, R S÷D \rightarrow R S: Source D: Destination R: Result The settings of the E and V flags change. The other flags are turned or 0.49 ms (when both S and D are word-length parameters) 1.35 ms (when either S or D is a long-length parameter) When D = 0, the E flag is turned on and nothing is performed. \diamondsuit DIV FW000, FW001, FW002 FW001 0010 FW002 0010 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW000 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H20 \bigcirc DIV H20 \bigcirc GW001 \bigcirc GW001 \bigcirc GW001<</td></td> | DIVIDEDivides the contents of the so
quotient in the area specified \bigcirc DIV S, D, RS: SourceD: DestinationR: ResultThe settings of the E and V fl0.49 ms (when both S and D a
1.35 ms (when either S or D iWhen D = 0, the E flag is turn \bigcirc DIV FW000, FW001, FW001 \bigcirc DIV FW000, FW001, FW001 \bigcirc DIV FW002 \bigcirc DIV H22, [GW000], FW
0010 \bigcirc DIV H22, [GW000], FW
H0022 — \bigcirc \bigcirc DIV H22, [GW000], FW
H002 — \bigcirc \bigcirc DIV H22, [GW000], FW
H00 = \bigcirc \bigcirc DIV H22, [GW000], FW
H00 = \bigcirc \bigcirc DIV H23, [GW000], FW
H00 = \bigcirc \bigcirc DIV H24, [GW000], FW
H002 = \bigcirc \bigcirc DIV H25, [GW000], FW
H00 = \bigcirc \bigcirc DIV H26, [GW000], FW
H00 = \bigcirc \bigcirc DIV H27, [GW000], FW
H00 = \bigcirc \bigcirc DIV H28, [GW000], FW
H00 = \bigcirc \bigcirc DIV H29, [GW000], FW
H00 = \bigcirc \bigcirc DIV H29, [GW000], FW
H00 = \bigcirc \bigcirc DIV H20, [GW000], FW
H00 = \bigcirc </td <td>DIVIDE
Divides the contents of the source by the quotient in the area specified by the rest
\bigcirc DIV S, D, R
S: Source
D: Destination
R: Result
The settings of the E and V flags change
0.49 ms (when both S and D are word-1
1.35 ms (when either S or D is a long-le
When D = 0, the E flag is turned on and
\bigcirc DIV FW000, FW001, FW002
FW000 0100
FW001 0010
FW002 0010
\downarrow DIV H22, [GW000], FW100
H0022 \bigcirc GW0
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW0
\downarrow DIV H22, [GW000], FW100
H0022 \bigcirc O
\bigcirc O
\bigcirc O
\bigcirc O</br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></br></td> <td>DIVIDE
Divides the contents of the source by those of
quotient in the area specified by the result pro-
\bigcirc DIV S, D, R
S: Source
D: Destination
R: Result
The settings of the E and V flags change.
0.49 ms (when both S and D are word-length
1.35 ms (when either S or D is a long-length
When D = 0, the E flag is turned on and not
\bigcirc DIV FW000, FW001, FW002
FW001 0010
FW001 0010
FW002 0010
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
\bigcirc GW001
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
\bigcirc DIV H22, [GW000], FW100
\bigcirc OI10
\bigcirc OI10</td> <td>DIVIDE Divides the contents of the source by those of the desquotient in the area specified by the result parameter. \bigcirc DIV S, D, R S÷D S: Source D: Destination R: Result The settings of the E and V flags change. The other 0.49 ms (when both S and D are word-length paramet 1.35 ms (when either S or D is a long-length paramet When D = 0, the E flag is turned on and nothing is paramet Image: Content of the settings of the E flag is turned on and nothing is paramet \bigcirc DIV FW000, FW001, FW002 FW001 0100 Image: Content of the settings of the E flag is turned on and nothing is paramet \bigcirc DIV FW000, FW001, FW002 FW001 0010 Image: Content of the settings of the E flag is turned on and nothing is parameter \bigcirc DIV FW000, FW001, FW002 FW001 0010 Image: Content of the settings of the E flag is turned on and nothing is parameter \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the E flag is turned on and nothing is parameter. Image: Constant of the settings of the E flag is turned on and nothing is parameter. \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the E flag is turned on and nothing is parameter. Image: Constant of the settings of the E flag is turned on and nothing is parameter. \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the the settings of the settin</td> <td>DIVIDE Divides the contents of the source by those of the destination a quotient in the area specified by the result parameter.</td> <td>DIVIDE Divides the contents of the source by those of the destination and stores quotient in the area specified by the result parameter. \bigcirc DIV S, D, R S÷D \rightarrow R S: Source D: Destination R: Result The settings of the E and V flags change. The other flags are turned or 0.49 ms (when both S and D are word-length parameters) 1.35 ms (when either S or D is a long-length parameter) When D = 0, the E flag is turned on and nothing is performed. \diamondsuit DIV FW000, FW001, FW002 FW001 0010 FW002 0010 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW000 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H20 \bigcirc DIV H20 \bigcirc GW001 \bigcirc GW001 \bigcirc GW001<</td> | DIVIDE
Divides the contents of the source by the quotient in the area specified by the rest
\bigcirc DIV S, D, R
S: Source
D: Destination
R: Result
The settings of the E and V flags change
0.49 ms (when both S and D are word-1
1.35 ms (when either S or D is a long-le
When D = 0, the E flag is turned on and
\bigcirc DIV FW000, FW001, FW002
FW000 0100
FW001 0010
FW002 0010
\downarrow DIV H22, [GW000], FW100
H0022 \bigcirc GW0
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW0
\downarrow DIV H22, [GW000], FW100
H0022 \bigcirc O
\bigcirc O
 | DIVIDE
Divides the contents of the source by those of
quotient in the area specified by the result pro-
\bigcirc DIV S, D, R
S: Source
D: Destination
R: Result
The settings of the E and V flags change.
0.49 ms (when both S and D are word-length
1.35 ms (when either S or D is a long-length
When D = 0, the E flag is turned on and not
\bigcirc DIV FW000, FW001, FW002
FW001 0010
FW001 0010
FW002 0010
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
FW100 0002
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
\bigcirc GW001
\bigcirc DIV H22, [GW000], FW100
H0022 \bigcirc GW000
\bigcirc GW001
\bigcirc DIV H22, [GW000], FW100
\bigcirc OI10
\bigcirc OI10 | DIVIDE Divides the contents of the source by those of the desquotient in the area specified by the result parameter. \bigcirc DIV S, D, R S÷D S: Source D: Destination R: Result The settings of the E and V flags change. The other 0.49 ms (when both S and D are word-length paramet 1.35 ms (when either S or D is a long-length paramet When D = 0, the E flag is turned on and nothing is paramet Image: Content of the settings of the E flag is turned on and nothing is paramet \bigcirc DIV FW000, FW001, FW002 FW001 0100 Image: Content of the settings of the E flag is turned on and nothing is paramet \bigcirc DIV FW000, FW001, FW002 FW001 0010 Image: Content of the settings of the E flag is turned on and nothing is parameter \bigcirc DIV FW000, FW001, FW002 FW001 0010 Image: Content of the settings of the E flag is turned on and nothing is parameter \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the E flag is turned on and nothing is parameter. Image: Constant of the settings of the E flag is turned on and nothing is parameter. \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the E flag is turned on and nothing is parameter. Image: Constant of the settings of the E flag is turned on and nothing is parameter. \bigcirc DIV H22, [GW000], FW100 Image: Constant of the settings of the the settings of the settin | DIVIDE Divides the contents of the source by those of the destination a quotient in the area specified by the result parameter. | DIVIDE Divides the contents of the source by those of the destination and stores quotient in the area specified by the result parameter. \bigcirc DIV S, D, R S÷D \rightarrow R S: Source D: Destination R: Result The settings of the E and V flags change. The other flags are turned or 0.49 ms (when both S and D are word-length parameters) 1.35 ms (when either S or D is a long-length parameter) When D = 0, the E flag is turned on and nothing is performed. \diamondsuit DIV FW000, FW001, FW002 FW001 0010 FW002 0010 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW000 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW001 \bigcirc DIV H20 \bigcirc DIV H20 \bigcirc GW001 \bigcirc GW001 \bigcirc GW001< | | |

Divides the remainder	e contents	of the so	1 .1	~						
Divides the contents of the source by those of the destination and stores the remainder in the area specified by the result parameter.										
ф моd	S, D, R				S÷D	$\rightarrow R$				
S: Source D: Destina R: Result	S: Source D: Destination R: Result									
The settings of the E and V flags change. The other flags are turned off.										
0.50 ms (w 1.40 ms (w	0.50 ms (when both S and D are word-length parameters) 1.40 ms (when either S or D is a long-length parameter)									
When D = When an o	When $D = 0$, the E flag is turned on and nothing is performed. When an overflow occurs, R is set to 0.									
 MOD FW000, FW001, FW002 FW000 0100 FW001 0012 FW002 0004 GW000 0000 GW001 0012 FW100 0010 										
S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant		
Direct word-length	\bigcirc	\bigcirc	0	Direc word	t -length	\bigcirc	\bigcirc	×		
Direct long-length	\bigcirc	0	0	Directions-	rt length	\bigcirc	0	×		
Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indire word	ect -length	×	\bigtriangleup	\bigtriangleup		
Indirect long-length	×	\bigtriangleup	\bigtriangleup	$\begin{array}{c c} \text{Indirect} \\ \text{long-length} \end{array} \times \qquad \bigtriangleup$						
	MOD S: Source D: Destina R: Result The setting 0.50 ms (w 1.40 ms (w When D = When an o MOD MOD OIV H H F S, D Direct word-length Direct long-length Indirect word-length Indirect long-length Indirect	\bigcirc MOD S, D, RS: SourceD: DestinationR: ResultThe settings of the E0.50 ms (when both1.40 ms (when eitheWhen D = 0, the E fWhen an overflow of \diamondsuit MOD FW000,FW000FW001FW002 \diamondsuit DIV H22, [GWH0022 —FW100S, DBit-typePI/ODirect\overlagstyle DirectNod IndirectWord-lengthNod IndirectVord-lengthIndirectXIndirectXIndirectXIndirectXIndirectXIndirectXNog-length	\bigcirc MOD S, D, RS: SourceD: DestinationR: ResultThe settings of the E and V flat0.50 ms (when both S and D at1.40 ms (when either S or D itWhen D = 0, the E flag is turnWhen an overflow occurs, R it \diamondsuit MOD FW000, FW001,FW0000100FW0010012FW0020004 \diamondsuit DIV H22, [GW000], FVH0022 \bigcirc FW1000010S, DBit-typePI/ODirectword-lengthDirectIndirect× \triangle Indirect× \triangle	\(\mathbf{P}\) MOD S, D, R S: Source D: Destination R: Result The settings of the E and V flags change 0.50 ms (when both S and D are word-I 1.40 ms (when either S or D is a long-le When D = 0, the E flag is turned on and When an overflow occurs, R is set to 0. MOD FW000, FW001, FW002 FW000 DIV H22, [GW000], FW100 H0022 Could DIV H22, [GW000], FW100 DIV H22, [GW000], FW100 H0022 Could S, D Bit-type Direct Direct Direct Direct Direct Direct Ding-length <td>\bigcirc MOD S, D, R S: Source D: Destination R: Result The settings of the E and V flags change. The settings of the E and D are word-length p 1.40 ms (when both S and D are word-length p 1.40 ms (when either S or D is a long-length pa When D = 0, the E flag is turned on and nothin When an overflow occurs, R is set to 0. \diamondsuit MOD FW000, FW001, FW002 FW000 0100 FW001 0012 0004 0012 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW000 GW001 GW001 FW100 0010 S, D Bit-type Word-type Onto FW100 O100 S, D Bit-type Constant Direct \bigcirc \bigcirc Direct \bigcirc \bigcirc Direct \bigcirc \bigcirc Indirect \land $△$ Indirect \land $△$</td> <td>Second Second Secon</td> <td></td> <td></td>	\bigcirc MOD S, D, R S: Source D: Destination R: Result The settings of the E and V flags change. The settings of the E and D are word-length p 1.40 ms (when both S and D are word-length p 1.40 ms (when either S or D is a long-length pa When D = 0, the E flag is turned on and nothin When an overflow occurs, R is set to 0. \diamondsuit MOD FW000, FW001, FW002 FW000 0100 FW001 0012 0004 0012 \bigcirc DIV H22, [GW000], FW100 H0022 \bigcirc GW000 GW001 GW001 FW100 0010 S, D Bit-type Word-type Onto FW100 O100 S, D Bit-type Constant Direct \bigcirc \bigcirc Direct \bigcirc \bigcirc Direct \bigcirc \bigcirc Indirect \land $△$ Indirect \land $△$	Second Secon				

SCL	SCALE CHANGE										
Function	Converts the result is	he scale o n the area	of the sour specified	ce with th l by the re	e con sult p	tents of aramete	the destin r.	nation and	stores		
Parameters and	♦ SCL	S, D1, D	2, R			S×D	01÷D2 →	· R			
processing	S: Source D1: Destin D2: Destin R: Result	S: Source D1: Destination 1 D2: Destination 2 R: Result									
Flags	The settings of the E and V flags change. The other flags are turned off.										
Processing time	1.38 ms	1.38 ms									
Remarks	When a muspecified b the E flag i is set to 0.	When a multiplication overflow occurs, the overflow value is written in the area specified by the result parameter and processing is terminated. When D2=0, the E flag is turned on and nothing is performed. When an overflow occurs, R is set to 0.									
Examples	<pre> SCL SCL</pre>	♦ SCL FW000, FW001, FW002, FW003 FW000 3320 FW001 0010 FW002 0066 FW003 0805 (3320×/10÷/66) ♦ SCL GW000, GW001, H1110, FW100 GW000 2222 GW001 0012 H1110 FW100 0024									
Valid					_						
parameters	S, D1, D2	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant		
\triangle : A parameter	Direct word-length	0	0	0	Dii wo	rect rd-length	0	0	×		
when an odd-	Direct long-length	0	0	0	Dir lor	rect Ig-length	0	0	×		
numbered address is used.	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Inc wo	lirect rd-length	×	\bigtriangleup	\bigtriangleup		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Inc	lirect ig-length	×	\bigtriangleup	\bigtriangleup		

AND	AND											
Function	ANDs the	contents	of the sou	rce with t	hose of	the de	stination	and stores	s the			
	logical pro	duct in th	e area spe	ecified by	the resu	ılt par	ameter.					
Parameters and	\diamondsuit and	S, D, R				S & a	$\& D \rightarrow I$	R				
processing	S: Source											
	D: Destina	tion										
	R: Result	R: Result										
Flags	The setting of the E flag changes. The other flags are turned off.											
Processing time	0.36 ms											
Remarks	When R is	a word-le	ength para	ameter, th	e low-or	rder w	ord of the	e operatio	n result			
	is written.											
Examples	♦ AND FW000, FW001, FW002											
	FW000 0001											
	FW001 00FF											
	FW002 0001 ← &&											
	<u> </u>											
	\bigcirc and	H1234,	[GW000]	, FW100	_							
		H1234 -	- &&	\rightarrow	GW00	0	0010					
				<u>ר</u>	GWUU)1	0011					
		FVV100	0010									
Valid												
narameters		Bit-type	Word-type			_	Bit-type	Word-type				
purumeters	S, D	PI/O	PI/O	Constant		R	PI/O	PI/O	Constant			
\triangle : A parameter	Direct word-length	\bigcirc	\bigcirc	0	Direc word-	t -length	\bigcirc	0	×			
error is detected when an odd-	Direct long-length	0	0	0	Direc long-	t length	\bigcirc	0	×			
numbered	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indire word-	ect -length	×	\bigtriangleup	\bigtriangleup			
uuurooo 15 uoou.	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indire long-	ect length	×	\bigtriangleup	\bigtriangleup			

OR	OR										
Function	ORs the co	ontents of	the sourc	e with the	ose	of the dest	ination ar	nd stores t	he		
	logical sun	n in the a	rea specifi	ied by the	res	ult parame	eter.				
Parameters and	\diamondsuit or s	5, D, R				S]	$D \rightarrow R$				
processing	S: Source										
	D: Destina	tion									
	R: Result	R: Result									
Flags	The setting	The setting of the E flag changes. The other flags are turned off.									
Processing time	0.36 ms										
Remarks	When R is is written.	When R is a word-length parameter, the low-order word of the operation result is written.									
Examples		OR FW000 FW001 FW002									
	F	FW000 4321									
	FW001 1234										
	FW002 5335 (1)										
	Т										
	\bigcirc or H	11234, [0	GW000],	FW100		r					
	H	1234 —	$-\Box$		G۷	V000	0010				
					G۷	V001 [0011				
	F	W100	1235								
Valid											
parameters	S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant		
\triangle : A parameter	Direct word-length	0	0	0		Direct word-length	\bigcirc	0	×		
when an odd-	Direct long-length	0	0	0		Direct long-length	\bigcirc	0	×		
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup		Indirect word-length	×	\bigtriangleup	\bigtriangleup		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup		Indirect long-length	×	\bigtriangleup	\bigtriangleup		

EOR	EXCLUSIVE OR										
Function	EORs the clogical sun	contents on in the ar	of the sour rea specifi	rce with the	nose of result	the des	stination a	and stores	s the		
Parameters and	0 EOR	S, D, R				S ^ /	$D \rightarrow R$	Ł			
processing	S: Source										
	D: Destina	tion									
	R: Result	R: Result									
Flags	The setting of the E flag changes. The other flags are turned off.										
Processing time	0.36 ms										
Remarks	When R is is written.	a word-le	ength para	umeter, th	e low-c	order w	ord of the	e operatio	n result		
Examples	0 EOR	O EOR FW000, FW001, FW002									
		FW000 4321									
	FW001 1234										
	FW002 5115 <										
	4										
	\bigcirc EOR	H1234,	[GW000]	, FW100							
		H1234 -		\rightarrow	- GW0	00	0010				
				- L	- GW0	01	0011				
		FW100	1225								
Valid											
parameters	S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant		
$\wedge \cdot \mathbf{\Lambda}$ parameter	Direct	\bigcirc	\bigcirc	\bigcirc	Dire	ect	\bigcirc	\bigcirc	×		
$rac{\Delta}$. A parameter error is detected	word-length	\bigcirc		\bigcirc	word	d-length	\bigcirc	\bigcirc	~		
when an odd-	long-length	\bigcirc	\bigcirc	\bigcirc	long	ct length	\bigcirc	\bigcirc	\times		
numbered	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indi wore	rect d-length	×	\bigtriangleup	\bigtriangleup		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	India long	\bigtriangleup					

NOT	NOT (Negative)										
Function	Inverts the by the resu	contents lt parame	(bits) of t eter.	he source	and sto	ores the	e result in	the area	specified		
Parameters and	\diamondsuit NOT	S, R				S (bi	it inversio	$(nn) \rightarrow R$			
processing	S: Source R: Result	S: Source R: Result									
Flags	The setting	g of the E	flag chan	ges. Th	e other	flags a	re turned	off.			
Processing time	0.36 ms										
Remarks											
Examples	О NOT	 ♦ NOT FW000, FW002 FW000 4321 FW002 BCDE NOT NOT [GW000], FW100 GW000 0010 GW001 0011 FW100 FFEE 									
Valid											
parameters	S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant		
\triangle : A parameter	Direct word-length	0	0	0	Direc word	et -length	\bigcirc	0	×		
when an odd-	Direct long-length	\bigcirc	0	0	Direction Direct	et length	\bigcirc	0	\times		
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indir word	ect -length	×	\bigtriangleup	\bigtriangleup		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indir long-	ect ·length	×	\bigtriangleup	\bigtriangleup		

EQUAL										
Compares the area sports of in the stores 0 in the store 0 in the st	Compares the contents of the source with those of the destination. Stores 1 in the area specified by the result parameter when a match is found. Otherwise, stores 0 in the area.									
C EQU S: Source D: Destina R: Result	$ \begin{array}{c} \abovedisplayskiplimits{ \belowdisplayskiplimits{ \belowdisp$									
The setting of the E flag changes. The other flags are turned off.										
0.40 ms	0.40 ms									
Word-leng	th data is	sign-exte	nded to lo	ong-leng	th dat	a before i	it is comp	ared.		
$\begin{array}{c c} & EQU FW000, FW001, FW002 \\ & & FW000 & 4321 \\ & & FW001 & 1234 \\ & & FW002 & 0000 \\ & & & Comparison \end{array}$ $\begin{array}{c c} & EQU HF234, [GW000], FW100 \\ & & HF234 & Comparison & GW000 & 0000 \\ & & & GW001 & F234 \\ & & FW100 & 0000 \\ & & & HF234 \text{ is } HFFFF234. \end{array}$										
S, D	Bit-type PI/O	Word-type PI/O	Constant	F	२	Bit-type PI/O	Word-type PI/O	Constant		
Direct word-length	\bigcirc	0	0	Direct word-	length	\bigcirc	0	×		
Direct long-length	0	0	0	Direct long-l	ength	\bigcirc	0	×		
Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indire word-	ct length	×	\bigtriangleup	\bigtriangleup		
Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indire long-l	ct ength	×	\bigtriangleup	\bigtriangleup		
	EQUAL Compares to the area sponding stores 0 in a Compares to the area sponding EQU S: Source D: Destinat R: Result The setting 0.40 ms Word-length O.40 ms EQU S, D EQU S, D Direct word-length Direct long-length Indirect word-length Indirect long-length	EQUAL Compares the content the area specified by stores 0 in the area. C EQU S, D, R S: Source D: Destination R: Result The setting of the E 0.40 ms Word-length data is C EQU FW000, FW001 FW001 FW001 FW002 C EQU HF234, HF234 - FW100 S, D Bit-type PI/O Direct word-length C C Direct word-length C C Direct word-length C Direct word-length C Direct word-length C Direct N C C C C C C C C C C C C C	EQUALCompares the contents of the the area specified by the result stores 0 in the area. \bigcirc EQU S, D, RS: Source D: Destination R: ResultThe setting of the E flag chan 0.40 ms0.40 msWord-length data is sign-exter FW000 \bigcirc EQU FW000, FW001, FW001 $PW000$ 4321 FW002 \bigcirc EQU FW000, Comparison FW001 \bigcirc EQU HF234, [GW000] HF234 \bigcirc EQU HF234, [GW000] HF234 \bigcirc EQU HF234, [GW000] HF234 \bigcirc Direct long-length \bigcirc \bigcirc \bigcirc Direct long-length \bigcirc \bigcirc \bigcirc Indirect long-length \bigcirc \triangle \square <td>EQUAL Compares the contents of the source with the area specified by the result parameters stores 0 in the area.</td> <td>EQUAL Compares the contents of the source with those the area specified by the result parameter when stores 0 in the area.</td> <td>EQUALCompares the contents of the source with those of the the area specified by the result parameter when a mat stores 0 in the area.\diamondsuit EQU S, D, RS = 1 S \neq JS = 1 S \neq JD: Destination R: ResultThe setting of the E flag changes. The other flags a 0.40 msWord-length data is sign-extended to long-length dat\diamondsuit EQU FW000, FW001, FW002 FW001 FW001 EQU FW002 FW000 FW001 FW002 O000 OGW00 GW00 GW00 Comparison GW00 HF234, [GW000], FW100 HF234 is\clubsuit S, DBit-type PI/O PI/OConstant PI/O O O OR Direct long-length\circlearrowright S, DBit-type PI/O PI/OO O O O OR Direct long-length\square Direct word-lengthO O O OO O O Indirect Indirect X A AA A Indirect Indirect Indirect X A A</td> <td>EQUAL Compares the contents of the source with those of the destinat the area specified by the result parameter when a match is four stores 0 in the area. \bigcirc EQU S, D, R $S = D 1 \rightarrow S \neq D 0 \rightarrow S \rightarrow$</td> <td>EQUAL Compares the contents of the source with those of the destination. Stot the area specified by the result parameter when a match is found. Other stores 0 in the area. \diamondsuit EQU S, D, R S = D 1 \rightarrow R S: Source S \neq D 0 \rightarrow R D: Destination R: Result The setting of the E flag changes. The other flags are turned off. 0.40 ms Word-length data is sign-extended to long-length data before it is comp \diamondsuit EQU FW000, FW001, FW002 FW001 1234 FW002 0000 O000 Comparison \diamondsuit EQU HF234, [GW000], FW100 HF234 Comparison \bigcirc EQU HF234, [GW000], FW100 HF234 Gw001 FW100 0000 HF234 is HFFFF234.</td>	EQUAL Compares the contents of the source with the area specified by the result parameters stores 0 in the area.	EQUAL Compares the contents of the source with those the area specified by the result parameter when stores 0 in the area.	EQUALCompares the contents of the source with those of the the area specified by the result parameter when a mat stores 0 in the area. \diamondsuit EQU S, D, RS = 1 S \neq JS = 1 S \neq JD: Destination R: ResultThe setting of the E flag changes. The other flags a 0.40 msWord-length data is sign-extended to long-length dat \diamondsuit EQU FW000, FW001, FW002 FW001 FW001 EQU FW002 FW000 FW001 FW002 O000 OGW00 GW00 GW00 Comparison GW00 HF234, [GW000], FW100 HF234 is \clubsuit S, DBit-type PI/O PI/OConstant PI/O O O OR Direct long-length \circlearrowright S, DBit-type PI/O PI/OO O O O OR Direct long-length \square Direct word-lengthO O O OO O O Indirect Indirect X A AA A Indirect Indirect Indirect X A A	EQUAL Compares the contents of the source with those of the destinat the area specified by the result parameter when a match is four stores 0 in the area. \bigcirc EQU S, D, R $S = D 1 \rightarrow S \neq D 0 \rightarrow S \rightarrow$	EQUAL Compares the contents of the source with those of the destination. Stot the area specified by the result parameter when a match is found. Other stores 0 in the area. \diamondsuit EQU S, D, R S = D 1 \rightarrow R S: Source S \neq D 0 \rightarrow R D: Destination R: Result The setting of the E flag changes. The other flags are turned off. 0.40 ms Word-length data is sign-extended to long-length data before it is comp \diamondsuit EQU FW000, FW001, FW002 FW001 1234 FW002 0000 O000 Comparison \diamondsuit EQU HF234, [GW000], FW100 HF234 Comparison \bigcirc EQU HF234, [GW000], FW100 HF234 Gw001 FW100 0000 HF234 is HFFFF234.		

NEQ	NOT EQUAL										
Function	Compares the contents of the source with those of the destination. Stores 1 in the area specified by the result parameter when a match is not found. Otherwise, stores 0 in the area.										
Parameters and processing	 NEQ S: Source D: Destinat R: Result 										
Flags	The setting	The setting of the E flag changes. The other flags are turned off.									
Processing time	0.40 ms										
Remarks	Word-leng	th data is	sign-exte	nded to lo	ong-leng	th dat	a before i	it is comp	ared.		
Examples	NEQ FW000, FW001, FW002FW000 4321 FW001 1234 FW002 0001 ComparisonNEQ HF234, [GW000], FW100HF234ComparisonGW000 0000 FW100 0001 HF234 is HFFFFF234.										
Valid parameters	S D	Bit-type	Word-type	Constant		5	Bit-type	Word-type	Constant		
\triangle : A parameter error is detected when an odd-	Direct word-length Direct long-length	PI/O	PI/O		Direct word- Direct long-l	t length t length	PI/O	PI/O			
numbered	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indire word-	ect length	×	\bigtriangleup	\bigtriangleup		
auuress 15 useu.	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indire long-l	ect length	×	\bigtriangleup	\bigtriangleup		

GT	GREATER THAN										
Function	Compares the area sp destination	the conte ecified by . Other	nts of the 7 the resul wise, store	source wi t paramet es 0 in the	ith those er when e area.	of the so	e destinat ource is g	ion. Sto reater tha	res 1 in n the		
Parameters and processing	GT S S: Source D: Destina R: Result	, D, R tion				S >] $S \leq]$	$\begin{array}{ccc} D & 1 \rightarrow \\ D & 0 \rightarrow \end{array}$	R R			
Flags	The setting	The setting of the E flag changes. The other flags are turned off.									
Processing time	0.40 ms	0.40 ms									
Remarks	Word-leng	th data is	sign-exte	nded to lo	ong-leng	th dat	a before i	it is comp	ared.		
Examples	♀ GTF F F ♀ GTH H	 									
Valid parameters	S, D	Bit-type	Word-type	Constant	F	٦	Bit-type	Word-type	Constant		
\triangle : A parameter	Direct word-length			0	Direct word-	t ·length			×		
error is detected when an odd-	Direct long-length	0	0	0	Direct long-l	t ength	\bigcirc	0	×		
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indire word-	ect length	×	\bigtriangleup	\bigtriangleup		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indire long-l	ength	×	\bigtriangleup	\bigtriangleup		

GE	GREATER	GREATER OR EQUAL									
Function	Compares the area spe equal to the	Compares the contents of the source with those of the destination. Stores 1 in the area specified by the result parameter when the source is greater than or equal to the destination. Otherwise, stores 0 in the area.									
Parameters and processing	GE S S: Source D: Destina R: Result	, D, R tion				S≧] S<]	$\begin{array}{ccc} D & 1 \rightarrow \\ D & 0 \rightarrow \end{array}$	R R			
Flags	The setting	The setting of the E flag changes. The other flags are turned off.									
Processing time	0.40 ms										
Remarks	Word-leng	th data is	sign-exte	nded to lo	ong-leng	th dat	a before i	it is comp	ared.		
Examples	♀ GE F F F ♀ GE H H	 Ģ GE FW000, FW001, FW002 FW000 4321 FW001 1234 Guparison GE H1234, [GW000], FW100 H1234 Comparison GW000 0000 H1234 FW100 0000 									
Valid parameters	S, D	Bit-type PI/O	Word-type PI/O	Constant	F	२	Bit-type PI/O	Word-type PI/O	Constant		
\triangle : A parameter	Direct word-length	0	\bigcirc	0	Direct word-	t length	0	0	×		
error 1s detected when an odd-	Direct long-length	0	0	0	Direct long-l	t ength	0	0	×		
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indire word-	ct length	×	\bigtriangleup	\bigtriangleup		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indire long-l	ct ength	×	\bigtriangleup	\bigtriangleup		

LT	LESS TH	۹N								
Function	Compares the area sp destination	the conte ecified by . Other	nts of the 7 the resul wise, store	source with the source with th	ith those er when e area.	of the so	e destinat ource is s	ion. Sto maller tha	ores 1 in an the	
Parameters and processing	C LT S, S: Source D: Destina R: Result	D, R tion				S <] S ≧]	$\begin{array}{ccc} D & 1 \rightarrow \\ D & 0 \rightarrow \end{array}$	R R		
Flags	The setting	The setting of the E flag changes. The other flags are turned off.								
Processing time	0.40 ms									
Remarks	Word-leng	th data is	sign-exte	nded to lo	ong-leng	th dat	a before i	it is comp	ared.	
Examples	♀ LT F\ F\ F\ ♀ LT H' H' F\	 LT FW000, FW001, FW002 FW000 4321 FW001 1234 FW002 0000 Comparison LT H1234, [GW000], FW100 H1234 Comparison GW000 0000 F234 FW100 0001 								
Valid parameters	S, D	Bit-type PI/O	Word-type PI/O	Constant		R	Bit-type PI/O	Word-type PI/O	Constant	
\triangle : A parameter	Direct word-length	0	0	0	Direc word-	t ·length	\bigcirc	0	×	
error is detected when an odd-	Direct long-length	0	0	0	Direct long-l	t length	\bigcirc	0	×	
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indire word-	ect length	×	\bigtriangleup	\bigtriangleup	
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indire long-l	ect length	×	\bigtriangleup	\bigtriangleup	

LE	LESS OR	LESS OR EQUAL									
Function	Compares the area speed equal to the	Compares the contents of the source with those of the destination. Stores 1 in the area specified by the result parameter when the source is smaller than or equal to the destination. Otherwise, stores 0 in the area.									
Parameters and processing	C LE S, S: Source D: Destina R: Result	D, R tion				$S \leq D$ S > D	$\begin{array}{ccc} 1 & \rightarrow \\ 0 & 0 & \rightarrow \end{array}$	R R			
Flags	The setting	of the E	flag chan	ges. Th	e other fla	ags ar	e turned	off.			
Processing time	0.40 ms	0.40 ms									
Remarks	Word-leng	th data is	sign-exte	nded to lo	ong-lengtl	h data	ı before i	it is comp	ared.		
Examples	↓ LE F\ F\ F\ ↓ ↓ LE H ¹ H	 LE FW000, FW001, FW002 FW000 4321 FW001 1234 GW002 0000 Comparison LE H1234, [GW000], FW100 H1234 Comparison GW000 0000 GW001 F234 FW100 0001 									
Valid parameters	S, D	Bit-type PI/O	Word-type PI/O	Constant	R		Bit-type PI/O	Word-type PI/O	Constant		
\triangle : A parameter	Direct word-length	0	0	\bigcirc	Direct word-le	ength	0	0	×		
error is detected when an odd-	Direct long-length	0	0	0	Direct long-ler	ngth	0	0	×		
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indirect word-le	t ength	×	\bigtriangleup	\bigtriangleup		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-let	t ngth	×	\bigtriangleup	\bigtriangleup		

TST	TEST	TEST									
Function	Tests the co	ontents o	f the source	ce and set	the P, Z	Z, and N flags.					
Parameters and processing						S > 0 : P = 1, Z = 0, N = 0 S = 0 : P = 0, Z = 1, N = 0 S < 0 : P = 0, Z = 0, N = 1					
Flags	The setting	s of the I	E, P, Z, an	d N flags	change.	. The other flags are turned off.					
Processing time	0.17 ms										
Remarks	Word-leng	Word-length data is sign-extended to long-length data before it is tested.									
Examples	() ТST 	=W000 =W000 =W020 GW000 GW000 GW001 GW020	4321 2000 FFFF F234 1000		(]	Test					
Valid					1						
parameters	S	Bit-type PI/O	Word-type PI/O	Constant							
\triangle : A parameter	Direct word-length	\bigcirc	0	\bigcirc							
when an odd-	Direct long-length	\bigcirc	0	0							
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup							
	Indirect long-length	×	\bigtriangleup	\bigtriangleup							

MOV	MOVE	MOVE										
Function	Moves the	contents	of the sou	rce to the	destina	tion.						
Parameters and	↓ мо∨	S, D				S –	→ D					
processing	S: Source											
	D: Destina	tion										
Flags	The setting	, of the E	flag chan	ges. The	e other	flags a	re turned	off.				
Processing time	0.28 ms	0.28 ms										
Remarks	When the s	When the size of data to be moved does not conform, its type is converted.										
Examples	Ф мо	O MOV FW000, FW002										
	I	FW000 4321										
		FW002 4321 -										
	4											
	↓ MOV HF234, @ [H180000]											
		HF234 -			→ H1	80000) FFFF	-				
						<u>م</u> من 2014						
					HF	234 18	6 HFFFF	FZ34.				
Valid												
parameters	S	Bit-type PI/O	Word-type PI/O	Constant		D	Bit-type PI/O	Word-type PI/O	Constant			
\triangle : A parameter	Direct word-length	\bigcirc	0	\bigcirc	Direc word	ct -length	\bigcirc	0	×			
error is detected when an odd-	Direct long-length	\bigcirc	0	\bigcirc	Direct long-	ct ·length	\bigcirc	0	×			
numbered	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indir word	ect -length	×	\bigtriangleup	\bigtriangleup			
audi 055 15 0500.	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indir long-	ect ·length	×	\bigtriangleup	\bigtriangleup			

МОМ	MOVE MU	JLTI								
Function	Moves the data items	contents at one tin	of the sou ne.	rce to the	de	stination n	word-ler	ngth or lor	ng-length	
Parameters and processing	$ \begin{array}{c} \bigcirc \text{ MOM S, n, D} \\ \text{S: Source} \\ \text{D: Destination} \\ \text{n: Number of word-length or long-length data items trans} \end{array} $							ed at one t	time	
Flags	The setting	The settings of the E and V flags change. The other flags are turned off.								
Processing time	0.39+0.02>	<n ms<="" td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td></n>								
Remarks	When n is When S is set. When	When n is less than or equal to 0 or greater than 256, nothing is processed. When S is a constant, its value is converted to the type of D before the value is set. When S and D have different types, type conversion is performed.								
Examples	МОМ	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$								
Valid		Rit-type	Word-type				Rit-type	Word-type		
parameters	S	PI/O	PI/O	Constant		D	PI/O	PI/O	Constant	
\triangle : A parameter	Direct word-length	0	0	0		Direct word-length	\bigcirc	0	×	
when an odd-	Direct long-length	0	0	0		Direct long-length	\bigcirc	0	×	
numbered address is used.	Indirect word-length	×	\bigtriangleup	\bigtriangleup		Indirect word-length	×	\bigtriangleup	\bigtriangleup	
	Indirect long-length	×	\bigtriangleup	\bigtriangleup		Indirect long-length	×	\bigtriangleup	\bigtriangleup	

EXC	EXCHANGE									
Function	Exchanges the contents of the source and those of the destination with each other.									
Parameters and	$\diamondsuit EXC S, D \qquad \qquad S \longleftrightarrow D$									
processing	S: Source									
	D: Destination									
Flags	The setting of the E flag changes. The other flags are turned off.									
Processing time	0.31 ms									
Remarks	When the size of data to be moved does not conform, its type is converted.									
Examples										
	FW000 1234									
	FW002 4321									
	↓ EXC @ H170000, @ [H180000]									
	$H170000 [F234] \longleftrightarrow H180000 0010 \\ 2 0001 00001 00001 00001 0001 $									
	After exchange									
	H170000 7FFF H180000 FFFF									
	2 F234									
Valid										
parameters	S, D Bit-type Word-type PI/O PI/O Constant									
\triangle : A parameter										
error is detected	Direct									
when an odd-	long-length X									
numbered address is used	$\frac{\text{Indirect}}{\text{word-length}} \times \bigtriangleup$									
uuuroos 15 uoou.	$\begin{array}{c c} \text{Indirect} \\ \text{long-length} \end{array} \times \bigtriangleup \bigtriangleup$									

PSH	FIFO PUS	SH								
Function	Pushes the only word-	contents length da	of the sou ta.	irce into tl	ne FIFO table	e. The F	IFO table	contains		
Parameters and processing	 PSH S: Source TB: First a 	S, TB e ddress of	the FIFO	table	Pointer	Data 1 Data 2	FIFC n (da ZEF add	D table ta size) RO flag_ ress		
Flags	The setting The other f	of the E lags are t	flag chan urned off	ges.			FUL add	L flag _ ress		
Processing time	0.31 ms					Push data		ata 1		
Remarks	When n is greater that When the v than 0 or th value of the	less than n 256, no value of the ne data siz e pointer,	or equal to thing is pro- he pointer ze is less to noting is		Data n Data storage area specified by the data size					
	processed. flag is turn pushed, the n, the FUL off. Whe	rocessed. When the value of the pointer is equal to the data size, the FULL ag is turned on and nothing is processed. After the contents of the source are ushed, the contents of the pointer are incremented. When the pointer reaches, the FULL flag is turned on. Otherwise, the ZERO and FULL flags are turned ff. When TB is a constant, it is assumed to be the table address.								
Examples	Ф psh	\diamondsuit PSH FW000, DW000								
	FW000 1234 PSH									
		DW000								
		/ DW004								
		DW005	Pointe	er						
		DW006	Data	1						
			Data	2						
		DW008	1234							
Valid						-				
parameters	S	Bit-type PI/O	Word-type PI/O	Constant	ТВ	Bit-type PI/O	Word-type PI/O	Constant		
\triangle : A parameter	Direct word-length	0	0	0	Direct word-length	×	0	\bigtriangleup		
when an odd-	Direct long-length	\bigcirc	0	0	Direct long-length	×	0	\bigtriangleup		
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indirect word-length	×	\bigtriangleup	\bigtriangleup		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup		

POP	FIFO POF	D									
Function	Pops data f The FIFO t	from the I table cont	FIFO table tains only	e and store word-len	es the popped gth data.	data in tl	ne destina	tion.			
Parameters and processing	POPD: DestinTB: First a	TB, D ation ddress of	the FIFO	table	Рорг	bed data	FIFC n (da ZEF add	D table ita size) RO flag_ ress			
Flags	The setting The other f	; of the E lags are t	flag chan urned off	ges.	Pointer D	ata 1 🔸 ata 2 🔫	FUL	L flag _ ress			
Processing time	0.32+0.01>	<n ms<="" td=""><td></td><td></td><td>$\longrightarrow D_i$</td><td>ata 3 🔫 ata 4</td><td></td><td>ata 1</td></n>			$ \longrightarrow D_i$	ata 3 🔫 ata 4		ata 1			
Remarks	When n is greater that When the v than 0 or the value of the processed. nothing is p decremente Otherwise, is assumed	Vhen n is less than or equal to 0 or preater than 256, nothing is processed. When the value of the pointer is less han 0 or the data size is less than the ralue of the pointer, noting is processed. When the value of the pointer is 0, the ZERO flag is turned on and nothing is processed. After data is popped, the contents of the pointer are lecremented. When the pointer reaches 0, the ZERO flag is turned on. Otherwise, the ZERO and FULL flags are turned off. When TB is a constant, it is assumed to be the table address.									
Examples	O POP	POP DW000, FW000									
Valid		FW000 DW000 DW004 DW005 DW006 DW007 DW008 DW009	Point 1234 Point 1234 Data Data Data	4 er 4 2 3 4							
v alla parameters		Bit-type	Word-type			Bit-type	Word-type				
parameters	IB	PI/O	PI/O	Constant	D	PI/O	PI/O	Constant			
\triangle : A parameter error is detected	Direct word-length	×	0	\bigtriangleup	Direct word-length	0	0	×			
when an odd-	Direct long-length	×	0	\bigtriangleup	Direct long-length	0	0	×			
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indirect word-length	×	\bigtriangleup	\bigtriangleup			
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup			

ADDRESS SET										
Transfers the	he addres	s of the so	ource to the	ne destin	nation	. Valid	only for P	PI/O.		
↓ AST \$	S, D				Add	ress of S	$\rightarrow D$			
S: Source										
D: Destinat	tion									
The setting	of the E	flag chan	ges. Th	e other f	flags a	re turned	off.			
0.25 ms										
When word data.	When word-length data is specified in D, the address is converted to word-length data.									
$ \begin{array}{c c} AST FW000, [FW002] \\ FW000 \\ FW002 \\ FW003 \\ \hline 2000 \\ \hline Address \\ of X000 \\ \hline 000A \\ \hline 2 \\ \hline 0000 \\ \hline \end{array} $										
	Dittion	Word two				Dit turo	Mord two			
S	PI/O	PI/O	Constant		D	PI/O	PI/O	Constant		
Direct word-length	0	0	×	Direc word	t -length	0	0	×		
Direct long-length	\bigcirc	\bigcirc	×	Direct long-	t length	\bigcirc	\bigcirc	×		
Indirect word-length	×	×	×	Indire word	ect -length	×	\bigtriangleup	\bigtriangleup		
Indirect long-length	×	×	×	Indire long-	ect length	×	\bigtriangleup	\bigtriangleup		
	Transfers the setting AST S S: Source D: Destinate The setting 0.25 ms When word data. AST F AST F AST F Direct word-length Direct long-length Indirect long-length	Transfers the address \Phi AST S, D S: Source D: Destination The setting of the E 0.25 ms When word-length of data. \Phi AST FW000, FW000 - FW002 FW003 \Phi AST X000, @ Address of X000 S Bit-type PI/0 Direct Iong-length Direct Indirect Xword-length Xword-length	Transfers the address of the so	S Bit-type Word-type Constant S Bit-type Word-type Constant Direct O × Indirect × Nord-length S S S S	S Bit-type Word-type Constant Direct O X Direct Direct O X Direct Indirect X X X	Source Add D: Destination Add The setting of the E flag changes. The other flags at 0.25 ms The other flags at 0.25 ms When word-length data is specified in D, the address data. FW000 FW000 FW002 FW001 FW002 FW003 OOOE Address Address Address H180000 of X000 2 S Bit-type PI/O PI/O Direct O Nord-length Constant Direct O Nord-length N Indirect X Word-length Indirect Nord-length Indirect Indirect X Nord-length Indirect Indirect X X	S Bit-type Word-type Constant O X X X D O X D D D D D D D D O X X X D </td <td>Transfers the address of the source to the destination. Valid only for P \diamondsuit AST S, D Address of S \rightarrow D S: Source D D: Destination The other flags are turned off. 0.25 ms When word-length data is specified in D, the address is converted to wordata. \diamondsuit AST FW000, [FW002] FW000 FW002 000E FW003 000E Address Address \diamondsuit AST X000, @ [H180000] Address H180000 000A 0000 S Bit-type PI/O PI/O PI/O PI/O Direct O Indirect X word-length X Indirect X Mord-length X Indirect X</td>	Transfers the address of the source to the destination. Valid only for P \diamondsuit AST S, D Address of S \rightarrow D S: Source D D: Destination The other flags are turned off. 0.25 ms When word-length data is specified in D, the address is converted to wordata. \diamondsuit AST FW000, [FW002] FW000 FW002 000E FW003 000E Address Address \diamondsuit AST X000, @ [H180000] Address H180000 000A 0000 S Bit-type PI/O PI/O PI/O PI/O Direct O Indirect X word-length X Indirect X Mord-length X Indirect X		

SCH	SEARCH										
Function	Searches the destination by the results source are	ne destina before th lt parame found.	tion for the specific eter, the m	ne content ed area (m umber (n)	s of the sourc) is reached. of the steps v	e from th Stores, where the	e beginnin in the area contents o	ng of the a specified of the			
Parameters and	\diamondsuit scн	S, D, m,	R		S		_	R			
processing	S: Source D: Destina m: Number R: Result	tion r of steps	to be sear	ched	Data 5 D (0) (1)	Search ta		<u>n</u>			
Flags	The setting The other f	g of the E lags are t	flag chan urned off.	ges.	-Search rar						
Processing time	0.45+0.01>	<n ms<="" td=""><td></td><td></td><td>(n)</td><td>Data</td><td></td><td></td></n>			(n)	Data					
Remarks	When m is	less than	or equal	to 0 or	(m-1)						
	A match or encountere area specif data to be s detected.	A match occurs when the first occurrence of the contents of the source is encountered. If a match is not found within the search range, -1 is set in the area specified by the result parameter. When the type (long or word) of the data to be searched is different from that of data in the destination, an error is detected. The step numbers (n) in the destination start with 0.									
Examples	♦ SCH	🗘 SCH DW000, FW000, 5, FW005									
		DW000	1234		→ Se	earch					
		FW000 FW001 FW002 FW003 FW004 FW005	0000 1234 0000 1234 0000 0001	First occurr	ence						
Valid		Ditt					NA				
parameters	S, m	Bit-type PI/O	vvord-type PI/O	Constant	D, R	Bit-type PI/O	vvord-type PI/O	Constant			
\triangle : A parameter	Direct word-length	0	0	0	Direct word-length	0	0	×			
when an odd-	Direct long-length	\bigcirc	\bigcirc	\bigcirc	Direct long-length	\bigcirc	0	\times			
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indirect word-length	×	\bigtriangleup	\bigtriangleup			
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup			

BTD	$BINARY \to BCDCONVERSION$											
Function	Converts the contents of the source from the binary form to the BCD form and stores the result in the area specified by the result parameter. \bigcirc BTD S R (BCD)											
Parameters and	O BTD	S, R				S (bi	inary) \rightarrow	R (BCD)			
processing	S: Source R: Result											
Flags	The setting	s of the H	E and V fl	ags chang	ge. The	other	r flags are	e turned of	ff.			
Processing time	1.03 ms											
Remarks	When S < 0 V flag is tu	When $S < 0$, nothing is processed. In this case, the E flag is turned on and the 7 flag is turned off. If an overflow occurs, H9999 or H99999999999999999999999999999										
Examples	ф втd	$\begin{array}{c c} & \text{BID FW000, FW002} \\ & \text{FW000} & 007B \\ & \text{FW002} & 0123 \\ \hline \end{array} \\ \hline \text{FW002} & 0123 \\ \hline \end{array} \\ \hline \text{BTD} \\ \hline \end{array} \\ \hline \text{BTD} \\ \hline \end{array} \\ \hline \begin{array}{c} \text{BTD} \\ \text{BTD} \\ \hline \end{array} \\ \hline \begin{array}{c} \text{BTD} \\ 1234 \\ 2 \\ \hline \end{array} $										
Valid												
parameters	S	Bit-type PI/O	Word-type PI/O	Constant	F	۲	Bit-type PI/O	Word-type PI/O	Constant			
\triangle : A parameter	Direct word-length	0	0	0	Direct word-	t length	0	0	×			
when an odd-	Direct long-lengthOODirect long-lengthO×											
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indire word-	ct length	×	\bigtriangleup	\bigtriangleup			
uuuress 15 useu.	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indire long-l	ct ength	×	\bigtriangleup	\bigtriangleup			

DTB	$BCD \rightarrow BINARY CONVERSION$											
Function	Converts the stores the r	ne conten esult in tl	ts of the s he area sp	ource from	n the B the res	CD fo ult pa	rm to the rameter.	binary for	rm and			
Parameters and		S, R				S (B	CD) \rightarrow	R (binary)			
processing	S: Source R: Result											
Flags	The setting	s of the H	E and V fla	ags chang	e. The	e othe	r flags are	e turned of	ff.			
Processing time	0.46 ms											
Remarks	If any of A	to F is u	sed in S, t	he E flag	is turne	d on a	nd nothir	ig is proce	essed.			
Examples	џ dtb	FW000, FW000 FW002 H9999999 H9999999	FW002 1234 04D2 999, @ [H 999 — (H180000	(→ H	DTB) 00 05F 2 E0F	5 F				
Valid		Rit type	Word type		·		Rit type	Word type				
parameters	S	PI/O	PI/O	Constant		R	PI/O	PI/O	Constant			
\triangle : A parameter	Direct word-length	\bigcirc	0	0	Direc word	rt -length	0	0	×			
when an odd-	Direct long-length	Direct Org-length O O O O Direct O O X										
numbered	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indir word	ect -length	×	\bigtriangleup	\bigtriangleup			
uuuress 15 useu.	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indire long-	ect length	×	\bigtriangleup	\bigtriangleup			

SEG	BINARY	→ 7-S	EG	MENT	CONV	ERS	SIC	ON				_		
Function	Converts the stores the r	ie conte esult in	ents 1 the	of the s e area sp	source fi becified	om by t	the	e bina result	ry f par	orm ame	to 7- ter.	-segme	ent c	lata and
Parameters and	♦ SEG	S, R						S (bina	ary)	\rightarrow	R (7-s	egn	ent data)
processing	S: Source							L						
	R: Result													
Flags	The setting	, of the	Εfl	ag char	nges. 7	he o	oth	er flag	gs a	re tu	rned	off.		
Processing time	0.35 ms													
Remarks	S is double	d in siz	ze ar	nd writt	en in R.									
Examples	🗘 seg	FW00	0, F	W002										
		FW00(0	567	8									
		FW002	2	585	<u>F</u> ←			- Con	ivers ame	ion to nt dat	7- a			
		FW003	3	707	F				3					
		ייהרר	~4 0		TI 14 0 0 0	<u></u>								
	\bigcirc Seg	♀ SEG HDEF01234, @ [H180000]												
		HDEE	∩12	·34 —	Convei	sion	to	7-	→⊢	1120	000	30	14F	1
			012	Ъ	Segm	enco	ldia		. 1	1100	2	47	7E	
											4	30	6D	
											6	79	33]
	7-segn	nent co	orre	sponde	ence ta	ole								n
	No.	0	1	2 3	4 5	6	,	7 8	9	Α	В	C]	D	E F
	Data	7E 3	6 0	D 79	33 5B	5F	7(0 7F	7B	77	1F	4E 31	D 4	F 47
Val: 4														
Valla parameters		Bit-type	e V	Vord-type			Г			Bit-t	vpe	Word-	tvpe	
parameters	S	PI/O		PI/O	Constar	nt		R		PI	/0	PI/0	0	Constant
\triangle : A parameter	Direct word-length	\bigcirc		\bigcirc	\bigcirc		I v	Direct	noth	C	\supset	0)	\times
error is detected	Direct	\cap		\bigcirc			I	Direct	19	C))	\times
when an odd-	long-length		+	\bigcirc		4	10	ong-leng	gth))	~
numbered	word-length	×		\bigtriangleup	\bigtriangleup		1 V	naireci vord-len	ngth	>	<	\triangle	7	\bigtriangleup
addiess 15 used.	Indirect long-length	×		\bigtriangleup	\triangle		I: 1e	ndirect ong-leng	gth	>	<	\triangle	7	\bigtriangleup
							-							

ASP	BINARY → ASCII CONVERSION (PACKED MODE)															
Function	Converts th	ne conten	ts of	the s	soui	ce fi	on	n tł	he l	oina	ry fo	orm	to A	SCII da	ata	and
	stores the r	esult in the	he ar	rea sp	beci	fied	by	the	e re	esult	par	ame	ter ii	n packe	d r	node.
Parameters and	\diamondsuit ASP :	S, R								S (bina	ıry)	\rightarrow	R (ASC	CII,	packed)
processing	S: Source															
	R: Result															
Flags	The setting	of the E	flag	char	nges	з. Т	he	ot	her	flag	gs ai	re tu	rned	off.		
Processing time	0.47 ms															
Remarks	S is double	d in size	and	writt	en i	n R.										
Examples	0 ASP	FW000,	FW	002												
		FW000		5678	8 -					_						
	I	FW002		3530	<u>6</u> +				-(Co	nver SCII	sion t data	•)			
		FW003		3738	8											
			7 24	0	பா	٥٩٩	00.	1								
	\downarrow ASF	↓ ASP HDEF01234, @ [H180000]														
		HDFF01	234		(Conve	rsic II da	on t	0)	→ Н	180	000	444	5]
			_0.							·	• •		2	463	0	-
													4	313	2	
													6	333	4	
	Corres	pondenc	ce ta	ible I	bet	wee	ר A	S۱)	CII	and	d bi	nary	dat	a	1	
	Binary	0 1	2	3	4	5	6		7	8	9	Α	В	C D		E F
	Data	30 31	32	33	34	35	36	3	37	38	39	41	42	43 44	. 4	5 46
Valid																
v allu narameters		Bit-type	Word	d-tvpe				Γ		_		Bit-t	vpe	Word-tv	pe	
parameters	S	PI/O	Р	I/O	Co	onstar	t			R		PI/	0	PI/O	İ	Constant
\triangle : A parameter	Direct word-length	\bigcirc	(С		\bigcirc			Dir woi	ect d-len	gth	C)	0		\times
error is detected	Direct	\bigcirc	($\overline{)}$		\bigcirc			Dir	ect	0.	C)	\cap		×
when an odd-	long-length Indirect	0				0		-	long	g-leng	gth					
address is used	word-length	×	4	\bigtriangleup		\bigtriangleup			woi	d-len	gth	>	<	\triangle		\bigtriangleup
uuur 055 15 useu.	Indirect	×		$ \land $		\bigtriangleup		ſ	Ind	irect	oth	×	<	\triangle		\bigtriangleup
	iong-iongui		<u> </u>					L	1011	5-10118	501					

ASU	BINARY ·	→ ASCI	I CONVE	RSION	(UI	NPAC	KED) MODE)	
Function	Converts the stores the r	ne conten esult in th	ts of the s ne area sp	ource fro ecified by	m t / th	he bina e result	ry f t par	form to A cameter in	SCII data 1 unpacke	and d mode.
Parameters and		S, R				S (b	oinai	$ry) \rightarrow R$	(ASCII,	unpacked)
processing	S: Source R: Result									
Flags	The setting	of the E	flag chan	ges. Th	e of	ther fla	gs a	re turned	off.	
Processing time	0.49 ms									
Remarks	S is multip	lied by fo	our in size	and write	en	in R.				
Examples	O ASU	FW000,	FW002							
		FW001 FW002 FW003 FW004 FW005	5678 3035 3036 3037 3038	3 		Cc	onver ASCI	rsion to I data		
	♀ ASU Corres	HDEF01 HDEF01 pondenc	234, @ 234 ce table t	H18000 Convers ASCII	0] ion t data	CII an	d bi	→ H18(nary dat	0000 3 2 3 4 3 6 3 8 3 A 3 C 3 E 3	044 045 046 030 031 032 033 034
	Binary	0 1	2 3	4 5	6	7 8	9	A B	C D	E F
	Data	30 31	32 33	34 35 3	6	37 38	39	41 42	43 44 4	45 46
Valid							1			
parameters	S	Bit-type PI/O	Word-type PI/O	Constant		R		Bit-type PI/O	Word-type PI/O	Constant
\triangle : A parameter	Direct word-length	\bigcirc	0	\bigcirc		Direct word-ler	ngth	\bigcirc	0	×
when an odd-	Direct long-length	\bigcirc	0	\bigcirc		Direct long-len	gth	\bigcirc	0	\times
numbered address is used.	Indirect word-length	×	\bigtriangleup	\bigtriangleup		Indirect word-ler	ngth	×	\bigtriangleup	\bigtriangleup
	Indirect long-length	×	\bigtriangleup	\bigtriangleup		Indirect long-len	gth	×	\bigtriangleup	\bigtriangleup

APB	ASCII \rightarrow BINARY CONVERSION (PACKED MODE)											
Function	Converts the binary form	e content and stor	ts of the s res the res	source fro sult in the	om . e are	ASC ea sj	CII d pecif	ata fied	in packe by the 1	ed mode te esult para	o the umeter.	
Parameters and	O APB S	5, R					S (.	ASC	CII, pacl	$(xed) \rightarrow 1$	R (binary)	
processing	S: Source											
	R: Result											
Flags	The setting	of the E	flag chan	ges. T	he o	the	flag	gs ai	re turned	d off.		
Processing time	0.57 ms											
Remarks	Data with the data in S contained and nothing	e size in ntains ar is proce	R multip y of H30 ssed.	to H39	wo and	is fo H4	etche 1 to 1	ed fi H46	from S and 5, the E is	nd conver flag is tur	ted. If ned on	
Examples	O APB F	W000,	FW002									
	F	W000	3132]			7					
	F	W001	3334]								
	F	W002	1234	←		-(Conv bin	/ersi ary c	on to lata			
	\diamondsuit apb d	→ APB DW000, @ [H180000]										
	D	0W000 1 2 3	4645 4443 3938 3736	Con bin	versi ary (ion to data	~)-	→	H1800	00 FED 2 987	C 6	
	Corresp	ondenc	e table k	between	AS	SCII	and	l bii	narv da	ta		
	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	
Valid parameters		Rit-type	Word-type		٦				Rit-type	Word-type	<u></u>	
parameters	S	PI/O	PI/O	Constant	:		R		PI/O	PI/O	Constant	
\triangle : A parameter	Direct word-length	$\begin{array}{c c c c c c c c c c c c c c c c c c c $										
error is detected when an odd-	Direct long-length	$\begin{array}{c c c c c c c c c c c c c c c c c c c $										
numbered	Indirect word-length	×	\triangle	\bigtriangleup		Ind wo	irect rd-len	gth	×	\triangle	\bigtriangleup	
add1055 15 0500.	Indirect long-length	×	\bigtriangleup	\bigtriangleup		Ind lon	irect g-leng	gth	×	\bigtriangleup	\bigtriangleup	
					_							

AUB	ASCII \rightarrow	BINA	RY	′ CO	NVE	R	SIOI	N (UI	NP	ACł	KED) MC	DE)				
Function	Converts the binary form	ne cont n and s	tent stor	ts of t es th	the s e res	our ult	ce fi in th	ron ne a	n A are	ASC ea sj	CII c peci	lata fied	in u by t	npac he re	ked esult	moc t par	le to ame	o the eter.	
Parameters and	Ó AUB	S, R									S (A	ASC	II, u	npac	ked	$) \rightarrow$	R	(bina	ary)
processing	S: Source R: Result																		
Flags	The setting	of the	Ε	flag (chan	ges	. Т	The	0	the	flag	gs a	re tu	rned	off.				
Processing time	0.57 ms																		
Remarks	Data with t data in S co and nothing	he size ontains g is pro	e in s of oce	R m any ssed.	ultip H30	to	d by H39	for an	ur d	is f H4	etch 1 to	ed f H46	rom 6, the	S ar e E f	nd co lag i	onve s tui	rtec	l. I l on	f
Examples	ф aub	FW00	1,	FW0	02														
	ф AUB	FW001 3035 FW002 3036 FW003 3037 FW004 3038 FW005 5678 Conversion to binary data DW000 1130 DW000 1130 DW000 1130 Conversion to binary data 2 4567 0032 2233 3334 4435 5536 TTTT																	
	Corres	ponde	enc	e tal	ole b	et\	vee	n A	١S	CII	an	d bi	nary	dat	a				
	Binary	0	1	2	3	4	5	6)	7	8	9	Α	В	С	D	E	F	
	ASCII	30	31	32	33	34	35	36)	37	38	39	41	42	43	44	45	46	
Valid																			
parameters	S	Bit-typ PI/O	e	Word Pl/	-type ′O	Со	onstar	nt			R		Bit-t PL	ype /O	Wor F	™d-typ PI/O	e C	consta	ant
\triangle : A parameter	Direct word-length	\times		C)		\times			Dir wo	ect rd-ler	ngth	C)		0		\times	
when an odd-	Direct long-length	×		C)		\times			Dir lon	ect g-len	gth	C)		0		×	
numbered address is used.	Indirect word-length	×		Δ	7		\bigtriangleup			Ind wo	irect rd-ler	ngth	>	<		\triangle		\triangle	
	Indirect long-length	×		Δ	7		\bigtriangleup			Ind lon	irect g-len	gth	>	<		\bigtriangleup		\triangle	

ABS	ABSOLUT	FE VALU	JE						
Function	Stores the a the result p	absolute v varameter	value of th	ne content	s of the	sourc	e in the a	rea specif	ied by
Parameters and	0 ABS	S, R				S	$\rightarrow R$		
processing	S: Source R: Result								
Flags	The setting	s of the H	E and V fl	ags chang	e. The	othe	r flags are	e turned o	ff.
Processing time	0.32 ms								
Remarks	If an overfiparameter.	low occur	rs, H7FFF	FFFF is s	et in the	area	specified	by the rea	sult
Examples	ф Abs ф Abs Н	FW000, FW002 FW002 DW000, DW000 180000 2	FW002 FF90 0064 @ [H180 FFFE 0000 0005	2 	(A	Absol	ute value		
Valid		Rit-type	Word-type				Rit-type	Word-type	
parameters	S	PI/O	PI/O	Constant	F	२	PI/O	PI/O	Constant
\triangle : A parameter	Direct word-length	\bigcirc	0	0	Direct word-	t length	\bigcirc	0	×
when an odd-	Direct long-length	\bigcirc	\bigcirc	0	Direct long-l	ength	\bigcirc	\bigcirc	×
numbered	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indire word-	ct length	×	\bigtriangleup	\bigtriangleup
uuuress 15 useu.	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indire long-l	ct ength	×	\bigtriangleup	\bigtriangleup

а.										
Sign-converts the contents of the source and stores the result in the area specified by the result parameter. $\bigcirc NECS P$ $-S \rightarrow R$										
0 NEG	S, R				-S -	→ R				
S: Source R: Result										
The setting	s of the E	E and V fl	ags chang	e. Th	e othe	r flags are	e turned of	ff.		
0.32 ms										
If an overfl result parar	ow occur neter.	rs, H7FFF	or H7FF	FFFFF	is set i	in the area	a specified	d by the		
<pre>♀ NEG</pre> ↓ NEG H	FW000, FW000 FW002 DW000, DW000 180000 2	FW002 1000 F000 @ [H180 1234 FFFF EDC0		(NEG)				
c	Bit-type	Word-type	Constant		D	Bit-type	Word-type	Constant		
5	PI/O	PI/O	Constant		R.	PI/O	PI/O	Constant		
Direct word-length	\bigcirc	\bigcirc	\bigcirc	Direct	et -length	\bigcirc	\bigcirc	\times		
Direct long-length	\bigcirc	0	0	Direction Direct	et length	0	0	×		
Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indir word	ect -length	×	\bigtriangleup	\bigtriangleup		
Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indir long-	ect ·length	×	\bigtriangleup	\bigtriangleup		
	specified by Specified by NEG S: Source R: Result The setting 0.32 ms If an overfl result paran O NEG NEG H S Direct long-length Indirect long-length Indirect long-length	Sign-converts the example of the setting of the resurce NEG S, R S: Source R: Result The settings of the E 0.32 ms If an overflow occurresult parameter. NEG FW000, FW002 NEG DW000, FW002 NEG DW000, DW000 H180000 2 S Bit-type PI/O Direct Indirect word-length Indirect Word-length	Sign-converts the contents of specified by the result parameter Image: Problem stress of the settings of the E and V flat 0.32 ms If an overflow occurs, H7FFF result parameter. Image: Problem stress If an overflow occurs, H7FFF result parameter. Image: Probability of the parameter. Indirect Image: Prive Prive Prive Prive Prive Prive Prive Prive Prive Prive Prive Prive Image: Prive Prive <td< td=""><td>Signe-converts the contents of the source Specified by the result parameter. NEG S, R S: Source R: Result The settings of the E and V flags chang 0.32 ms If an overflow occurs, H7FFF or H7FF, result parameter. NEG FW000, FW002 FW002 F000 FW002 F000 FW002 F000 FW000 1234 H180000 FFFF EDCC EDCC S Bit-type Word-type PI/O PI/O O Direct O O Indirect X A Indirect X A</td><td>Signeconvertes the contents of the source and staspecified by the result parameter. \bigcirc NEG S, R S: Source R: Result The settings of the E and V flags change. Th 0.32 ms If an overflow occurs, H7FFF or H7FFFFFF result parameter. \bigcirc NEG FW000, FW002 FW000 1000 FW002 F000 \bigcirc NEG DW000, @ [H180000] DW000 1234 H180000 FFFF EDCC S Bit-type NEG DW000, @ [H180000] DW000 DW000 1234 H180000 FFFF EDCC O Direct O Direct O Direct O Direct O Indirect \land Indirect \land Indirect \land</td><td>Signeton vertes the contents of the source and stores the specified by the result parameter. -S - S: Source R: Result The settings of the E and V flags change. The other 0.32 ms If an overflow occurs, H7FFF or H7FFFFFFFF is set is result parameter. V NEG FW000, FW002 FW000 1000 FW002 F000 V NEG DW000, @ [H180000] DW000 1234 H180000 FFFF Q NEG DW000, @ [H180000] DW000 1234 H180000 FFFF V NEG DW000, @ [H180000] DW000 1234 H180000 FFFF V NEG DW000 1234 H180000 FFFF V NEG DW000 1234 H180000 FFFF Vord-length NEG Direct O Word-length O Direct O Indirect A Indirect A Indirect A </td><td>Significative intermitted into source and stores the result is specified by the result parameter. $S \rightarrow R$ Significative intermitted into source and stores the result is specified by the result parameter. $S \rightarrow R$ Significative intermitted into source and stores the result is specified by the result parameter. $S \rightarrow R$ If an overflow occurs, H7FFF or H7FFFFFFFF is set in the area result parameter. $I = 0$ NEG FW000, FW002 $FW002$ FW002 $I = 000$ FW002 $I = 000$ FW000 $I = 000$ DW000 $I = 234$ H180000 $FFFF$ EDCC $N = 0$ Direct $O O$ Direct $O O$ Indirect $A O$</td><td>Subjectorizers are contents of the source and stores the result in the area specified by the result parameter.</td></td<>	Signe-converts the contents of the source Specified by the result parameter. NEG S, R S: Source R: Result The settings of the E and V flags chang 0.32 ms If an overflow occurs, H7FFF or H7FF, result parameter. NEG FW000, FW002 FW002 F000 FW002 F000 FW002 F000 FW000 1234 H180000 FFFF EDCC EDCC S Bit-type Word-type PI/O PI/O O Direct O O Indirect X A Indirect X A	Signeconvertes the contents of the source and staspecified by the result parameter. \bigcirc NEG S, R S: Source R: Result The settings of the E and V flags change. Th 0.32 ms If an overflow occurs, H7FFF or H7FFFFFF result parameter. \bigcirc NEG FW000, FW002 FW000 1000 FW002 F000 \bigcirc NEG DW000, @ [H180000] DW000 1234 H180000 FFFF EDCC S Bit-type NEG DW000, @ [H180000] DW000 DW000 1234 H180000 FFFF EDCC O Direct O Direct O Direct O Direct O Indirect \land Indirect \land Indirect \land	Signeton vertes the contents of the source and stores the specified by the result parameter. -S - S: Source R: Result The settings of the E and V flags change. The other 0.32 ms If an overflow occurs, H7FFF or H7FFFFFFFF is set is result parameter. V NEG FW000, FW002 FW000 1000 FW002 F000 V NEG DW000, @ [H180000] DW000 1234 H180000 FFFF Q NEG DW000, @ [H180000] DW000 1234 H180000 FFFF V NEG DW000, @ [H180000] DW000 1234 H180000 FFFF V NEG DW000 1234 H180000 FFFF V NEG DW000 1234 H180000 FFFF Vord-length NEG Direct O Word-length O Direct O Indirect A Indirect A Indirect A	Significative intermitted into source and stores the result is specified by the result parameter. $S \rightarrow R$ Significative intermitted into source and stores the result is specified by the result parameter. $S \rightarrow R$ Significative intermitted into source and stores the result is specified by the result parameter. $S \rightarrow R$ If an overflow occurs, H7FFF or H7FFFFFFFF is set in the area result parameter. $I = 0$ NEG FW000, FW002 $FW002$ FW002 $I = 000$ FW002 $I = 000$ FW000 $I = 000$ DW000 $I = 234$ H180000 $FFFF$ EDCC $N = 0$ Direct $O O$ Direct $O O$ Indirect $A O$	Subjectorizers are contents of the source and stores the result in the area specified by the result parameter.		

DCD	DECODE	DECODE											
Function	Decodes th	e content	ts of the s	ource and	stores the res	ult in the	area spec	ified by					
	the result p	arameter	•										
Parameters and	\diamondsuit dcd	S, R	S	n	\rightarrow								
processing	S: Source				0	r		(LSB)					
	R: Result				R 0 ~	0	1 0 ~	- 0					
Flags	The setting	, of the E	flag chan	ges. The	e other flags a	re turned	off.						
Processing time	0.38 ms												
Remarks	The valid b	oits in S d	lepend on	the data s	ize specified	in R. W	hen word	-length is					
	specified, t	he low-o	rder four	bits are va	lid. When l	ong-leng	th is speci	fied, the					
	low-order	five bits a	are valid.										
Examples	\Diamond dcd	FW000,	FW002										
		FW000	0003	<u>i</u>									
		FW002 1000 CD											
		FVV002 1000 CD											
	\downarrow DCD												
		DW000	0000	:									
		DWOOT	0011										
	Н	180000	0000)		>							
		2	0001										
Valid				1									
parameters	S	Bit-type PI/O	Word-type PI/O	Constant	R	Bit-type PI/O	Word-type PI/O	Constant					
\triangle : A parameter	Direct word-length	0	0	\bigcirc	Direct word-length	\bigcirc	0	×					
error 1s detected when an odd-	Direct long-length	0	0	0	Direct long-length	0	0	×					
numbered address is used.	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indirect word-length	×	\bigtriangleup	\bigtriangleup					
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup					

ECD	ENCODE										
Function	Encodes the the result p	e content arameter	s of the so	ource and	stores the res	ult in the	area spec	ified by			
Parameters and processing	¢ ECD S: Source R: Result	S, R	S	0 0 →	n ~ 0 1 1 S _ n	?~	(LSB) ?				
Flags	The setting	of the E	flag chan	ges. Th	e other flags a	re turned	off.				
Processing time	0.38+0.01>	<n ms<="" td=""><td></td><td></td><td></td><td></td><td></td><td></td></n>									
Remarks	When S=0 to 1, when	, nothing it is searc	is process thed from	sed. Onl the most	y one bit is de significant bi	ecoded. t (MSB).	It is the f	irst bit set			
Examples	џ еср ф еср н	FW000, FW000 FW002 [DW000 DW000 DW001 180000 2	FW002 0456 0005], @ [H18 0000 0080 0080	6 6 80000] 0 0 0 0 0 0 0 0 0 0 0 0 0	ECD)					
Valid parameters	9	Bit-type	Word-type	Constant	В	Bit-type	Word-type	Constant			
1	5	PI/O	PI/O	Constant		PI/O	PI/O	Constant			
\triangle : A parameter	word-length O O O Word-length O O X										
error is detected when an odd-	Direct long-length	\bigcirc	0	0	Direct long-length	0	0	×			
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indirect word-length	×	\bigtriangleup	\bigtriangleup			
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup			

LSR	LOGICAL SHIFT RIGHT												
Function	Shifts the c stores the r	contents c result in th	of the sour he area sp	ce right b ecified by	y tl / th	ne contents e result par	of the decameter.	estination,	and				
Parameters and processing	 Control <	S, D, R tion	S R	0 0 0 ~ 0	*	RS-D	(LSI	 B) The let dependent depend	ngth of RS ls on the pe; 15 bits rd-length or g-length.				
Flags	The setting	g of the E	flag chan	ges. Th	e ot	her flags a	re turned	off.					
Processing time	0.37 ms												
Remarks	The valid to specified, to low-order to	The valid bits in D depend on the data type in S. When word-length is pecified, the low-order four bits are valid. When long-length is specified, the ow-order five bits are valid.											
Examples	♀ LSR ♀ LSR H	FW000, FW000 FW001 FW002 [DW000] DW000 DW001 180000 2	FW001, 0456 0004 0045 , 2, @ [H 8765 4321 21D9 50C8	FW002)	- 2					
Valid		Bit-type	Word-type				Bit-type	Word-type					
parameters	S, D	PI/O	PI/O	Constant		R	PI/O	PI/O	Constant				
\triangle : A parameter	Direct word-length	0	0	0		Direct word-length	\bigcirc	0	×				
when an odd-	Direct long-length	0	0	\bigcirc		Direct long-length	\bigcirc	0	×				
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup		Indirect word-length	×	\bigtriangleup	\bigtriangleup				
	Indirect long-length	×	\bigtriangleup	\bigtriangleup		Indirect long-length	×	\bigtriangleup	\bigtriangleup				
LSL	LOGICAL	SHIFT I	_EFT										
------------------------------	---	---	--	--	------------------------------	------------------------	----------------------------	------------------	--	--			
Function	Shifts the c the result in	Shifts the contents of the source left by the contents of the destination, and stores the result in the area specified by the result parameter.											
Parameters and processing	O D (LSB) S: Source S Image: Source The length of LSE depends on the data type; 15 bits R: Result 0 (LSB) The length of LSE depends on the data type; 15 bits D: Destination R 0 ~ 0 31 bits for long-length.												
Flags	The setting	The setting of the E flag changes. The other flags are turned off.											
Processing time	0.37 ms												
Remarks	The valid t specified, t low-order	oits in D o he low-o five bits a	lepend on rder four l are valid.	the data to the da	type in S. W llid. When l	/hen word ong-lengt	d-length is th is speci	fied, the					
Examples	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$												
Valid		Bit-type	Word-type		·	Bit-type	Word-type						
parameters	S, D	PI/O	PI/O	Constant	R	PI/O	PI/O	Constant					
\triangle : A parameter	Direct word-length	\bigcirc	0	0	Direct word-length	0	0	×					
when an odd-	Direct long-length	0	0	0	Direct long-length	0	0	×					
numbered address is used.	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indirect word-length	×	\bigtriangleup	\bigtriangleup					
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup					

n (the sign parameter. length of RS nds on the type; 15 bits 'ord-length bits for -length.
length of RS nds on the type; 15 bits 'ord-length bits for ·length.
off.
alid bits in ow-order bits are
Constant
×
×
\bigtriangleup
\bigtriangleup

ASL	ARITHMETIC SHIFT LEFT								
Function	Shifts the contents of the source left by the contents of the destination, and stores the result in the area specified by the result parameter. Sets the full-scale value if an overflow occurs.								
Parameters and processing	O ASL S, D, R S: Source S R: Result O D: Destination R Image: Contract of the second s								ngth of LSB ls on the pe; 15 bits rd-length or for long-
Flags	The setting	s of the H	E and V fl	ags chang	ge.	The other	r flags are	e turned of	ff.
Processing time	0.41 ms								
Remarks	The valid b specified, t low-order	oits in D c he low-o five bits a	lepend on rder four l re valid.	the data bits are va	typ 11id	e in S. W . When l	'hen word ong-lengt	d-length is th is speci	fied, the
Examples	 ASL FW000, FW001, FW002 FW000 0456 FW001 0004 FW002 4560 ASL ASL [DW000], 2, @ [H180000] DW000 4765 DW001 4765 4321 H180000 7FFF ASL Overflow (The V flag is turned or 								ned on.)
Valid		Ditterre	Manal true a				Ditterre	Mand two	
parameters	S, D	PI/O	PI/O	Constant		R	PI/O	PI/O	Constant
\triangle : A parameter	Direct word-length	0	0	0		Direct word-length	0	0	×
when an odd-	Direct long-length	0	0	0		Direct long-length	0	0	\times
numbered address is used.	Indirect word-length	×	\bigtriangleup	\bigtriangleup		Indirect word-length	×	\bigtriangleup	\bigtriangleup
	Indirect long-length	×	\bigtriangleup	\bigtriangleup		Indirect long-length	×	\bigtriangleup	\bigtriangleup

ROR	ROTATE	RIGHT	-						
Function	Rotates the stores the r	e contents result in th	of the some of the some of the some of the some of the solution of the solutio	urce right ecified by	by tl v the	he conten result par	ts of the cameter.	destinatio	n, and
Parameters and processing	 Contract Roman Contract Roman<td colspan="7">O RS-D (LSB) S: Source S Image: Source The length of RS depends on the data type; 15 bits for word-length or 31 bits for long-length. D: Destination Image: Source Image: Source Image: Source Image: Source</td>	O RS-D (LSB) S: Source S Image: Source The length of RS depends on the data type; 15 bits for word-length or 31 bits for long-length. D: Destination Image: Source Image: Source Image: Source Image: Source							
Flags	The setting	The setting of the E flag changes. The other flags are turned off.							
Processing time	0.37 ms								
Remarks	The valid t specified, t low-order	oits in D o he low-o five bits a	lepend on rder four l are valid.	the data t bits are va	type ilid.	in S. W When le	Then word ong-lengt	l-length is h is speci	fied, the
Examples	♥ ROR	FW000, FW001 FW002 [DW000 DW000 DW001 180000 2	FW001, 8456 0004 6845 0], 2, @ [l 8765 4321 61D9 50C8	FW002]	ROR)	- 2	
Valid		Bit-type	Word-type				Bit-type	Word-type	
parameters	S, D	PI/O	PI/O	Constant		R	PI/O	PI/O	Constant
\triangle : A parameter	Direct word-length	\bigcirc	0	0	D w	Direct Vord-length	\bigcirc	0	×
when an odd-	Direct long-length	0	0	0	D lo	Direct Dirg-length	\bigcirc	0	×
numbered address is used.	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Ir w	ndirect /ord-length	×	\bigtriangleup	\bigtriangleup
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Ir lo	ndirect ong-length	×	\bigtriangleup	\bigtriangleup

ROL	ROTATE	LEFT								
Function	Rotates the stores the r	Rotates the contents of the source left by the contents of the destination, and stores the result in the area specified by the result parameter.								
Parameters and processing	 ROL S: Source R: Result D: Destina 	OD(LSB)S: SourceSImage: Constraint of the second se								
Flags	The setting	The setting of the E flag changes. The other flags are turned off.								
Processing time	0.37 ms									
Remarks	The valid to specified, to low-order	oits in D o he low-o five bits a	depend on rder four l are valid.	the data to the da	type in S. W Ilid. When le	hen word ong-lengt	l-length is h is speci	fied, the		
Examples	♀ ROL ♀ ROL Н	 ROL FW000, FW001, FW002 FW000 8456 FW001 0004 4568 ROL ROL [DW000], 2, @ [H180000] DW000 8765 DW001 4321 H180000 1D95 ROL 								
Valid parameters		Bit-type	Word-type			Bit-type	Word-type			
parameters	S, D	PI/O	PI/O	Constant	R	PI/O	PI/O	Constant		
\triangle : A parameter	Direct word-length	Direct word-lengthOODirect word-lengthO×								
when an odd-	Direct long-length	0	0	0	Direct long-length	0	0	×		
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indirect word-length	×	\bigtriangleup	\bigtriangleup		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup		

LIM	LIMITER										
Function	Compares the contents of the source with the values of the specified boundaries (contents of destinations D1 and D2), and stores the result in the area specified by the result parameter.										
Parameters and processing	C LIM S S: Source R: Result D1, D2: D0	S, D1, D2	2, R		(F D1 ⁴	R) D1 D2	_ (i	S)			
Flags	The setting	The settings of the E and V flags change. The other flags are turned off.									
Processing time	0.52 ms										
Remarks	When D1 <	When $D1 < D2$, the E flag is turned on.									
Examples	$ \begin{array}{c c c c c c c c c c c c c c c c c c c $										
Valid parameters		Bit-type	Word-type	0		Bit-type	Word-type				
parameters	S, D1, D2	PI/O	PI/O	Constant	R	PI/O	PI/O	Constant			
\triangle : A parameter	Direct word-length	Direct O O Direct word-length O O X									
error 1s detected when an odd-	Direct long-length	0	0	0	Direct long-length	0	0	×			
numbered address is used.	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indirect word-length	×	\bigtriangleup	\bigtriangleup			
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup			

BND	DEAD BA	ND								
Function	Compares the contents of the source with the values of the specified boundaries (contents of destinations D1 and D2), and stores the contents within the boundaries as a dead area (data 0) in the area specified by the result parameter.									
Parameters and processing	O BND S: Source R: Result D1, D2: De	 ♦ BND S, D1, D2, R S: Source R: Result D1, D2: Destination (R) (R) (R) (R) (D2 (D1 								
Flags	The setting	s of the H	E and V fl	ags chang	e. The othe	r flags are	e turned of	ff.		
Processing time	0.52 ms									
Remarks	When D1 <	< D2, the	E flag is t	turned on.						
Examples	ф вnd	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$								
parameters	S. D1. D2	Bit-type	Word-type	Constant	R	Bit-type	Word-type	Constant		
\triangle : A parameter error is detected	Direct word-length	Direct word-lengthOOODirect word-lengthOOXDirect word-lengthOOOX								
when an odd-	long-length	0	0	0	long-length	0	0	×		
address is used.	word-length	×	\bigtriangleup	\bigtriangleup	word-length	×	\bigtriangleup	\bigtriangleup		
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup		

ZON	DEAD ZO	NE								
Function	Adds a bias (contents of destination D1 or D2) to the contents of the source according to its sign (positive or negative), and stores result in the area specified by the result parameter.									
Parameters and processing								ightarrow (S) nes are the	same.	
Flags	The setting	s of the H	E and V fl	ags chang	ge.	The other	flags are	e turned of	ff.	
Processing time	0.52 ms									
Remarks	When D1 <	< D2, the	E flag is t	turned on.						
Examples	$\begin{array}{c c c c c c c c c c c c c c c c c c c $									
Valid parameters	0.04.00	Bit-type	Word-type	Ormatant		5	Bit-type	Word-type	Ormatant	
parameters	S, D1, D2	PI/O	PI/O	Constant		R	PI/O	PI/O	Constant	
\triangle : A parameter	Direct word-length	word-length O O O O O O O O O O O O O O O O O O O								
when an odd-	Direct long-length	0	0	0		Direct long-length	0	0	×	
numbered address is used.	Indirect word-length	×	\bigtriangleup	\bigtriangleup		Indirect word-length	×	\bigtriangleup	\bigtriangleup	
	Indirect long-length	×	\bigtriangleup	\bigtriangleup		Indirect long-length	×	\bigtriangleup	\bigtriangleup	

ROT	ROOT										
Function	Stores the i area specif	Stores the integer part of the root obtained from the contents of the source in the area specified by the result parameter.									
Parameters and processing	 ROTS: SourceR: Result	S, R	V V	When S ≧ When S <	0: Root of S \rightarrow R \rightarrow R	→ R					
Flags	The setting	The settings of the E and V flags change. The other flags are turned off.									
Processing time	0.77 ms										
Remarks											
Examples	 ROT FW000, FW002 FW000 0456 FW002 0021 ROT [DW000], @ [H180000] DW000 0000 DW001 0080 H180000 0000 ROT 										
Valid		Rit_type	Word type			Rit type	Word-type				
parameters	S	PI/O	PI/O	Constant	R	PI/O	PI/O	Constant			
\triangle : A parameter	Direct word-length	\bigcirc	0	0	Direct word-length	\bigcirc	0	×			
when an odd-	Direct long-length	\bigcirc	0	0	Direct long-length	\bigcirc	0	×			
numbered address is used	Indirect word-length	×	\bigtriangleup	\bigtriangleup	Indirect word-length	×	\bigtriangleup	\bigtriangleup			
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup			

5 APPLICATION INSTRUCTIONS

MAX	MAXIMUN	Λ									
Function	Compares	the conte	nts of the	source wi	ith those of the result parameters	e destinat	tion, and s	stores the			
Parameters and processing	MAX S: Source R: Result D: Destina	$ \begin{array}{c} \bigcirc & \text{MAX S, D, R} \\ \text{S: Source} \\ \text{R: Result} \\ \text{D: Destination} \end{array} \end{array} \qquad \begin{array}{c} \text{When } S \geqq D: S \rightarrow R \\ \text{When } S \le D: D \rightarrow R \end{array} $									
Flags	The setting	s of the H	E and V f	lags chang	ge. The other	r flags ar	e turned o	ff.			
Processing time	0.42 ms										
Remarks											
Examples	↓ мах ↓ мах	 MAX FW000, FW001, FW002 FW000 0456 FW001 0004 O456 MAX MAX [DW000], 2, @ [H180000] DW000 8765 DW001 4321 H180000 0000 MAX 									
Valid parameters		Bit-type	Word-type	Orrestant		Bit-type	Word-type	Ormstant			
purumeters	S, D	PI/O	PI/O	Constant	R	PI/O	PI/O	Constant			
\triangle : A parameter	Direct word-length	\bigcirc	0	0	Direct word-length	0	0	×			
when an odd-	Direct long-length	$\begin{array}{c c c c c c c c c c c c c c c c c c c $									
numbered	Indirect word-length	$\begin{array}{c c c c c c c c c c c c c c c c c c c $									
address is used.	Indirect long-length	×	\triangle	\triangle	Indirect long-length	×	\triangle	\bigtriangleup			

MIN	MINIMUM									
Function	Compares smaller val	the conter ue in the	nts of the area spec	source wa	ith those of th he result parar	e destinat neter.	tion, and s	stores the		
Parameters and processing	OMIN SS: SourceR: ResultD: Destina	S, D, R tion	VV	When S ≦ When S >	$\begin{array}{ccc} D:S \rightarrow R\\ D:D \rightarrow R \end{array}$					
Flags	The setting	The settings of the E and V flags change. The other flags are turned off.								
Processing time	0.42 ms									
Remarks										
Examples	() MIN F F F MIN [I H	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$								
Valid parameters	<u>е</u> п	Bit-type	Word-type	Constant		Bit-type	Word-type	Constant		
1	3, D	PI/O	PI/O	Constant		PI/O	PI/O	Constant		
\triangle : A parameter	Word-length	\bigcirc	0	\bigcirc	Direct word-length	\bigcirc	\bigcirc	\times		
error is detected when an odd-	Direct long-length	irect O O O O O O O O O O O O O O O O O O O								
numbered address is used	Indirect word-length	$ \begin{array}{c c} \operatorname{irect} \\ \operatorname{rd-length} \end{array} \times & \bigtriangleup & \bigtriangleup & & \operatorname{Indirect} \\ \operatorname{word-length} \end{array} \times & \bigtriangleup & \bigtriangleup & \bigtriangleup & \end{array} $								
	Indirect long-length	×	\bigtriangleup	\bigtriangleup	Indirect long-length	×	\bigtriangleup	\bigtriangleup		

CLR	CLEAR									
Function	Clears a specified I/O area. TCLR, UCLR, and CCLR also clear the respective measured value areas.									
Parameters and processing	Name S Name: CLR application instruction name									
	S :	Source (To	p of the specified I/O area.)							
Flags	All fla	gs are set to	0.							
Processing time	See the	e table below	ν.							
Remarks										
Explanation				1						
		Name	Function	Processing time						
		XCLR	Clears X000 to XFFF.	0.92 ms						
		YCLR	Clears Y000 to YFFF.	0.92 ms						
		GCLRClears G000 to GFFF.0.92 ms								
		RCLR	Clears R000 to RFFF.	0.92 ms						
		KCLR	Clears K000 to KFFF.	0.92 ms						
		TCLR	Clears T000 to T3FF and also clears the measured area of T.	2.46 ms						
		UCLR	Clears U000 to U3FF and also clears the measured area of U.	1.70 ms						
		CCLR	Clears C000 to C3FF and also clears the measured area of C.	1.70 ms						
		VCLR	Clears V000 to VFFF.	0.92 ms						
		ECLR	Clears E000 to EFFF.	0.92 ms.						
		FCLR	Clears S020 to S027.	0.15 ms						
		JCLR	Clears J000 to JFFF.	0.92 ms						
		QCLR Clears Q000 to QFFF. 0.92 ms								
		HHCLRClears HH000 to HH1FF.1.70 ms								
Examples										
	¢н	HCLR HH	000							

SUPPLEMENT

Supplement 1 Work Flow Based on HI-FLOW Program

A HI-FLOW program is created using a programming tool such as a personal computer and executed by the PCs. When the execution result is to be monitored, minimum necessary data is received from the PCs, and it is combined with the program in the tool and output. This aims at increasing the monitor speed by decreasing the amount of communication.

The storage media such as FD and HD are also used to save and load the created program.



Supplement 2 PCs Memory

The HI-FLOW program to be executed on the PCs are placed in the area shown below. The program is actually stored in the memory of the PC, as shown in the memory map below.



ascending order.

The processes are not always arranged in

Supplement 3 Online Mode

"Offline" means that the text to be edited is made into a program of the programming tool regardless of the contents of PCs memory.

"Online" means that the text to be edited or monitored is made into a program for the PCs. When the PCs are placed under monitoring, not all necessary data has been read from the PCs since it takes a long time to do so by communication. For this reason, it is necessary that the program of the tool matches that of the PCs. This matching is enabled by transmission or reception.

A HI-FLOW program is self-contained in a process. Therefore, if only one process contains a matching program, the process can be subjected to editing or monitoring. Transmission and reception of all processes or just one process are available for saving time and many other purposes.

(1) All-processes transmission

The following shows the data flow when transmitting all HI-FLOW programs on the tool to the PCs at one time.



After all the processes have been transmitted, the processes and tables in memory are arranged in ascending order.

(2) All-processes reception

The following shows the data flow when receiving all HI-FLOW programs from the PCs with the tool.



On reception, the processes and tables in the memory are not always arranged in ascending order.

(3) One-process transmission

The following shows the data flow when transmitting one specific HI-FLOW process on the tool to the PCs.



After the process has been transmitted, it is placed after all other processes in the memory.

(4) One-process reception

The following shows the data flow when receiving one specific HI-FLOW process on the PCs with the tool.



On reception, the processes and tables on the memory are not always arranged in ascending order.

Supplement 4 Check for Progress

HI-FLOW indicates the current position on the user program with the monitor cursor. The HI-FLOW system in the PCs controls the progress of the current position. This section describes how the user program transferred to the PCs is checked for progress by the PCs.

Item	Description
Basic rules	 The progress is checked at each by the PCs. Processes activated with ACT are checked sequentially in ascending order of process numbers. In the same process, routes are checked sequentially in ascending order of route numbers. (The route numbers on the screen are increased from left to right and from top to bottom.) In the same route, steps are checked sequentially in ascending order of step numbers. Upon completion of checking a step, the next step is checked for progress. If the program cannot proceed to the next step, the route having the next route number is checked for process. At the next scan, progress check for this process and route starts with this step.
Called process	• The called process is checked for progress after progress check for the calling process and calling route is done. When progress check for the called process is completed but process execution is not completed, the next route after the calling process and calling route is checked. When process execution is completed, the next step after the calling route is checked.
Process control	 The process called with ACT is checked for progress at the time of activation. When processes have smaller process numbers than the process number of the process called with ACT, these processes are checked for progress at the next scan. When the process numbers are larger, these processes are checked at the same scan as for the process. RST, STP, and CLR in Control Box and Process Start are processed at they are specified.
Around-the-clock monitoring	 The condition to start a process (ACT, STP, RST, or CLR) and multi-entry condition are checked before the process is checked for progress. When the condition is satisfied, the processing is performed before process progress check. Y-output interlock conditions are checked at each scan and Y-output is turned on or off before the first process is checked for progress.

Item	Description
Branch	• After a branch step (If or Jump) is executed, the destination step is checked for progress. Therefore, the route being executed may not be checked for progress for one scan or it may be checked twice. Also, a closed loop without progress conditions may be executed infinitely.
Repetition	• After Repeat End, Repeat Start is checked for progress. Therefore, repetition without progress conditions may result in an infinite loop.
Forcible termination	• When Escape is executed, the next process is checked for progress. When the process is a called process, the next step after the calling process and calling route is checked for progress.
Synchronization	 After execution of Parallel Start, the next step is checked for progress. Parallel End or Route End checks the next step after Parallel End for the joined route when all synchronous routes are terminated. When some synchronous routes are not terminated, Parallel End or Route End stops the local route and checks the next route for progress.
Selection	 After execution of Select, the next step is checked for progress. When the condition for Wait in Selective Branching is satisfied, the other selected routes are stopped and the next step is checked for progress. When the condition is not satisfied, the next route is checked. Select End or Route End checks the next step after the Select End for the joined route. If the joined route is stopped, Select End or Route End activates it.
Wait for condition	 When the condition is satisfied, Wait checks the next step for progress. When the condition is not satisfied, Wait checks the route having the next number. At the next scan, progress check for this process and route starts with this step. Wait in Selective Branching checks whether the previous step is an ON statement before proceeding to the next step. If so, Wait in Selective Branching clears to 0 the ON statement.
Symbol with no delay	• This symbol enables the program to proceed to the next step with no delay in any case. The following steps proceed to the appropriate step with no delay: Process Start, Route Start, Parallel Start, Select, Multi-entry, Box, Control Box, Function, Process End at termination of a called process, Route End and Parallel End at synchronization end, Route End and Select End at selective joining, and branch steps (Repeat Step, Repeat End, If, and Jump).

Supplement 5 Relationships between a HI-FLOW Program and the CPU Load

A HI-FLOW program runs on a PCs as part of the operating system. As the amount of the HI-FLOW program increases, therefore, the load of the operating system in the PCs increases. As a result, the following problems arise:

- More sequence cycle time than the specified value is required.
- The states of LEDs on the PCs cannot change.
- When the PCs is reset, LEDs do not light up.

If the load is further increased, the entire system dose not rum correctly, for example, the sequence cycle stops. How to create an efficient HI-FLOW program is explained as well as a measure to judge the load.

How to create an efficient HI-FLOW program

- 1. The load of the HI-FLOW program is determined by the number of steps being executed. It is not affected by the vertical (route) length of the HI-FLOW program. Therefore, the load of a program consisting of too many processes and routes is high.
- 2. Do not make unnecessary loops.

Do not make loops that are not required or have no stop points.



3. Use timers so that their numbers are specified consecutively in ascending order. Wait timers, parallel timers, and counters having lower numbers have lower load.

4. In the same route, use wait timers having the same number.

In the same route, multiple timers are never used at the same time. Use wait timers having the same number. Do not use wait timers having a larger number as far as possible.

5. Use a minimum number of called processes.

A program divided into subroutines is more understandable. During execution, however, the load is higher compared when called processes are not used. Carefully determine a program structure.

6. Do not use Control Box steps consecutively, if possible.

The execution load of the Control Box step is very high. Do not use Control Box steps consecutively as far as possible. If it is unavoidable to use them in such a way, use consecutive specification of processes in an efficient way.



7. Set a minimum number of system control bits.

System control bits (see utilities in the operation manual) must be checked in each sequence cycle or during execution of each step, increasing the load. Set a minimum number of system control bits.

8. Use a minimum number of Multi-entry steps.

Multi-entry steps must be checked in each sequence cycle. When too many Multi-entry steps are used, the load becomes high. Use a minimum number of Multi-entry steps.

9. Take care so that the Multi-entry step does not enter an in-loop.

The Multi-entry step checks condition expressions in each cycle. If a condition is satisfied, Multi-entry starts execution from the step. If consecutive conditions rather than an edge condition are satisfied, Multi-entry enters an infinite loop. Be sure to edge-trigger for conditions Multi-entry.



If X001 is not satisfied and X000 is left on in the program on the left, steps 1 to 4 are executed in each sequence cycle. To prevent this, set an edge condition for X000.

10. Set STP or RST in the Process Start step a minimum number of times.

The Process Start step with STP or RST specified checks conditions in each sequence cycle, increasing the load significantly. Set STP or RST a minimum number of times.

11. Take care when setting CLR in the Process Start step.

The Process Start step with CLR specified clears PI/O each time a condition is satisfied, increasing the load. (The Process Start step with RST, STP, or ACT specified does not check further conditions once a condition is satisfied.) Create conditions to be checked by the Process Start step with CLR specified.

12. Do not use consecutive application instructions if possible.

Application instructions perform operation continuously. If they are written consecutively, the sequence cycle may be extended. When writing application instructions, take care.

13. Do not use complex condition expressions if possible.

When complex condition expressions are used in a HI-FLOW program, analysis of them takes more time than a ladder program. When using complex condition expressions, write them in a ladder program then pass them to the HI-FLOW program.

Correlation between a HI-FLOW program and the CPU load

The following tables gives measures of times of sequence cycle during which processes can be executed on the S10/2 α , S10/2 α E, and S10/2 α H(f) for individual process counts. The conditions below are assumed:

(1) The following processes are used. Conditions are not satisfied. Other programs including a ladder program are not used.



- (2) All system control bits are invalid.
- (3) When the PCs are turned on with the CPU key switch set to RUN, the actual sequence cycle time is measured. (The sequence cycle accumulation counter in SW140 is measured for a certain time.)

[S10/2α]

: These process can be executed in the specified time.

: When these processes are set, the sequence cycle stops.

Specified sequence	Number of processes to be executed (n)						
cycle time (ms)	10	30	50	100	150	200	256
10							
20							
30							
50							
75							
100							
125							
150							
175							
200							

SUPPLEMENT

 $[S10/2\alpha E]$: These processes can be executed in the specified time.

Specified sequence	Number of processes to be executed (n)						
cycle time (ms)	10	30	50	100	150	200	256
10							
20							
30							
50							
75							
100							
125							
150							
175							
200							

[S10/2αH(f)]

: These processes can be executed in the specified time.

Specified sequence	Number of processes to be executed (n)						
cycle time (ms)	10	30	50	100	150	200	256
10							
20							
30							
50							
75							
100							
125							
150							
175							
200							

When actually designing the system, set the sequence cycle time based on the above table, taking the safety factor into account. Considering the ladder operation functions and the existence of C mode programs, about halt of the sequence cycle time in which program execution is possible on the PCs is appropriate.

INDEX

А

С

about the process	10	call	54
ABS	96	CCLR	112
ACT	32, 42	CLR	32, 44, 112
ADD	63	condition expression	35
AND	71	configuration of HI-FLOW program	ns 2
APB	94	constant	24
application instruction	58	control box	42
application instruction,			
explanation of functions	62	D	
application instruction, overview	58		
application instruction, parameters	58	DCD	98
application instruction,		DEC	66
system error flags	61	DIV	68
application instruction, type conversion		DTB	90
for operation	60		
application instruction, usage	58	E	
application instructions	6		
ASL	103	ECD	99
ASP	92	ECLR	112
ASR	102	EOR	73
assignment expression	37	EQU	75
AST	87	Escape	50
ASU	93	EXC	84
AUB	95	explanation of syntaxes	30

В

BND box BTD

F

107	FCLR	112
37	free comment	27
89	free label	27
	function	55

INDEX

G		М	
GCLR	112	MAX	110
GE	78	MIN	111
GT	77	MOD	69
		MOM	83
Н		MOV	82
		MUL	67
HHCLR	112	Multi-entry	53
how to use this manual	4		
		Ν	
Ι			
		NEG	97
If	47	NEQ	76
INC	65	NOT	74
J		О	
JCLR	112	OFF statement	39
Jump	49	ON statement	38
		operator	26
K		OR	72
		outline of the syntax	5
KCLR	112	output bit	35
		overview	4
L			
		Р	
label	23		
LE	80	Parallel Start, Parallel End	51
LIM	106	parallel timer	39
LSL	101	POP	86
LSR	100	Process Start, Process End	30
LT	79	process	10
		process information	28
		process information, comment	28
		process information, name	28

INDEX

process state	11	symbol	20
process state change	12	syntax	23
program	15		
PSH	85	Т	
Q		TCLR	112
		timer	35
QCLR	112	TRS (Timer Reset)	41
		TST	81
R		TUP (Timer Up)	40
DCID	112		
RCLR	112		
relationships between the setting of		U	
the key switch on the PCs and			
the states of processes	14	UCLR	112
Repeat Start, Repeat End	46	use of both the synchronization syntax and	
reserved word	24	selection syntax	16
ROL	105		
ROR	104	V	
ROT	109		
Route Start, Route End	34	variable	24
route	15	VCLR	112
RST	31, 43		

S

88
70
91
52
19
26
20
30, 44
64

Ţ	wait	35
,	wait timer	35
	Wait with Previous State Cleared	55
	When different routes are used as	
í	a route at which a branch starts and	
í	a route at which routes are joined	17
	When the same route is used as	
í	a route at which a branch starts and	
ť	a route at which routes are joined	16

W

	Х	
XCLR		112
	Y	
YCLR		112
	Ζ	
ZON		108