

29 | Ladder Instructions

This chapter describes GP-Pro EX logic instructions. Instructions that can be used in logic programs are described in detail.

29.1	Instructions.....	29-2
29.2	Instruction Notation List.....	29-6
29.3	About Addresses You can Set up as Operands.....	29-34
29.4	Number of Steps Per Instruction	29-42
29.5	Instruction Descriptions.....	29-43

29.1 Instructions

The following table provides a list of instructions available for the logic program. Models that support logic can use all of these instructions. The instructions are divided into the following nine categories: (1) Basic, (2) Timer, (3) Counter, (4) Read/Write, (5) Operation, (6) Function, (7) Comparison, (8) Conversion, (9) I/O Driver Instructions.

Category		Instruction Name	Instruction
Basic	Bit Basics	Normally Open	NO
		Normally Closed	NC
		Out	OUT
		Negative Out	OUTN
		Set	SET
		Reset	RST
	Pulse Basic	Positive Transition	PT
		Down Transition	NT
	Program Control	Jump	JMP
		Jump to Subroutine	JSR
		Return	RET
		Repeat Processing	FOR
			NEXT
		Inverse	INV
		Exit	EXIT
		Power Bar Control	PBC
		Power Bar Reset	PBR
	Logic Wait	LWA	
Timer Instruction	—	On Delay Timer	TON
		Off Delay Timer	TOF
		Pulse Timer	TP
		Duration On Delay Timer	TONA
		Duration Off Delay Timer	TOFA
Counter Instruction	—	Up Counter	CTU
		Down Counter	CTD
		Up/Down Counter	CTUD
R/W Instruction	Time Read/Write	Read Time	JRD
		Set Time	JSET
	Date Read/Write	Read Date	NRD
		Set Date	NSET

Continued

Category		Instruction Name	Instruction
Operation Instruction	Arithmetic Operation	Add	ADD
		Subtract	SUB
		Multiplication	MUL
		Division	DIV
		Modulation	MOD
		Increment	INC
		Decrement	DEC
	Time Operation	Time Addition	JADD
		Time Subtraction	JSUB
	Logical Operation	Logical AND	AND
		Logical OR	OR
		Logical XOR	XOR
		Logical NOT	NOT
	Transfer	Transfer (Copy)	MOV
		Block Transfer (Block Copy)	BLMV
		Multipoint Transfer (Multipoint Copy)	FLMV
		Exchange	XCH
	Shift	Shift Left	SHL
		Shift Right	SHR
		Arithmetic Shift Left	SAL
		Arithmetic Shift Right	SAR
	Rotation	Rotate Left	ROL
		Rotate Right	ROR
		Rotate Left with Carry Over	RCL
		Rotate Right with Carry Over	RCR

Continued

Category		Instruction Name	Instruction
Function Instruction	Calculation Function	Sum	SUM
		Average	AVE
		Square Root	SQRT
		Bit Count	BCNT
		PID	PID
	Trigonometric Function	Sine	SIN
		Cosine	COS
		Tangent	TAN
		Arc Sine	ASIN
		Arc Cosine	ACOS
		Arc Tangent	ATAN
		Cotangent	COT
	Other Function	Exponent	EXP
		Logarithm	LN
		Log Base 10	LG10
Compare Instruction	Arithmetic Compare	Comparison(=)	EQ
		Comparison(>)	GT
		Comparison(<)	LT
		Comparison(>=)	GE
		Comparison(<=)	LE
		Comparison(≠)	NE
	Time Compare	Time Compare(=)	JEQ
		Time Compare(>)	JGT
		Time Compare(<)	JLT
		Time Compare(>=)	JGE
		Time Compare(<=)	JLE
		Time Compare(≠)	JNE
	Date Compare	Date Compare(=)	NEQ
		Date Compare(>)	NGT
		Date Compare(<)	NLT
		Date Compare(>=)	NGE
		Date Compare(<=)	NLE
		Date Compare(≠)	NNE

Continued

Category		Instruction Name	Instruction
Convert Instruction	Numeric Value	BCD Convert	BCD
		BIN Convert	BIN
		Encode	ENCO
		Decode	DECO
		Convert to Radian	RAD
		Convert Degree	DEG
		Scale	SCL
	Type	Integer → Float Conversion	I2F
		Integer → Real Conversion	I2R
		Float → Integer Conversion	F2I
		Float → Real Conversion	F2R
		Real → Integer Conversion	R2I
		Real → Float Conversion	R2F
		Convert to Seconds	H2S
Convert Seconds to Time	S2H		
I/O Driver Instructions	CAN	SDO Read	SDOR
		SDO Write	SDOW
		Master Check	DGMT
		Slave Check	DGSL

29.2 Instruction Notation List

This list shows the categorized instruction names and symbols.

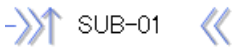





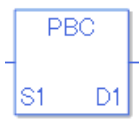

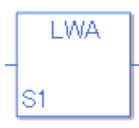
IMPORTANT

- The number of steps in each instruction depends on the data format of operands and whether a modifier is used.
- For details about the number of steps, refer to the section that describes each instruction.

29.2.1 Basic Instructions

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Ladder Symbol	
Basic Instructions	Bit Basic	Normally Open	NO	1 to 5 steps	1	
		Normally Closed	NC	1 to 5 steps	1	
		Out	OUT	1 to 5 steps	1	
		Negative Out	OUTN	1 to 5 steps	1	
		Set	SET	1 to 5 steps	1	
		Reset	RST	1 to 5 steps	1	
	Pulse Basic	Positive Transition	PT	2 to 5 steps	1	
		Down Transition	NT	2 to 5 steps	1	
	Program Control	Jump	JMP	2 Step	—	
		Positive Transition Jump	JMPP	2 to 5 steps	—	
Jump to Subroutine		JSR	2 Step	—		

Continued




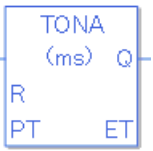
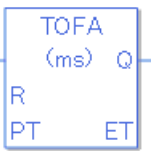
Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Ladder Symbol	
Basic Instructions	Positive Transition Subroutine Load	JSRP	2 Step	—		
	Return	RET	1 Step	—		
	Repeat Processing	FOR	2 to 4 steps	1	1	
		NEXT	1 Step	—	—	
	Inverse	INV	1 Step	—		
	Exit	EXIT	1 Step	—		
	Power Bar Control	PBC	3 Step	2	2	
		PBR	2 Step	1	1	
	Logic Wait Instruction	LWA	2 Step	1	1	

(Note)

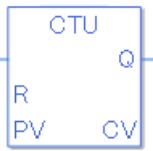
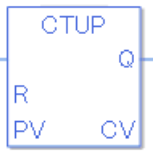
To use 1 step, the number of steps must be fewer than the clear bit variables (M address) + 1536. If more than 1536 bit variables are created with clear bit variable settings, there will be 2 steps.

Please configure retentive/volatile settings in the Retentive Settings dialog box.

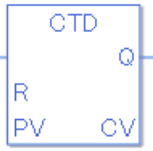
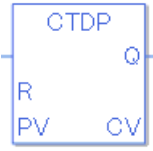
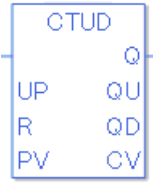
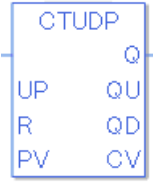
29.2.2 Timer Instruction

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Ladder Symbol
Timer Instruction	On Delay Timer	TON	2 Step	1	
	Off Delay Timer	TOF	2 Step	1	
	Pulse Timer	TP	2 Step	1	
	Duration On Delay Timer	TONA	2 Step	1	
	Duration Off Delay Timer	TOFA	2 Step	1	









29.2.3 Counter Instruction

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Counter Instruction	Up Counter	CTU	2 Step	1	Level	
		CTUP	2 Step	1	Up Edge	

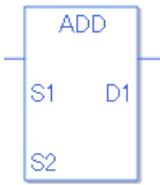

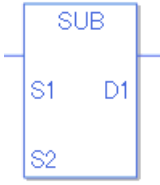
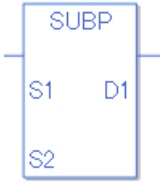
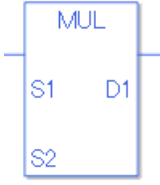
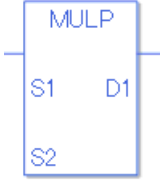
Continued

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Counter Instruction	Down Counter	CTD	2 Step	1	Level	
		CTDP	2 Step	1	Up Edge	
	Up/Down Counter	CTUD	2 Step	1	Level	
		CTUDP	2 Step	1	Up Edge	

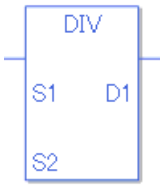
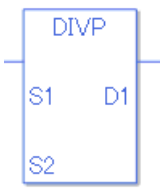
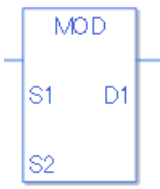
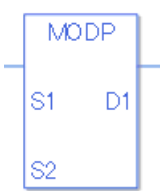




29.2.4 R/W Instructions

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
R/W Instruction	Time Read/Write	Time Read	JRD	6 Step	1	Level	
			JRDP	6 Step	1	Up Edge	
		Time Set	JSET	3 Step	2	Level	
			JSETP	3 Step	2	Up Edge	
R/W Instruction	Date Read/Write	Date Read	NRD	5 Step	1	Level	
			NRDP	5 Step	1	Up Edge	
		Date Set	NSET	3 Step	2	Level	
			NSETP	3 Step	2	Up Edge	

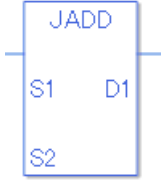
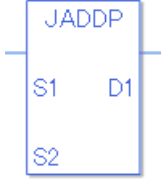
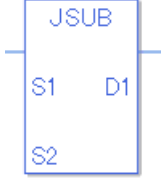
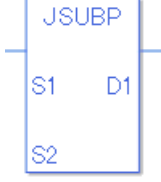
29.2.5 Arithmetic Operation

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Operation	Add	ADD	4 to 13 steps	3	Level	
		ADDP	4 to 13 steps	3	Up Edge	
	Subtract	SUB	4 to 13 steps	3	Level	
		SUBP	4 to 13 steps	3	Up Edge	
	Multiplication	MUL	4 to 13 steps	3	Level	
		MULP	4 to 13 steps	3	Up Edge	


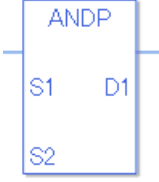
Continued

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol	
Operation	Arithmetic	Division	DIV	4 to 13 steps	3	Level	
			DIVP	4 to 13 steps	3	Up Edge	
		Modulation	MOD	4 to 13 steps	3	Level	
			MODP	4 to 13 steps	3	Up Edge	
		Increment	INC	2 to 4 steps	1	Level	
			INCP	2 to 4 steps	1	Up Edge	
	Decrement	DEC	2 to 4 steps	1	Level		
		DECP	2 to 4 steps	1	Up Edge		

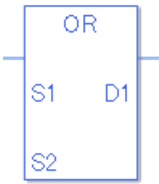
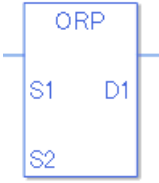




29.2.6 Time Operation

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol
Operation	Time Operation	Time Addition	JADD	4 Step	3	Level	
			JADDP	4 Step	3	Up Edge	
		Time Subtraction	JSUB	4 Step	3	Level	
			JSUBP	4 Step	3	Up Edge	







29.2.7 Logical Operation

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Operation	Logical	Logical AND	AND	4 to 13 steps	3	Level	
			ANDP	4 to 13 steps	3	Up Edge	

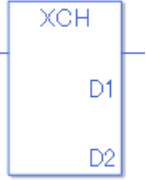
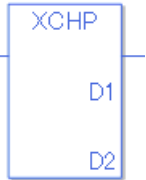
Continued

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Operation	Logical	Logical OR	OR	4 to 13 steps	3	Level	
			ORP	4 to 13 steps	3	Up Edge	
		Logical XOR	XOR	4 to 13 steps	3	Level	
			XORP	4 to 13 steps	3	Up Edge	
		Logical NOT	NOT	3 to 9 steps	2	Level	
			NOTP	3 to 9 steps	2	Up Edge	

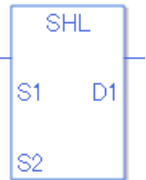
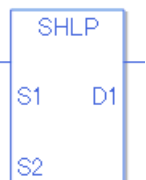
29.2.8 Transfer Instructions

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Operation	Transfer	Transfer (Copy)	MOV	3 to 9 steps	2	Level	
			MOVP	3 to 9 steps	2	Up Edge	
		Batch Transfer (Blocks Copy)	BLMV	6 to 10 steps	3	Level	
			BLMVP	6 to 10 steps	3	Up Edge	
		Multipoint Transfer (Multipoint Copy)	FLMV	4 to 10 steps	3	Level	
			FLMVP	4 to 10 steps	3	Up Edge	

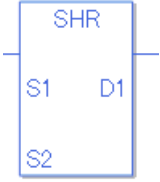
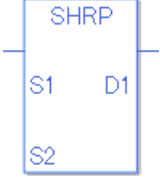
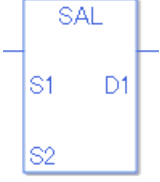
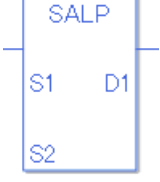
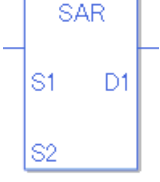
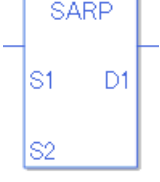
Continued

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Operation	Transfer	Exchange	XCH	3 to 7 steps	2	Level	
			XCHP	3 to 7 steps	2	Up Edge	

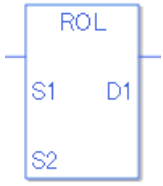
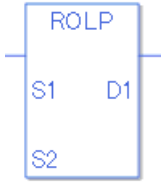
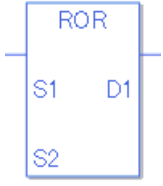
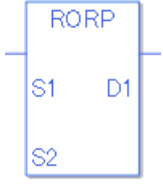
29.2.9 Shift Instructions

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Operation	Shift	Shift Left	SHL	4 to 10 steps	3	Level	
			SHLP	4 to 10 steps	3	Up Edge	

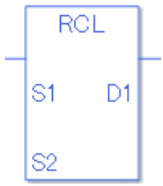
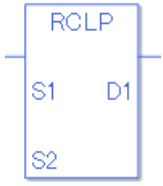
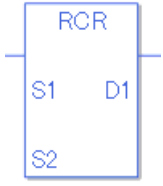
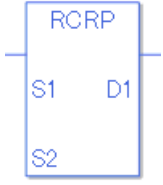
Continued

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Operation	Shift Right	SHR	4 to 10 steps	3	Level	
		SHRP	4 to 10 steps	3	Up Edge	
	Arithmetic Shift Left	SAL	4 to 10 steps	3	Level	
		SALP	4 to 10 steps	3	Up Edge	
	Arithmetic Shift Right	SAR	4 to 10 steps	3	Level	
		SARP	4 to 10 steps	3	Up Edge	







29.2.10 Rotation Instructions

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Operation	Rotation	Rotate Left	ROL	4 to 10 steps	3	Level	
			ROLP	4 to 10 steps	3	Up Edge	
		Rotate Right	ROR	4 to 10 steps	3	Level	
			RORP	4 to 10 steps	3	Up Edge	



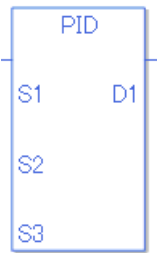
Continued

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Operation	Rotation	Rotate Left with Carry Over	RCL	4 to 10 steps	3	Level	
			RCLP	4 to 10 steps	3	Up Edge	
		Rotate Right with Carry Over	RCR	4 to 10 steps	3	Level	
			RCRP	4 to 10 steps	3	Up Edge	









29.2.11 Function Operations

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol	
Function	Calculate Function	Sum	SUM	4 to 10 steps	3	Level	
			SUMP	4 to 10 steps	3	Up Edge	
		Average	AVE	4 to 10 steps	3	Level	
			AVEP	4 to 10 steps	3	Up Edge	
		Square Root	SQRT	3 to 7 steps	2	Level	
			SQRTP	3 to 7 steps	2	Up Edge	







Continued

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Function	Calculate Function	Bit Count	BCNT	3 to 9 steps	2	Level	
			BCNTP	3 to 9 steps	2	Up Edge	
		PID	PID	10 to 18 steps	5	Level	







29.2.12 Trigonometric Function

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Function	Sine	SIN	3 to 7 steps	2	Level	
		SINP	3 to 7 steps	2	Up Edge	
	Cosine	COS	3 to 7 steps	2	Level	
		COSP	3 to 7 steps	2	Up Edge	
	Tangent	TAN	3 to 7 steps	2	Level	
		TANP	3 to 7 steps	2	Up Edge	
	Arc Sine	ASIN	3 to 7 steps	2	Level	
		ASINP	3 to 7 steps	2	Up Edge	

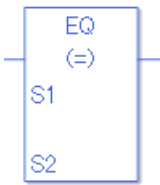
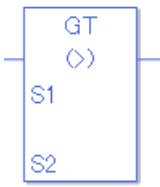
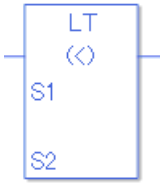
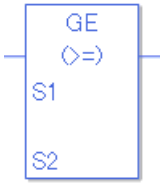
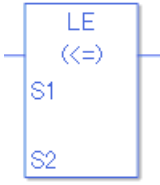
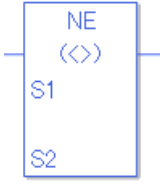
Continued

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Function	Trigonometric	Arc Cosine	ACOS	3 to 7 steps	2	Level	
			ACOSP	3 to 7 steps	2	Up Edge	
		Arc Tangent	ATAN	3 to 7 steps	2	Level	
			ATANP	3 to 7 steps	2	Up Edge	
		Cotangent	COT	3 to 7 steps	2	Level	
			COTP	3 to 7 steps	2	Up Edge	

29.2.13 Other Functions

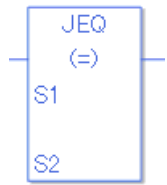
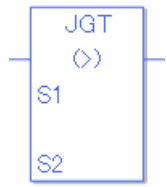
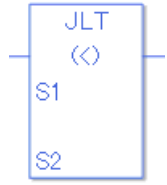
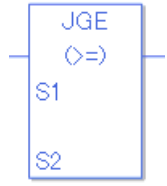
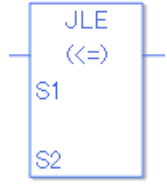
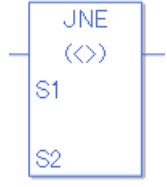
Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Chart
Function	Miscellaneous function	Exponential	EXP	3 to 7 steps	2	Level	
			EXPP	3 to 7 steps	2	Up Edge	
		Logarithm	LN	3 to 7 steps	2	Level	
			LNP	3 to 7 steps	2	Up Edge	
		Log Base 10	LG10	3 to 7 steps	2	Level	
			LG10P	3 to 7 steps	2	Up Edge	

29.2.14 Arithmetic Compare

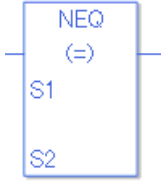
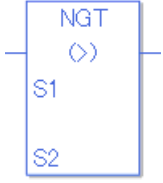
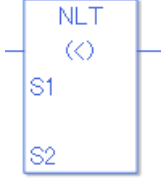
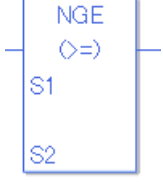
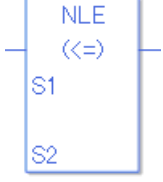
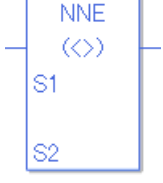
Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Compare Arithmetic Compare	Comparison (=)	EQ	3 to 9 steps	2	Level	
	Comparison (>)	GT	3 to 9 steps	2	Level	
	Comparison (<)	LT	3 to 9 steps	2	Level	
	Comparison (>=)	GE	3 to 9 steps	2	Level	
	Comparison (<=)	LE	3 to 9 steps	2	Level	
	Comparison (<>)	NE	3 to 9 steps	2	Level	

Continued










29.2.15 Time Compare

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol	
Compare	Time Compare	Time Compare (=)	JEQ	3 Step	2	Level	
		Time Compare (>)	JGT	3 Step	2	Level	
		Time Compare (<)	JLT	3 Step	2	Level	
		Time Compare (>=)	JGE	3 Step	2	Level	
		Time Compare (<=)	JLE	3 Step	2	Level	
		Time Compare (<>)	JNE	3 Step	2	Level	






29.2.16 Date Compare

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol	
Compare	Date Compare	Date Compare (=)	NEQ	3 Step	2	Level	
		Date Compare (>)	NGT	3 Step	2	Level	
		Date Compare (<)	NLT	3 Step	2	Level	
		Date Compare (>=)	NGE	3 Step	2	Level	
		Date Compare (<=)	NLE	3 Step	2	Level	
		Date Compare (<>)	NNE	3 Step	2	Level	







29.2.17 Data Conversion Instructions

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Convert	BCD Convert	BCD	3 to 7 steps	2	Level	
		BCDP	3 to 7 steps	2	Up Edge	
	BIN Convert	BIN	3 to 7 steps	2	Level	
		BINP	3 to 7 steps	2	Up Edge	
	Encode	ENCO	3 to 7 steps	2	Level	
		ENCOP	3 to 7 steps	2	Up Edge	
	Decode	DECO	3 to 7 steps	2	Level	
		DECOP	3 to 7 steps	2	Up Edge	
	Convert to Radian	RAD	3 to 7 steps	2	Level	











Continued

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Convert	Data Convert	Convert to Radian	RADP	3 to 7 steps	2	Up Edge	
		Convert Degree	DEG	3 to 7 steps	2	Level	
			DEGP	3 to 7 steps	2	Up Edge	
		Scale	SCL	7 to 11 steps	2	Level	
			SCLP	7 to 11 steps	2	Up Edge	

29.2.18 Type Conversion Instructions

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Convert	Type Convert	Integer → Float	I2F	3 to 7 steps	2	Level	
			I2FP	3 to 7 steps	2	Up Edge	
		Integer → Real	I2R	3 to 7 steps	2	Level	
			I2RP	3 to 7 steps	2	Up Edge	
		Float → Integer	F2I	3 to 7 steps	2	Level	
			F2IP	3 to 7 steps	2	Up Edge	

Continued

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Determination of Input	Ladder Symbol
Convert	Float → Real	F2R	3 to 7 steps	2	Level	
		F2RP	3 to 7 steps	2	Up Edge	
	Real → Integer	R2I	3 to 7 steps	2	Level	
		R2IP	3 to 7 steps	2	Up Edge	
	Real → Float	R2F	3 to 7 steps	2	Level	
		R2FP	3 to 7 steps	2	Up Edge	
	Convert to Seconds	H2S	3 to 5 steps	2	Level	
		H2SP	3 to 5 steps	2	Up Edge	
	Convert Seconds to Time	S2H	3 to 5 steps	2	Level	
		S2HP	3 to 5 steps	2	Up Edge	

29.2.19 I/O Driver

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in Instruction	Number of Operands	Ladder Symbol	
I/O Driver Instructions	CAN	SDO Read	SDOR	9 to 21 steps	6	
		SDO Write	SDOW	9 to 21 steps	6	
		Master Check	DGMT	7 to 15 steps	4	
		Slave Check	DGSL	5 to 9 steps	2	

29.3 About Addresses You can Set up as Operands

Outlines the symbol variables, connection device addresses, and constants that you can set as operands in each instruction.

Because the content that you can configure differs depending on the instruction, refer to each instruction description.

29.3.1 Connection Device Address

The address specified in the communication settings for a connection device.

Name	Type	Example	Description
Connection Device	Bit	[PLC1]X0000	The bit address for the communication device address specified in the communication settings
	Word	[PLC1]D0000	The word address for the connection device address specified in the communication settings

29.3.2 Symbol

This function changes addresses in external devices into names that users can easily understand. Make sure you map external device addresses to their respective names.

For example, To assign the name "RUN" to device address "X0000" on a Mitsubishi Electric Corporation PLC, define "RUN" and "X0000."

Name	Type	Example	Description
Symbol	Bit	RUN = X0000	This is a bit symbol configured in the symbol variables list and defined by the connection device address and the arbitrary name.
	Word	Data = D0000	It is a word symbol configured in the symbol variables list and defined by the connection device address and the arbitrary name.

29.3.3 LS Address

This is the address of an internal memory area on a GP unit. Please note that how you specify the address varies depending on the communication settings.

Name	Type	Example	Description
Internal Memory	Bit	[#INTERNAL]LS010000	Bit Specifications for GP Internal Memory
	Word	[#INTERNAL]LS0100	Word Specifications for GP Internal Memory
Memory link settings	Bit	[#MEMLINK]010000	Bit Specifications for GP Internal Memory
	Word	[#MEMLINK]0100	Word Specifications for GP Internal Memory

NOTE

- LS area and USR area words are 16-bits, but are treated as 32-bits when processed by the Logic. However, the top 16 bits are ignored when used in a display and other parts.

29.3.4 USR area

This is the internal memory area of a GP unit. Any specification method can be used. Addressing from 0-29999 is available.

Name	Type	Example	Description
USR area	Bit	[#INTERNAL]USR00 10000	Bit Specifications for GP Internal Memory
	Word	[#INTERNAL]USR00 100	Word Specifications for GP Internal Memory

NOTE

- LS area and USR area words are 16-bits, but are treated as 32-bits when processed by the Logic. However, the top 16 bits are ignored when used in a display and other parts.

29.3.5 System Variable

This is the system area of a GP unit. Any connection device settings can be used.

NOTE

- Some of the system variables used in logic programs work only when the logic program is enabled. Attention must be paid when [Disable] is selected for the logic program or you are using #L**** logic variables.

Name	Type	Example	Description
System Variable	Bit	#L_Clock100ms	GP System Variable Bit Type
		#L_Clock1sec	
	Integer	#L_ScanTime	GP System Variable Integer Type

29.3.6 Variable

Variables are available for all GP models. You can use the variables without being aware of device addresses. Variables can be used with modifiers (*1) and as arrays (*2). When using modifiers, you can access individual bits or bytes in integer variables.

Name	Type	Example	Description
Variable	Bit	Arbitrary Name	Bit-type variable. Arrays allowed.
	Integer	Same as above	Integer-type variable. Arrays and modifier allowed.
	Float	Same as above	32-bit float variable. Arrays allowed.
	Real	Same as above	64-bit real variable. Arrays allowed.
	Timer	Same as above	Timer variable. Structure* ³ variable.
	Counter	Same as above	Counter variable. Structure* ³ variable.
	Date	Same as above	Date variable. Structure* ³ variable.
	Sampling	Same as above	Time variable. Structure* ³ variable
	PID	Same as above	PID variable. Structure* ³ variable.

- *1 You can use three different types of modifiers: bit modifier, byte modifier, and word modifier. Only integer variables support modifiers.
Specification method: bit → VariableName.X[0], byte → VariableName.B[0], word → VariableName.W[0]
- *2 You can specify consecutive memory addresses using arrays with the following variable types: bit, integer, float, and real.
Specification method: VariableName[10]
- *3 Multiple variables grouped together are structures. Structure variables include: Timer, Counter, Time, Date, and PID.

■ **Structure Variable**

Timer Variable

Timer Variable	Variables Settings	Description
VariableName.TI	Bit Variable	Turns ON when timer begins counting.
VariableName.Q	Bit Variable	Turns ON when the timer finishes counting.
VariableName.R	Bit Variable	Resets the current value on the timer. 0 clear.
VariableName.PT	Integer Variable	The value set on the timer.
VariableName.ET	Integer Variable	The current value on the timer.

Counter Variable

Counter Variable	Variables Settings	Description
VariableName.R	Bit Variable	Resets the current value. Clear (0).
VariableName.Q	Bit Variable	Turns ON when the current value reaches the preset value.
VariableName.UP	Bit Variable	Turns ON while counting up.
VariableName.QU	Bit Variable	For Up/Down counters, turns ON when the current value reaches the preset value.
VariableName.QD	Bit Variable	For Up/Down counters, turns ON when the current value reaches 0 or less.
VariableName.PV	Integer Variable	Counter setting value.
VariableName.CV	Integer Variable	Current value on the counter.

Time Variable

Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD.
VariableName.MIN	Integer Variable	Minutes are input in BCD.
VariableName.SEC	Integer Variable	Seconds are input in BCD.

Date Variable

Date Variable	Variables Settings	Description
VariableName.YR	Integer Variable	The year is input in BCD.
VariableName.MO	Integer Variable	The month is input in BCD.
VariableName.DAY	Integer Variable	The day is input in BCD.

PID Variable

PID Variable	Variables Settings	Description
VariableName.Q	Bit Variable	Completion Flag for PID Instruction Processing
VariableName.PF	Bit Variable	Processing Deadband Range Flag
VariableName.UO	Bit Variable	Output Values over the Upper Limit
VariableName.TO	Bit Variable	Output Values over the Lower Limit
VariableName.IF	Bit Variable	Integral Setting
VariableName.KP	Integer Variable	Proportional Constant
VariableName.TR	Integer Variable	Integral Calculus Time
VariableName.TD	Integer Variable	Differential Calculus Time a Time
VariableName.PA	Integer Variable	PID Processing Deadband Range
VariableName.BA	Integer Variable	Bias (Offset)
Variable name.ST	Integer Variable	Frequency in Sampling

■ When defining constants as operands

When entering constant values, enter them as shown in the following.

Real constant	Use this when associating values with Real variables. Input Format0r (zero and lowercase "r") Example, 0r0.11
Float constant	Use this when associating values with Float variables. Input Format0f (zero and lowercase "f") Example, 0f0.11
Constant HEX Input	Use this when entering hexadecimal values in an integer variable. Input Format0x (zero and lowercase "x") Example, 0xFE

NOTE

- When fractional values exceed 4 digits, the following notation is used.
Example)0f0.00001 -> 0f1e-05
0f0.000001 -> 0f1e-06

29.3.7 Logic Device when using the Address Format

If you set Logic to Address Format, the following devices become available.

Name	Type	Name	Description
Logic Address	Bit	X_ /Y_ /M_	Bit-type logic address
	Integer	D_ /I_ /Q_	Integer-type logic address. You can use modifiers (same as variables).
	Float	F_	Float-type logic address.
	Real	R_	Real-type logic address
	Timer	T_	Timer-type logic address. It is a structure, the same as a variable.
	Counter	C_	Counter-type logic address. It is a structure, the same as a variable.
	Date	N_	Date-type logic address. It is a structure, the same as a variable.
	Time	J_	Time-type logic address. It is a structure, the same as a variable.
	PID	U_	PID-type logic address. It is a structure, the same as a variable.

29.4 Number of Steps Per Instruction

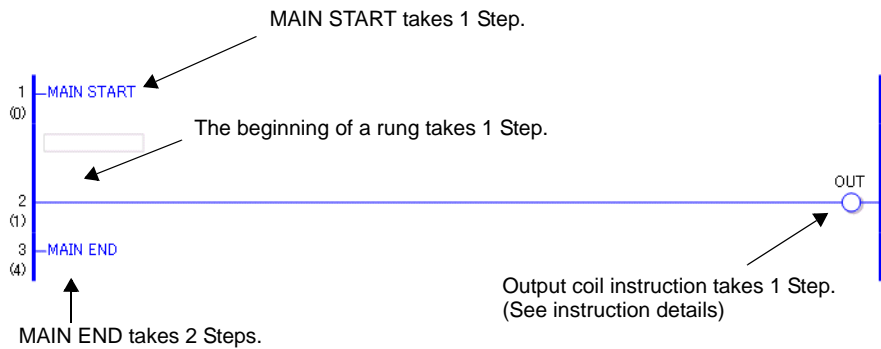
The conversion of the number of steps per instruction is described. (For details on the number of steps for each instruction, refer to the description of the relevant instruction.) The following program uses only the output coil OUT which is always ON.

Definition of variable OUT

Variable name → out

Retentive settings → volatile

Array element → none



The total is 5 steps.



For 1 step instructions, the number of steps indicated below a rung number and the actual number of steps may differ, as 1 step instructions are optimized upon saving and error checking.

29.5 Instruction Descriptions

29.5.1 Bit Instructions

■ NO (Normally Open) / NC (Normally Closed)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps in Instruction
NO (Normally Open)	S1 	Input	1 to 5
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps in Instruction
NC (Normally Closed)	S1 	Input	1 to 5

◆ Operand Settings

The following describes the specifiable content of Operand (S1).

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
External Device Address	Bit	—	2	O
	Word	Specify a bit in the word. (Example: [PLC1]D0000.00)	3	O
Internal Address	Bit	—	2	O
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	3	O
Symbol	Bit	—	2	O
	Word	—	—	X

Continued

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Variable Format	Bit	Arrays are not specified. Specify inputs, outputs, or up to 1536 Volatile items.	1	O
		Arrays are not specified. Retentive items or more than 1536 Volatile items.	2	O
		Specify bit array ([constant])	3	O
		Specify bit array ([variable])	4	O
	Integer	Arrays and modifiers are not specified	—	X
		Specify integer variable.X[constant]	3	O
		Specify integer variable.X[variable]	4	O
		Specify integer variable[constant/variable].X[constant/variable]	5	O
Variable Format	Float	—	—	X
	Real	—	—	X
	Timer	.Q / .TI / .R only	3	O
	Counter	.R / .UP / .QU / .QD / .Q only	3	O
	Date	—	—	X
	Time	—	—	X
	PID	.Q / .UO / .TO / .PF / .IF only	3	O

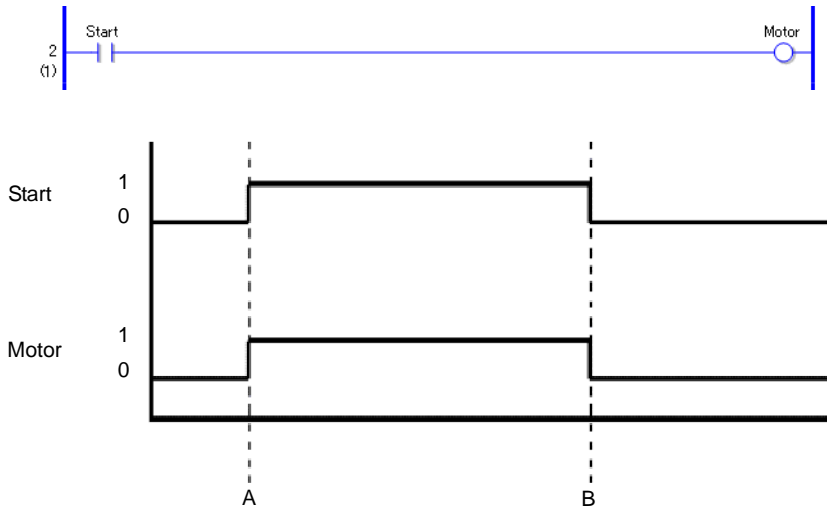
Continued

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Address Format	X_	—	1	O
	Y_	—	1	O
	M_	Within the clear type range (M_0000 to M_1535)	1	O
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.X[constant]	3	O
		D_****.X[address]	4	O
	F_	—	—	X
	R_	—	—	X
	T_	.Q / .TI / .R only	3	O
	C_	.R / .UP / .QU / .QD / .Q only	3	O
	N_	—	—	X
	J_	—	—	X
U_	.Q / .UO / .TO / .PF / .IF only	3	O	

◆ **Explanation of the NO Instruction**

- Use a NO instruction to determine the ON or OFF state. The NO instruction can be used to determine the ON or OFF state of an external input or an internal coil.
- You cannot use a NO instruction without including another instruction just to the left of the right power bar. The other instruction can be an output instruction or any instruction other than an input.

Program example



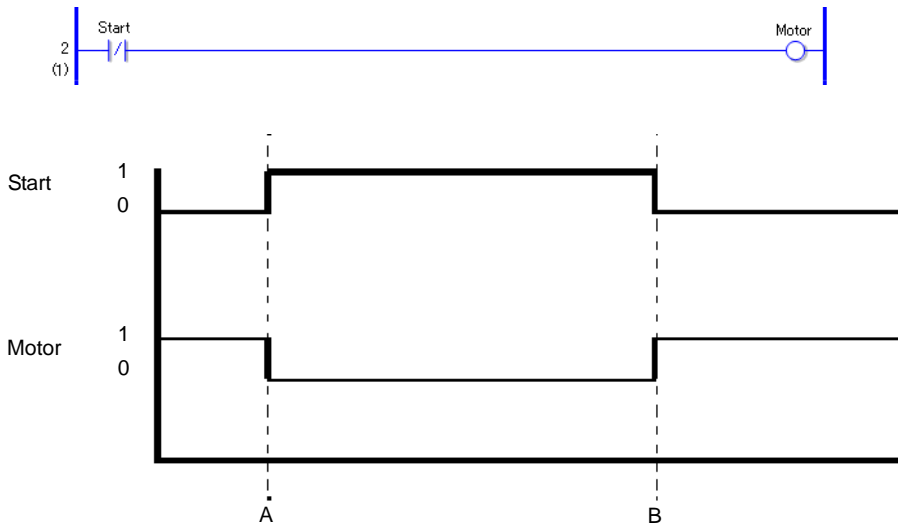
Point A When the bit variable Start turns ON, the NO instruction closes the contacts and the bit variable Motor turns ON.

Point B When the bit variable Start turns OFF, the NO instruction opens the contacts and the bit variable Motor turns OFF.

◆ **Explanation of the NC Instruction**

- Use a NC instruction to determine the ON or OFF state. The instruction can be used to determine the ON or OFF state of an external input or an internal coil.
- You cannot use a NC instruction without including another instruction just to the left of the right power bar. The other instruction can be an output instruction or any instruction other than an input.

Program example





Point A When the bit variable Start turns ON, the NC instruction opens the contacts and the bit variable Motor turns OFF.

Point B When the bit variable Start turns OFF, the NC instruction closes the contacts and the bit variable Motor turns ON.

Note: To retain the state when the power is turned OFF, set the symbol variable to Retentive. Use a keep address for the address format. (The keep setting cannot be used for external inputs and outputs.)

■ **OUT (Output Coil) / OUTN (Negative Output Coil)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps in Instruction
OUT (Output Coil)	D1 	Output	1 to 5
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps in Instruction
OUTN (Negative Output Coil)	D1 	Output	1 to 5

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1).

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
External Device Address	Bit	—	2	O
	Word	Specify a bit in the word. (Example: [PLC1]D0000.00)	3	O
Internal Address	Bit	—	2	O
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	3	O
Symbol	Bit	—	2	O
	Word	—	—	X

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Variable Format	Bit	Arrays are not specified. Up to 1536 Outputs set to Volatile	1	O
		Arrays are not specified. Retentive items or more than 1536 Volatile items.	2	O
		Specify bit array ([constant])	3	O
		Specify bit array ([variable])	4	O
	Integer	Arrays and modifiers are not specified	—	X
		Specify integer variable.X[constant]	3	O
		Specify integer variable.X[variable]	4	O
		Specify integer variable[constant/variable].X[constant/variable]	5	O
	Float	—	—	X
	Real	—	—	X
	Timer	.Q / .TI / .R only	3	O
	Counter	.R / .UP / .QU / .QD / .Q only	3	O
	Date	—	—	X
	Time	—	—	X
PID	.Q / .UO / .TO / .PF / .IF only	3	O	

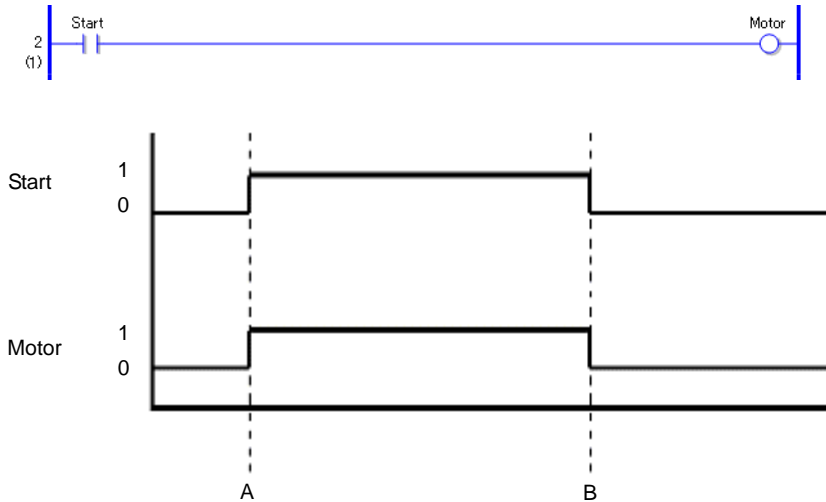
Continued

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	1	O
	M_	Within the clear type range (M_0000 to M_1535)	1	O
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.X[constant]	3	O
		D_****.X[address]	4	O
	F_	—	—	X
	R_	—	—	X
	T_	.Q / .TI / .R only	3	O
	C_	.R / .UP / .QU / .QD / .Q only	3	O
	N_	—	—	X
	J_	—	—	X
U_	.Q / .UO / .TO / .PF / .IF only	3	O	

◆ **Explanation of the OUT Instruction**

- Use an OUT instruction to output an ON or OFF result. Use the OUT instruction to turn ON or OFF an external input or an internal coil.
- Only one OUT instruction can be used in one rung. If a branch instruction is used, multiple OUT instructions can be used.
- Place OUT instructions immediately to the left of the right power bar.

Program example

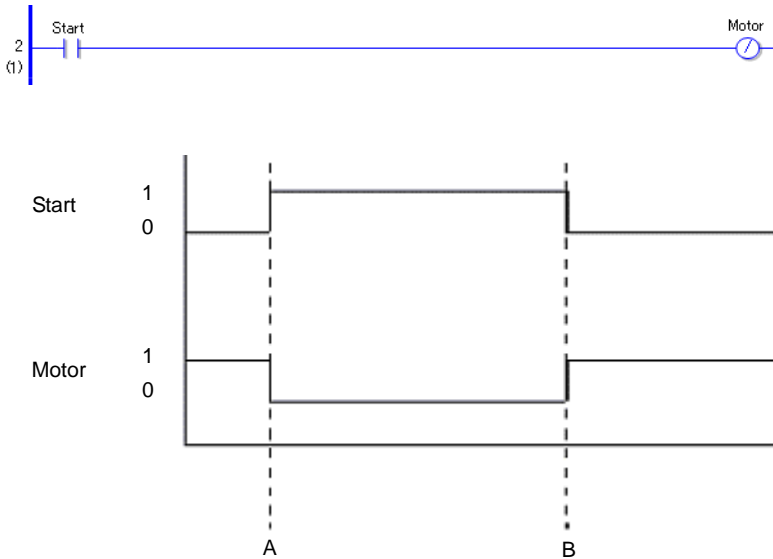


- Point A When the bit variable Start turns ON, the bit variable Motor of the OUT instruction turns ON.
- Point B When the bit variable Start turns OFF, the bit variable Motor of the OUT instruction turns OFF.

◆ **Explanation of the OUTN Instruction**

- Use an OUTN instruction to invert and output an ON or OFF result. Use this instruction to turn ON or OFF an external input or an internal coil.
- Only one OUTN instruction can be used in one rung. If a branch instruction is used, multiple OUTN instructions can be used.
- Place OUTN instructions immediately to the left of the right power bar.

Program example

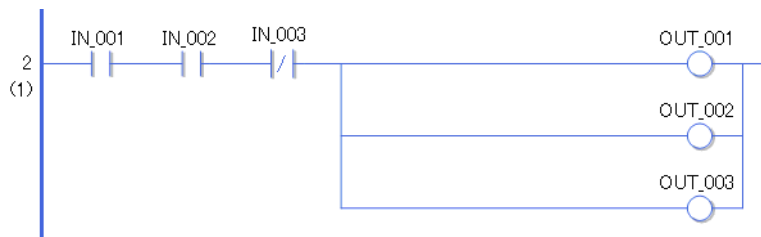


Point A When the bit variable Start turns ON, the bit variable Motor of the OUTN instruction turns OFF.

Point B When the bit variable Start turns OFF, the bit variable Motor of the OUTN instruction turns ON.

Note: To retain the state when the power is turned OFF, set the symbol variable to Retentive. Use a keep address for the address format. (The keep setting cannot be used for external inputs and outputs.)



When using multiple OUT and OUTN instructions



The example above shows how to use multiple OUT instructions by branching OUT instructions. An error will occur if OUT_001 and OUT_002 are placed in a series.

■ **SET (Set Coil) / RST (Reset Coil)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps in Instruction
SET (Set Coil)	D1 	Output	1 to 5
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps in Instruction
RST (Reset Coil)	D1 	Output	1 to 5

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1).

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
External Device Address	Bit	—	2	O
	Word	Specify a bit in the word. (Example: [PLC1]D0000.00)	3	O
Internal Address	Bit	—	2	O
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	3	O
Symbol	Bit	—	2	O
	Word	—	—	X

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Variable Format	Bit	Arrays are not specified. Up to 1536 Outputs set to Volatile	1	O
		Arrays are not specified. Retentive items or more than 1536 Volatile items.	2	O
		Specify bit array ([constant])	3	O
		Specify bit array ([variable])	4	O
	Integer	Arrays and modifiers are not specified	—	X
		Specify integer variable.X[constant]	3	O
		Specify integer variable.X[variable]	4	O
		Specify integer variable[constant/variable].X[constant/variable]	5	O
	Float	—	—	X
	Real	—	—	X
	Timer	.Q / .TI / .R only	3	O
	Counter	.R / .UP / .QU / .QD / .Q only	3	O
	Date	—	—	X
	Time	—	—	X
PID	.Q / .UO / .TO / .PF / .IF only	3	O	

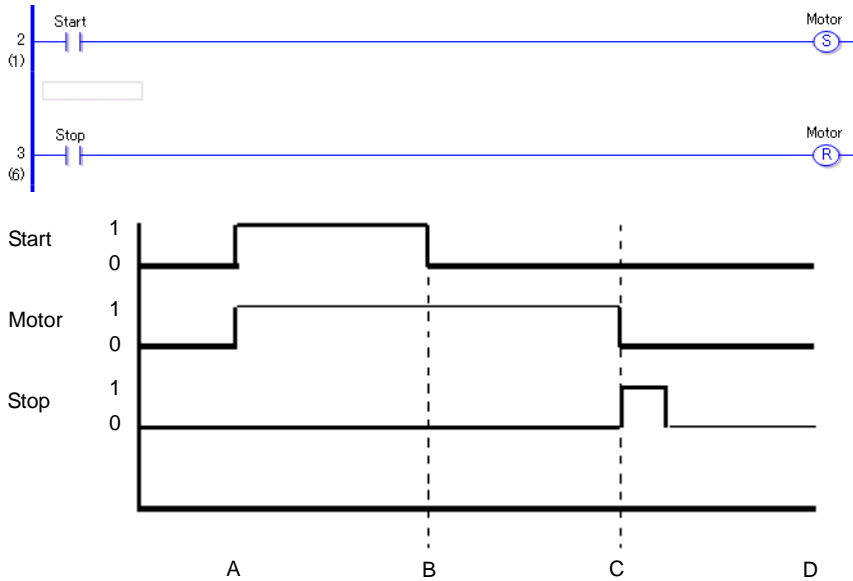
Continued

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	1	O
	M_	Within the clear type range (M_0000 to M_1535)	1	O
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.X[constant]	3	O
		D_****.X[address]	4	O
	F_	—	—	X
	R_	—	—	X
	T_	.Q / .TI / .R only	3	O
	C_	.R / .UP / .QU / .QD / .Q only	3	O
	N_	—	—	X
	J_	—	—	X
U_	.Q / .UO / .TO / .PF / .IF only	3	O	

◆ **Explanation of the SET and RST Instructions**

- The SET instruction keeps the ON state regardless of the input state.
- The RST instruction keeps the OFF state regardless of the input state.
- Use the SET and RST instructions to turn ON or OFF external outputs or internal coils.
- Only one OUT instruction can be used in one rung. If a branch instruction is used, multiple OUT instructions can be used.

Program example





- Point A The bit variable (Start) turns ON, the SET instruction executes, and then, bit variable (Motor) turns ON.
- Point B The bit variable (Start) turns OFF; however, bit variable (Motor) keeps the ON state.
- Point C The bit variable (Stop) turns ON, the RST instruction executes. Then, bit variable Motor turns ON.
 When the RST instruction turns the bit variable (Motor) ON, the state is cleared and the bit variable (Motor) changes from ON to OFF.
- Point D The bit variable (Motor) remains in the OFF state until the bit variable (Start) turns ON.

29.5.2 Pulse Instruction

■ PT (Positive Transition) / NT (Negative Transition)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps in Instruction
PT (Positive Transition)	S1 	Input	2 to 5
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps in Instruction
NT (Negative Transition)	S1 	Input	2 to 5

◆ Operand Settings

The following describes the specifiable content of Operand (S1).

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
External Device Address	Bit	—	2	O
	Word	Specify a bit in the word. (Example: [PLC1]D0000.00)	3	O
Internal Address	Bit	—	2	O
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	3	O
Symbol	Bit	—	2	O
	Word	—	—	X
Variable Format	Bit	Specify a bit	2	O
		Specify bit array ([constant])	3	O
		Specify bit array ([variable])	4	O

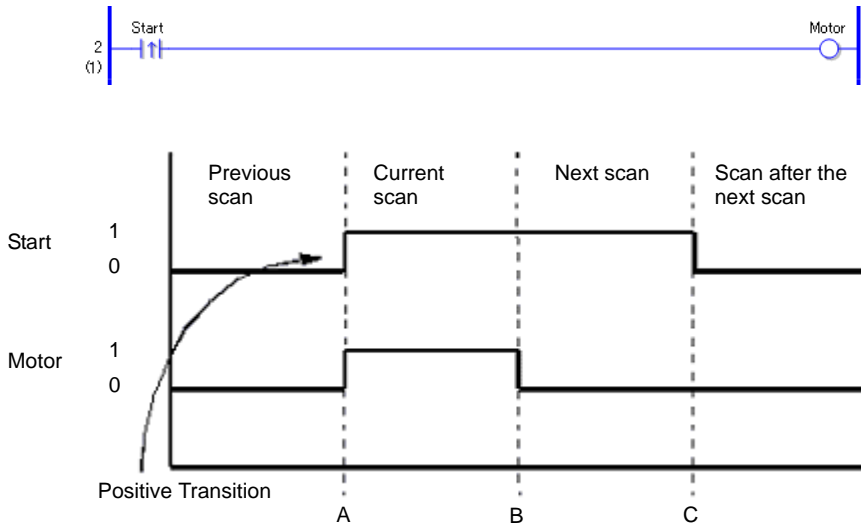
Continued

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Variable Format	Integer	Arrays and modifiers are not specified	—	X
		Specify integer variable.X[constant]	3	O
		Specify integer variable.X[variable]	4	O
		Specify integer variable[constant/variable].X[constant/variable]	5	O
	Float	—	—	X
	Real	—	—	X
	Timer	.Q / .TI / .R only	3	O
	Counter	.R / .UP / .QU / .QD / .Q only	3	O
	Date	—	—	X
	Time	—	—	X
PID	.Q / .UO / .TO / .PF / .IF only	3	O	
Address Format	X_	—	2	O
	Y_	—	2	O
	M_	—	2	O
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.X[constant]	3	O
		D_****.X[address]	4	O
	F_	—	—	X
	R_	—	—	X
	T_	.Q / .TI / .R only	3	O
	C_	.R / .UP / .QU / .QD / .Q only	3	O
	N_	—	—	X
	J_	—	—	X
U_	.Q / .UO / .TO / .PF / .IF only	3	O	

◆ **Explanation of the Positive Transition (PT) Instruction**

- When a PT instruction bit variable turns ON, only the first scan turns ON. Subsequent scans are OFF even though the bit variable may be in the ON state. You can use the PT instruction for counting the number of ON states.
- You cannot use a NO instruction without including another instruction just to the left of the right power bar. The other instruction can be an output instruction or any instruction other than an input.

Program example

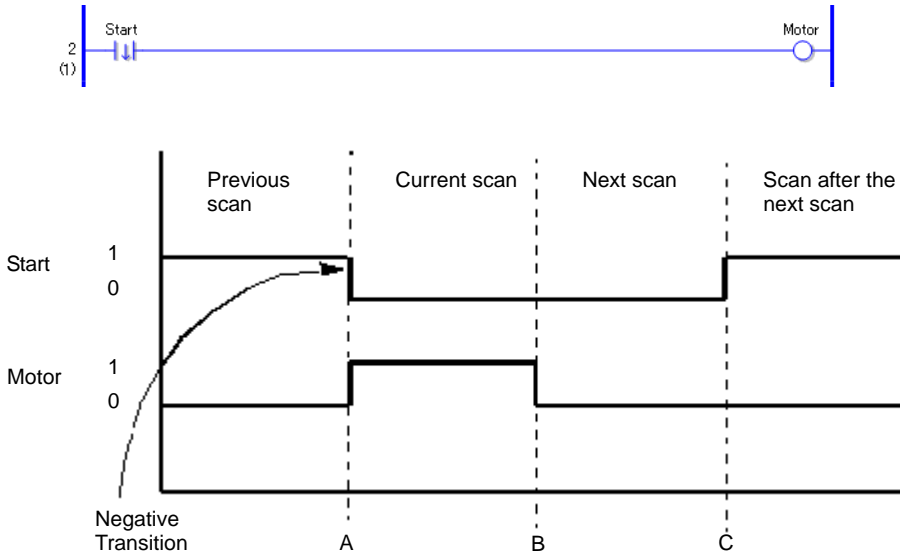


- Point A The variable (Start) turns ON, and then the variable motor turns ON.
- Point B After a scan is executed one time, the variable (Motor) is turned OFF.
- Point C The variable (Motor) remains OFF because the upward transition of the variable (Start) is not detected.

◆ **Explanation of the Negative Transition (NT) Instruction**

- When an NT instruction is executed, if the variable that was ON during the previous scan is turned OFF during the current scan, the NT instruction will execute only during the current scan. The NT instruction cannot execute on an initial scan, because the state of the previous scan is always considered to be OFF. Therefore, on an initial scan, the NT instruction will not be conducted even after the instruction is executed. The following example describes the features of the NT instruction.

Program example



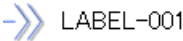
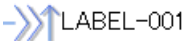
- Point A The variable (Start) turns OFF, and then variable motor turns ON.
- Point B After a scan is executed once, the variable motor will be turned OFF.
- Point C The variable (Motor) remains OFF because the upward transition of the variable (Start) is not detected.

(Supplementary) For the positive transition and negative transition instruction operands, you must pay attention when performing indirect addressing to each element, especially when an element is specifying an array or bit using variables. The variable in the operand of the previous execution is compared with the variable in the operand of the current execution, and then an instruction is executed. Therefore, if the variable value to be specified is different, the target for comparison will differ.

29.5.3 Program Control

■ JMP (Jump) / JMPP (Positive Transition Jump)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JMP (Jump)	 LABEL-001	Control	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JMPP (Negative Transition Jump)	 LABEL-001	Control	2

Up to 192 labels can be specified for a JMP instruction. When specifying a label for the JMP destination, previously specified label names will display. (If a label has not been specified, the label name will not display.) Insert the label first and then specify the label for the jump instruction.

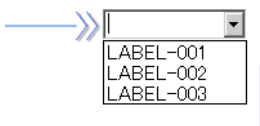
◆ Specifying Labels



Right-click and select [Insert Label], or on the [Logic] menu click [Insert Label].

You can select a label from 192 labels ranging from LABEL-001 to LABEL-192.

Label names cannot be arbitrarily specified.



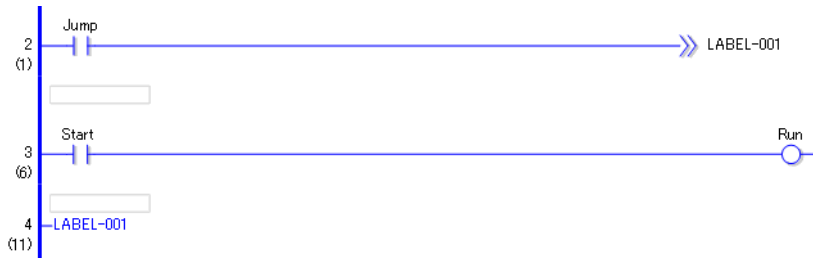
Only labels used in the program are displayed. The same label names cannot be used on the INIT, MAIN, and SUB screens.

When you execute a JMP instruction, the program will jump to the specified label. Unlike a JSR instruction, the program does not automatically return to the rung of the jump source. It is not possible to jump over the INIT or SUB block. Create a program that jumps to a label within a block. Also, note that if the program jumps up the program, it may result in an infinite loop.

A JMPP instruction executes a jump instruction only when an upward transition is detected. The processing after a jump is the same as the JMP instruction.

Program Example

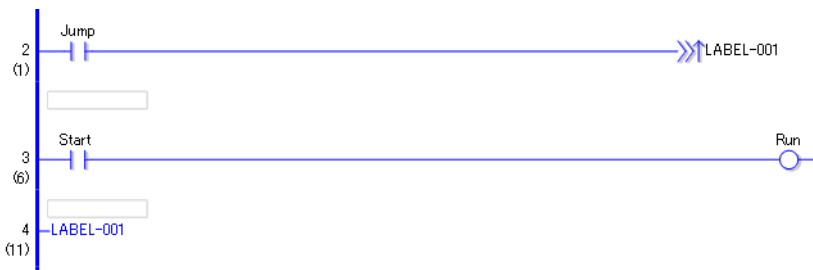
JMP



When the NO variable (Jump) is turned ON, the JMP instruction will be executed and the program will jump to the fourth rung set with the label name: "LABEL-001". After the jump, the program continues executing after the fourth rung. As long as the Normally Open (NO) instruction remains ON, the program in the third rung will not execute.

Program Example

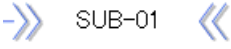
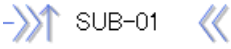
JMPP



Only the upward transition of the normally open instruction is detected, and the JMPP instruction executes. Then, the program jumps to the fourth rung with the label name: "LABEL-001". After the jump, the program continues executing after the fourth rung. During subsequent scans, the JMPP instruction does not execute, even if the normally open instruction remains ON. After one scan, the program in the third rung executes.

■ JSR (Subroutine Call) / JSRP (Positive Transition Subroutine Call)

Symbols and Features

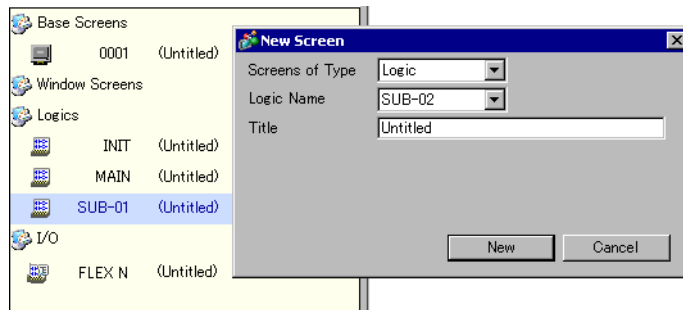
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JSR (Subroutine Call)		Control	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JSRP (Positive Transition Subroutine Call)		Control	2

Using JSR instructions you can specify up to 32 subroutines.

To define a jump to a subroutine, first create the subroutine. Without a subroutine, you cannot define a subroutine jump. You can define jumps only to subroutines that are already created.

◆ Specifying Subroutines

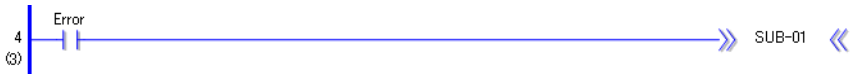
To create a subroutine screen, on the [Screen List Window] select [New Screen], or on the [Screen] menu click [New Screen].



The destinations you can specify for a subroutine instruction are SUB-01 to SUB-32. The subroutine name is fixed and cannot be arbitrarily named.

Program Example

JSR



When the normally open instruction is turned ON to indicate a problem, the JSR instruction will be executed. The JSR instruction jumps to the subroutine screen "SUB-01" and executes the program. When "SUB-01" ends, the program will return to the rung after the JSR instruction and continue executing. In subsequent scans, if the normally open instruction is still ON, the JSR instruction will execute. Place JSR instructions at the end of rungs. Place a JSR instruction in the last row.

Program Example

JSRP



When the upward transition of a normally open instruction is detected the JSRP instruction executes. The JSRP instruction jumps to the subroutine screen "SUB-01" and executes the program. When "SUB-01" ends, the program returns to the rung after the JSRP instruction and continues executing. In subsequent scans, if the normally open instruction remains ON, the JSRP instruction will not execute. After the first scan, the subroutine does not run, and the program continues executing rungs that follow. Place JSRP instructions at the end of rungs.

After one scan, the subroutine processing is not performed, and the processing in the next rung is performed.


Place a JSRP instruction in the last row.

◆ Restrictions

- (1) JSR and JSRP instructions are placed only at the right end of a row.
- (2) A subroutine jump is possible up to 128 times.
 - One stack is used for one subroutine jump. A total of 128 stacks can be used for a logic program.
 - Other instructions that use stacks are FOR and NEXT instructions. Each instance of FOR/ NEXT instructions use two stacks.

■ RET(Return)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
RET (Return)		Control	1

RET instructions return the program from a subroutine to the original JSR instruction call, and continues executing instructions in rungs that follow.

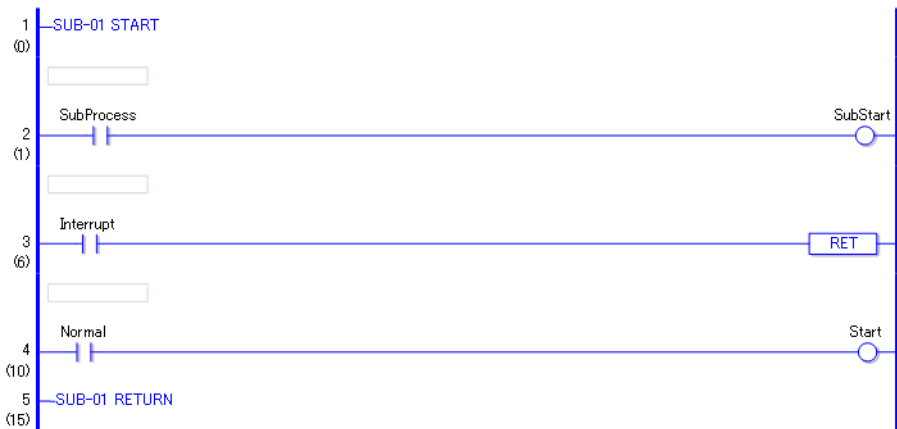
Use RET instructions to interrupt the subroutine and return to the MAIN program.

Because the program automatically returns to the caller after the subroutine processing ends, it is not always necessary to use an RET instruction.

Place RET instructions at the end of rungs. RET instructions can only be used in subroutines.

Program Example

RET





RET instructions can only be used in subroutines. When the subroutine call instruction is executed in MAIN, the program flow moves to the subroutine. The subroutine processes instructions in rungs 1 and 2. If the variable for the normally open instruction in rung 3 is ON, the RET instruction is executed and returns the program flow to MAIN without executing the fourth rung.

When the RET instruction is not executed, the program is executes the fourth rung, then returns the program to MAIN after the subroutine ends (END).

■ FOR NEXT (Repeat)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
FOR (Repeat)		Control	2 to 4
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NEXT (Repeat)		Control	1

◆ Operand Settings

The following table lists the configurable conditions of Operand (S1) in the FOR instruction.

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	2	O
Internal Address	Bit	—	—	X
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	2	O
Symbol	Bit	—	—	X
	Word	—	2	O

Continued

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer	Arrays and modifiers are not specified	2	O
		Specify integer variable[constant]	3	O
		Specify integer variable [Variable]	4	O
		Specify integer variable[constant/variable] .X[constant/variable]	—	X
	Float	—	—	X
	Real	—	—	X
	Timer	.PT / .ET only	2	O
	Counter	.PV / .CV only	2	O
	Date	.ΨP / .MO / .ΔΑΨ ονλψ	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O

Continued

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	2	O
	Q_	—	2	O
	D_	Modifiers are not specified	2	O
		D_****.X[constant]	—	X
		D_****.X[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	—	0 to 2147483647	2	O

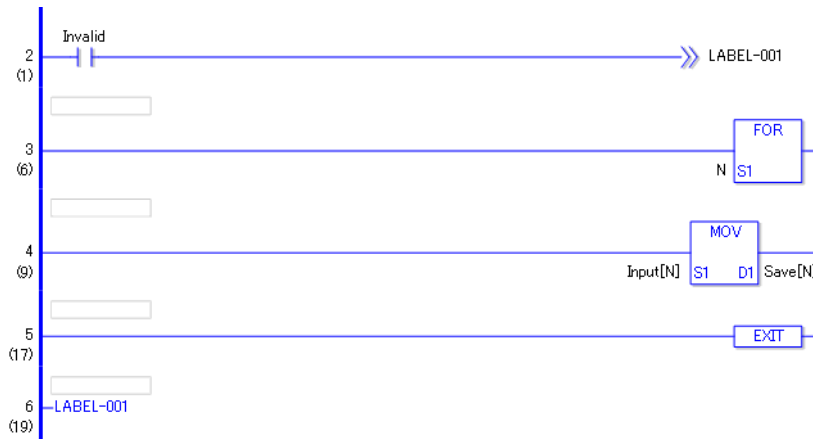
◆ Explanation of FOR and NEXT Instructions

FOR and NEXT instructions repeat the logic between FOR and NEXT the number of times specified in S1. After the processing between the FOR and NEXT instructions has been executed the number of times specified in S1, the rung that follows the NEXT instruction will run without any conditions. When S1 is 0 or less, the logic between FOR and NEXT will not execute and the program will jump to the rung that follows the NEXT instruction. Always use FOR and NEXT instructions as a pair. These instructions always run.

Program Example

FOR and NEXT

Other instructions cannot coexist on the same rung as FOR and NEXT instructions. You can use a JMP instruction to specify conditions for executing FOR and NEXT instructions. The following program example of FOR and NEXT instructions shows how you can use a condition to run FOR and NEXT instructions.




When the variable of the normally open instruction turns ON FOR and NEXT will not execute, and the program will jump to "LABEL-001". When the variable is OFF, the FOR and NEXT instructions execute. The value (N) of the FOR instruction's operand S1 indicates the number of times that the rungs between the FOR and NEXT instructions will be repeated. When S1 = 10, the FOR loop is repeated 10 times. After exiting the FOR loop, processing continues with instructions that follow the NEXT instruction.

◆ Restrictions

- (1) After inserting a FOR instruction, you need to also insert the corresponding NEXT instruction.
- (2) Do not insert instructions on the same rung before or after FOR to NEXT instructions. (You cannot set any conditions on rungs with FOR or NEXT instructions.)
- (3) You cannot change the number of executions between FOR and NEXT instructions.
- (4) You cannot exit FOR and NEXT instructions midway.
- (5) You can nest FOR and NEXT instructions up to 64 times. After exceeding 64 nests, a major error occurs and error code 4 is written to # L_FaultCode.
- (6) For each nest, two stacks are used. A total of 128 stacks can be used in the logic program. Other than the FOR and NEXT instructions, the JSR instruction also uses stacks. The JSR instruction uses only one stack.

■ INV(Invert)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
INV (Invert)		Control	1

◆ Explanation of the Invert (INV) Instruction

When an INV instruction is executed, invert processing is performed. If the state is OFF before the INV instruction is executed, the state will be inverted to ON.

If the state is ON before the INV instruction is executed, the state will change to OFF as a result of the INV instruction.

Program example




When the operand of the normally open instruction is ON, the INV instruction will execute and the OUT coil turns OFF.

When the operand of the normally open instruction is ON, the INV instruction will execute and the OUT coil turns OFF.

■ EXIT(End of Processing)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
EXIT (End of Processing)		Control	1

◆ Explanation of the EXIT Instruction

An EXIT instruction can be used only in the MAIN program. After this instruction is executed, the program jumps to END.

After the instruction has been executed, processing of instructions between EXIT and END is not performed. This instruction jumps to the END label in the same way as a jump instruction.

Program example





When the switch is turned ON, the EXIT instruction at the end of the rung will run.

Therefore, processing of instructions between EXIT and END is not performed.

■ **PBC (Power Bar Control) and PBR (Power Bar Reset)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
PBC (Power Bar Control)		Control	3
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
PBR (Power Bar Reset)		Control	2

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1) and (D1) in the PBC instruction.

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X

Continued

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Variable Format	Bit	Bit specifications (D1 operand only)	3	O
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	—	X
		Specify integer variable [Variable]	—	X
		Specify integer variable[constant/variable] .X[constant/variable]	—	X
	Float	—	—	X
	Real	—	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X

Continued

Name	Type	Condition	Number of Steps in Instruction	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	(D1 operand only)	3	O
	M_	(D1 operand only)	3	O
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.X[constant]	—	X
		D_****.X[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	—	0 to 7 (S1 operand only)	3	O

◆ **Operand Settings**

The following describes the specifiable content of operands (S1) in the PBR instructions.

Name	Type	Condition	Number of Steps	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify a bit in the word. (Example: [PLC1]D0000.00)	—	X
Internal Address	Bit	—	—	X
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer	Arrays and modifiers are not specified	—	X
		Specify integer variable.X[constant]	—	X
		Specify integer variable.X[variable]	—	X
		Specify integer variable[constant/variable].X[constant/variable]	—	X
	Float	—	—	X
	Real	—	—	X
	Timer	.Q / .TI / .R only	—	X
	Counter	.R / .UP / .QU / .QD / .Q only	—	X
	Date	—	—	X
	Time	—	—	X
	PID	.Q / .UO / .TO / .PF / .IF only	—	X

Continued

Name	Type	Condition	Number of Steps	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.X[constant]		—	X
		D_****.X[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.Q / .TI / .R only		—	X
	C_	.R / .UP / .QU / .QD / .Q only		—	X
	N_	—	—	X	
	J_	—	—	X	
U_	.Q / .UO / .TO / .PF / .IF only		—	X	
Constant	—	0 to 7 (S1 operand only)	2	O	

◆ **Explanation of the Power Bar Control (PBC) and Power Bar Reset (PBR) Instructions**

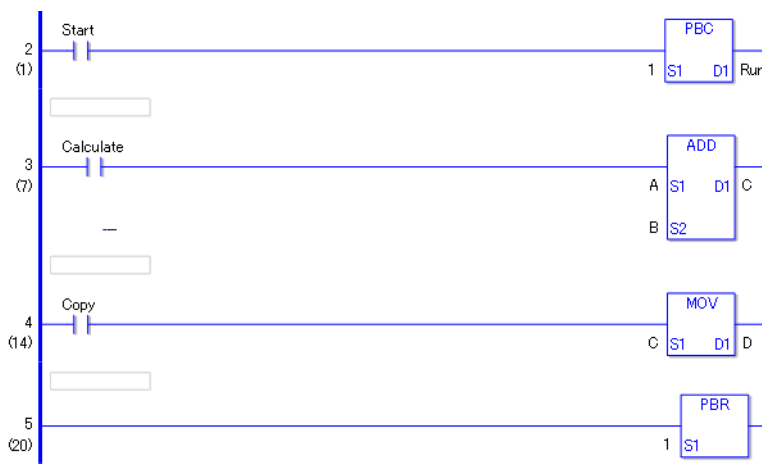
When a PBC instruction is executed, the program between PBC and PBR will be executed. PBC and PBR instructions can be used only in MAIN. They cannot be used in other parts of the program.

When the PBC instruction is turned ON, the bit variable in D1 will turn ON. The program running between PBC and PBR instructions executes ON processing until the PBC instruction turns OFF.

For every PBC instruction, one PBR instruction is always required.

PBC/PBR instruction S1 specifies nesting level. The processing of the specified level between PBC and PBR is executed.

Program example (without nesting)



When the variable of the normally open instruction is ON, the PBC instruction will execute. When the PBC instruction is executed, processing between PBC and PBR instructions is executed.

(1) When the PBC instruction is OFF (PBC execution bit is OFF)

The ADD instruction does not execute even when the normally open instruction in the third rung is ON.

The MOV instruction does not execute even when the normally open instruction in the fourth rung is ON.

(2) When the PBC instruction is ON (PBC execution bit is ON)

The ADD instruction is executed when the normally open instruction in the third rung turns ON.

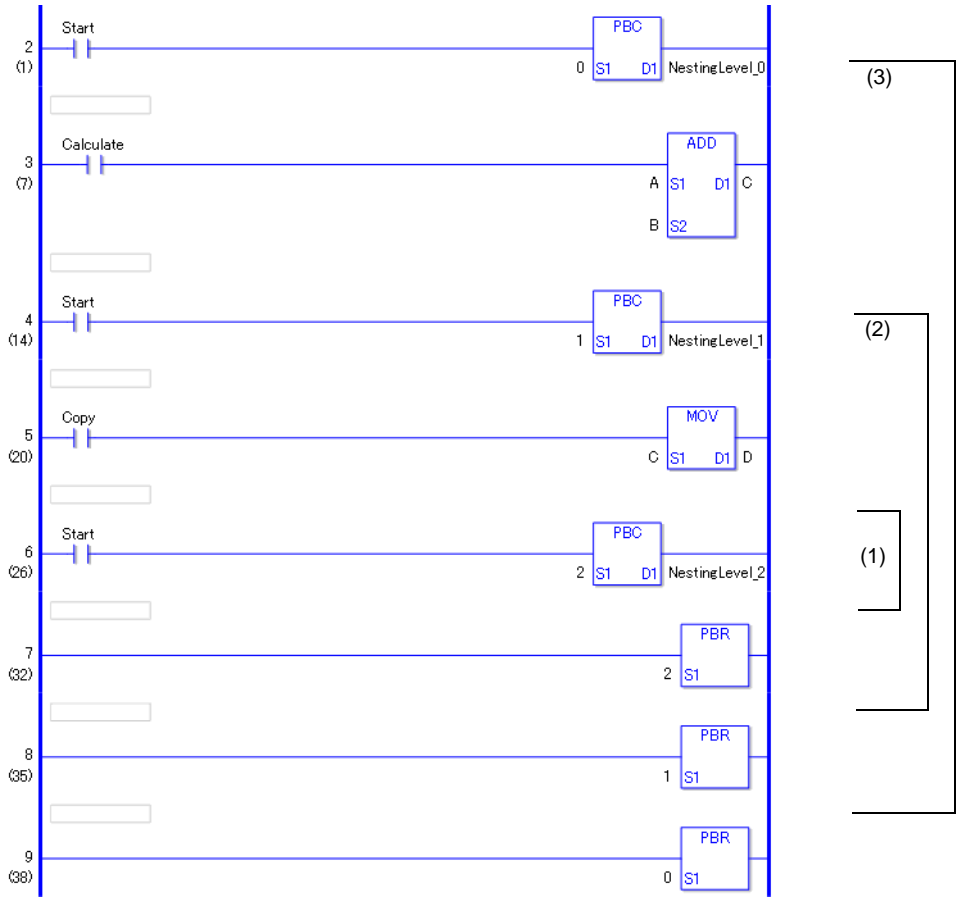
The MOV instruction is executed when the normally open instruction in the fourth rung turns ON.

◆ **State of Each Instruction**

Elements that keep their state: Elements driven by an accumulative timer, counter, or SET and RST instructions.

Elements that turn OFF: Elements driven by a timer and an OUT instruction.

Program example (with nesting, three levels)



◆ **PBC instruction Nesting**

A PBC instruction can be programmed with up to eight levels of nesting.

When a PBC instruction is used within a PBC instruction, nesting level numbers (S1) must be incremented.

(0→1→2→3→4→5→6→7)

To release nesting levels, use a PBR instructions.

(7→6→5→4→3→2→1→0)

For example, if you release the nested PBR 5 without releasing PBR 6 and PBR 7, nesting levels down to the fifth level will be released.


(1) This is nesting level 2. In the previous program, the state is low.

(2) This is nesting level 1. In the previous program, the state is medium.

(3) This is nesting level 0. In the previous program, the state is high.

■ **LWA(Logic Wait)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
LWA (Logic Wait)		Control	2

◆ **Operand Settings**

The following describes the specifiable content of operand (S1).

Name	Type	Condition	Number of Steps	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	—	X
		Specify integer variable [Variable]	—	X
		Specify integer variable[constant/variable] .X[constant/variable]	—	X

Continued

Name	Type	Condition	Number of Steps	Possible: O Not Possible: X
Variable Format	Float	—	—	X
	Real	—	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.X[constant]	—	X
		D_****.X[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	—	1 to 10	2	O

◆ **Explanation of the Logic Wait (LWA) Instruction**

An LWA instruction stops the logic for the time specified in S1. If a flicker occurs while a movie is being played, use the LWA instruction.

You can use LWA instructions to prevent flickering while a movie is played. The flow of execution Power always passes through the LWA instruction.

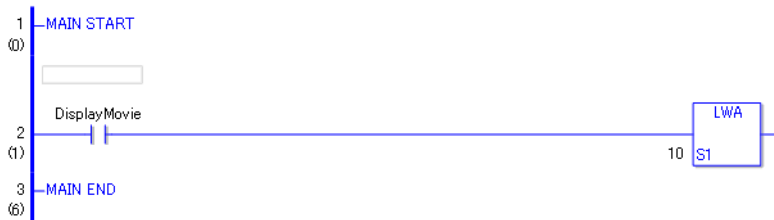
(Notes)

If a large number of LWA instructions are used, a WDT (watch dog time) error may occur. Attention must be paid when using LWA instructions since WDT errors affect the scan time.

Restrictions on use

- (1) If a large number of LWA instructions are used, a WDT (watch dog time) error may occur. Attention must be paid when using LWA instructions since WDT errors affect the scan time.
- (2) Only one LWA instruction can be placed in one rung.
- (3) An LWA instruction must be the last instruction on the rung, just to the left of the right power bar.
- (4) An LWA instruction can be used only in MAIN and SUB. It cannot be used in INIT.

Program example

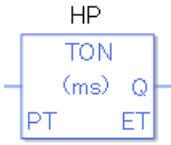
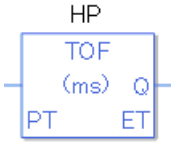


- (1) When the bit variable is turned ON, the LWA instruction will be executed.
- (2) When the LWA instruction is executed, the logic program stops for the time (1 to 10 ms) specified in operand S1.
- (3) After the specified time has elapsed, processing will continue on the next rung.

29.5.4 Timer Instruction

■ TON (ON Delay Timer) and TOF (OFF Delay Timer)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
TON (On Delay Timer)		Timer	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
TOF (Off Delay Timer)		Timer	2

◆ Explanation of the ON Delay Timer (TON) and OFF Delay Timer (TOF) Instructions

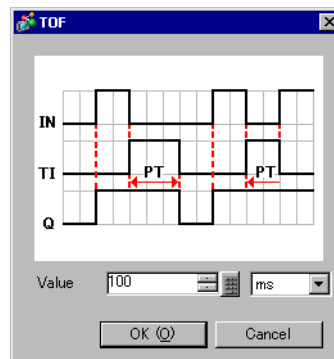
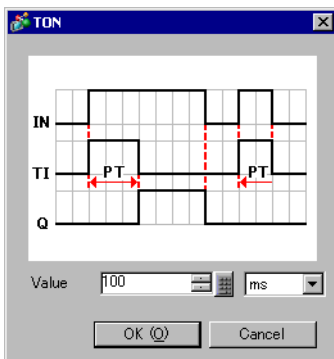
Timer variables used in TON and TOF instructions are structure variables. The following table lists the internal structures.

Timer Variable

Timer Variable	Variables Settings	Description
VariableName.TI	Bit Variable	Turns ON when the timer begins.
VariableName.Q	Bit Variable	Turns ON upon completion of the timer.
VariableName.PT	Integer Variable	The value set on the timer
VariableName.ET	Integer Variable	The current value on the timer

Double-click the timer instruction to display the following dialog box. Enter the preset time in this dialog box.

Enter the timer value and units.

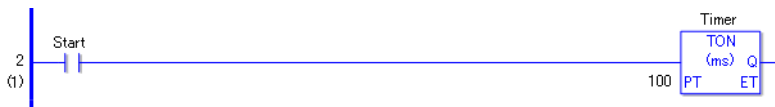


For time-based settings, double-click the timer instruction to display the setup dialog box.

Time base	Description	PT value/ET value
ms	Specify the time in units of ms.	The PT value is specified and displayed in units of ms. The ET value is displayed in units of ms. Setting range = 0 to 2147483647 x 1ms
10ms	Specify the time in units of 10 ms.	The PT value is set and displayed in units of 10 ms. The ET value is displayed in units of 10 ms. Setting range = 0 to 214748364 x 10 ms
0.1s	Specify the time in units of 0.1 s.	The PT value is specified and displayed in units of 0.1 s. The ET value is displayed in units of 0.1 s. Setting range = 0 to 21474836 x 100 ms
s	Specify the time in units of 1 s.	The PT value is specified and displayed in units of 1 s. The ET value is displayed in units of 1 s. Setting range = 0 to 2147483 x seconds

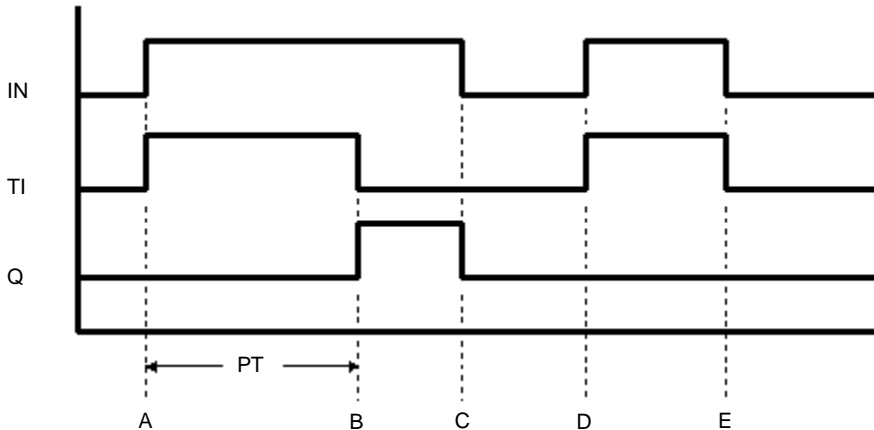
Program Example

TON



- (1) When the variable of the normally open instruction is turned ON, the elapsed time .ET will increase by the specified time-based units because the TON instruction is triggered.
 - The timer measurement bit .TI turns ON.
 - The timer output bit .Q turns OFF.
- (2) When the elapsed time .ET increases to equal the preset time .PT, the elapsed time .ET keeps the current value.
 - The timer measurement bit .TI turns OFF.
 - The timer output bit .Q turns ON and allows power to pass.
- (3) When start measurement is off (turned off), the elapsed time .ET will be reset to 0.
 - The timer measurement bit .TI turns OFF.
 - The timer output bit .Q turns OFF.

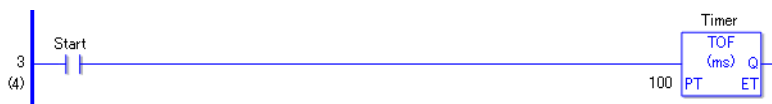
◆ Timing Chart for TON Instruction Operation



- Point A The timer turns ON and the timer measurement bit .TI turns ON. The timer measurement starts and the elapsed time .ET increases. The timer output bit .Q remains OFF.
- Point B When the elapsed time .ET equals the preset time .PT, the timer output bit .Q turns ON. The value of the elapsed time .ET remains the same as the preset time .PT. The timer measurement bit .TI turns OFF.
- Point C The timer turns OFF and the timer output bit .Q turns OFF. The elapsed time .ET resets to 0.
- Point D The timer turns ON and the timer measurement bit .TI turns ON. The timer measurement starts and the elapsed time .ET increases.
- Point E The timer turns OFF before the elapsed time .ET reaches the preset time .PT. While the timer output bit .Q remains OFF, the elapsed time .ET resets to 0.

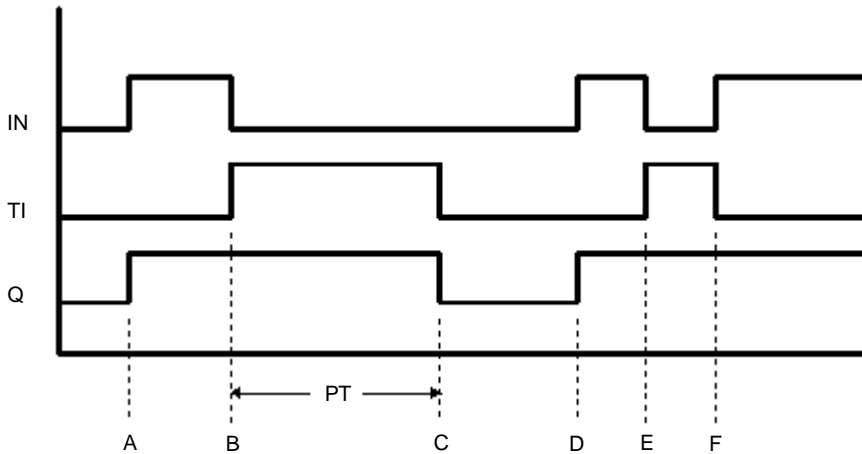
Program Example

TOF



- (1) When the variable for NO instructions is turned ON, the elapsed time .ET will be reset to 0 because the TON instruction is triggered.
 - The timer measurement bit .TI turns OFF.
 - The timer output bit .Q turns ON and allows power to pass.
- (2) When the TOF instruction is triggered and the measurement start bit is turned OFF, the elapsed time .ET will increase by the specified time-based units.
 - The timer measurement bit .TI turns ON.
 - The timer output bit .Q remains ON.
- (3) When the elapsed time .ET increases to equal the preset time .PT, the elapsed time .ET keeps the current value.
 - The timer measurement bit .TI turns OFF.

◆ Timing Chart for TOF Instruction Operation



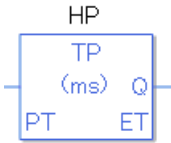
- Point A The timer turns ON. The timer measurement bit .TI remains OFF. The timer output bit .Q turns ON. The elapsed time .ET resets to 0.
- Point B The timer turns OFF. The timer starts measurement (.TI turns ON.) The timer output bit remains ON.
- Point C The elapsed time .ET equals the preset time .PT. The timer output bit .Q turns OFF. The timer stops measurement (.TI turns OFF). The elapsed time .ET remains equal to the setup time (ET = PT).
- Point D The timer turns ON. The timer measurement bit .TI remains OFF. The timer output bit .Q remains ON. The elapsed time .ET resets to 0.
- Point E The timer turns OFF. The timer starts measurement (.TI turns ON.) The timer output bit .Q remains ON.
- Point F The timer turns ON before the elapsed time .ET reaches the preset time .PT, and the timer stops measurement (.TI turns OFF). The timer output bit .Q remains ON and the elapsed time .ET resets to 0.

◆ Confirming Execution Results

- (1) If you input a value outside the defined range, an error occurs, the error code (6706) is written to #L_CalcErrCode, and the instruction does not execute. When troubleshooting, always check the error code in #L_CalcErrCode.

■ TP (Pulse Timer)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
TP (Positive Transition Timer)		Timer	2

◆ Explanation of the Pulse Timer (TP) Instruction

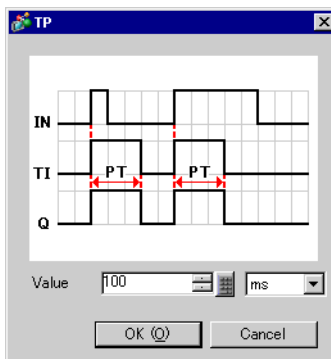
Timer variables used in TP instructions are structure variables. The following table lists the internal structures.

Timer Variable

Timer Variable	Variables Settings	Description
VariableName.TI	Bit Variable	Turns ON when the timer begins.
VariableName.Q	Bit Variable	Turns ON upon completion of the timer.
VariableName.PT	Integer Variable	The value set on the timer
VariableName.ET	Integer Variable	The current value on the timer

Double-click the timer instruction to display the following dialog box. Enter the preset time in this dialog box.

Enter the timer value and units.

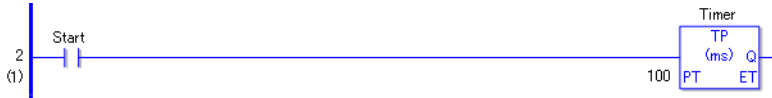


For time-based settings, double-click the timer instruction to display the setup dialog box.

Time base	Description	PT value/ET value
ms	Specify the time in units of ms.	The PT value is specified and displayed in units of ms. The ET value is displayed in units of ms. 0 to 2147483647 x 1 ms
10ms	Specify the time in units of 10 ms.	The PT value is set and displayed in units of 10 ms. The ET value is displayed in units of 10 ms. Setting range = 0 to 214748364 x 10 ms

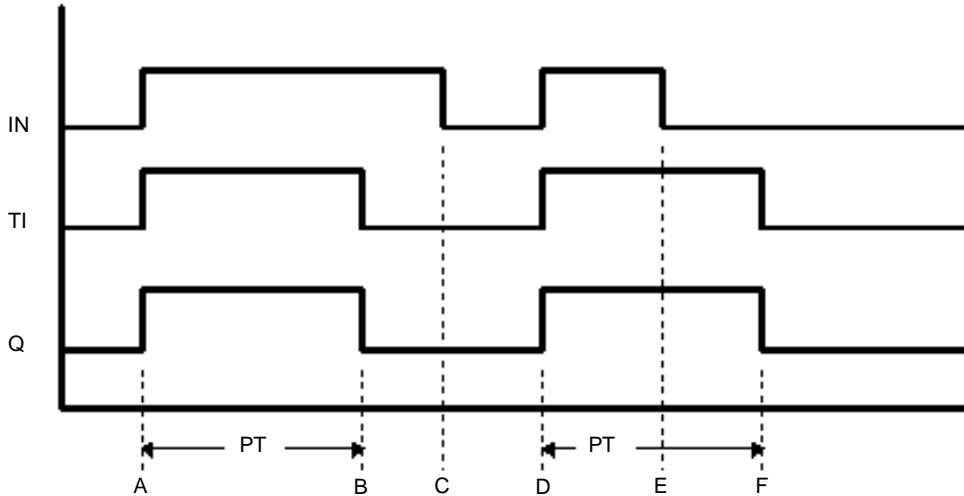
Time base	Description	PT value/ET value
0.1s	Specify the time in units of 0.1 s.	The PT value is specified and displayed in units of 0.1 s. The ET value is displayed in units of 0.1 s. Setting range = 0 to 21474836 x 100 ms
s	Specify the time in units of 1 s.	The PT value is specified and displayed in units of 1 s. The ET value is displayed in units of 1 s. Setting range = 0 to 2147483 x seconds

Program example



- (1) When the normally open instruction turns ON, the TP instruction is triggered. Because the TP instruction detects positive transitions, when the instruction is triggered, it starts the timer no matter what condition the timer was in.
The elapsed time .ET increases by the units specified as the time base.
 - The timer measurement bit .TI turns ON.
 - The timer output bit .Q turns ON and allows power to pass.
- (2) When the elapsed time .ET reaches the preset time, the TP instruction turns OFF.
The timer output bit .Q turns off after the preset time has elapsed, regardless of the power flow to the left of the TP instruction.
 - When $PT \leq ET$, it is immediately reset to 0.
 - When the elapsed time .ET equals the preset time .PT, the timer bit .TI is turned off.
 - When the TP instruction is off, the timer output bit .Q is off.
- (3) When the variable of the normally open instruction turns OFF, if the elapsed time .ET has reached the preset time .PT, the elapsed time .ET resets to 0.
 - The timer output bit .Q turns OFF.
 - Otherwise, the timer continues measurement and the timer output bit .Q remains ON.

◆ **Timing Chart for the TP Instruction**



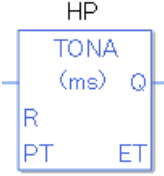
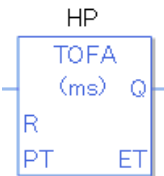
- Point A The timer turns ON. The timer starts measurement (.TI turns ON). The timer output bit .Q turns ON.
- Point B The elapsed time .ET equals the preset time .PT. The timer output bit .Q turns OFF. The timer stops measurement (.TI turns OFF). The elapsed time .ET remains equal to the preset time (ET = PT).
- Point C The timer turns OFF. The elapsed time .ET resets to 0.
- Point D The timer turns ON. The timer starts measurement (.TI turns ON). The timer output bit .Q turns ON.
- Point E The timer turns OFF. The timer continues measurement (.TI remains ON). The timer output bit .Q remains ON.
- Point F The elapsed time .ET equals the preset time .PT. The timer output bit .Q turns OFF. The timer stops measurement (.TI turns OFF). Because the timer input bit IN is OFF, the elapsed time .ET resets to 0.

◆ **Confirming Execution Results**

- (1) If you input a value outside the defined range, an error occurs, the error code (6706) is written to #L_CalcErrCode, and the instruction does not execute. When troubleshooting, always check the error code in #L_CalcErrCode.

■ **TONA (Accumulated ON Delay Timer) and TOFA (Accumulated OFF Delay Timer)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
TONA (Accumulated ON Delay Timer)		Timer	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
TOFA (Accumulated OFF Delay Timer)		Timer	2

◆ **Explanation of the Accumulated ON Delay Timer (TONA) and Accumulated OFF Delay Timer (TOFA) Instructions**

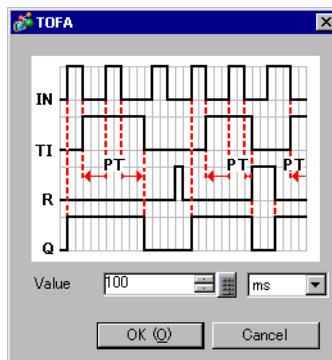
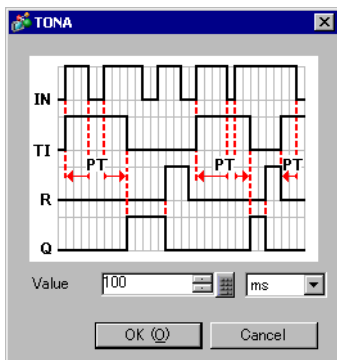
Timer variables in TONA and TOFA instructions are structure variables. The following table lists the internal structures.

Timer Variable

Timer Variable	Variables Settings	Description
VariableName.TI	Bit Variable	Turns ON when the timer begins.
VariableName.Q	Bit Variable	Turns ON upon completion of the timer.
VariableName.R	Bit Variable	Resets the current timer. Clear (0).
VariableName.PT	Integer Variable	The value set on the timer
VariableName.ET	Integer Variable	The current value on the timer

Double-click the timer instruction to display the following dialog box. Enter the preset time in this dialog box.

Enter the timer value and units.

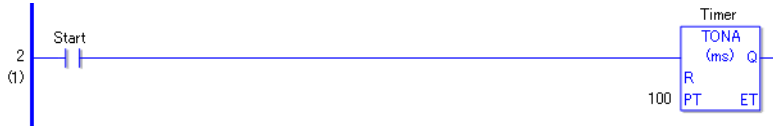


For time-based settings, double-click the timer instruction to display the setup dialog box.

Time base	Description	PT value/ET value
ms	Specify the time in units of ms.	The PT value is specified and displayed in units of ms. The ET value is displayed in units of ms. Setting range = 0 to 2147483647 x 1 ms
10ms	Specify the time in units of 10 ms.	The PT value is set and displayed in units of 10 ms. The ET value is displayed in units of 10 ms. Setting range = 0 to 214748364 x 10 ms
0.1s	Specify the time in units of 0.1 s.	The PT value is specified and displayed in units of 0.1 s. The ET value is displayed in units of 0.1 s. Setting range = 0 to 21474836 x 100 ms
s	Specify the time in units of 1 s.	The PT value is specified and displayed in units of 1 s. The ET value is displayed in units of 1 s. Setting range = 0 to 2147483 x seconds

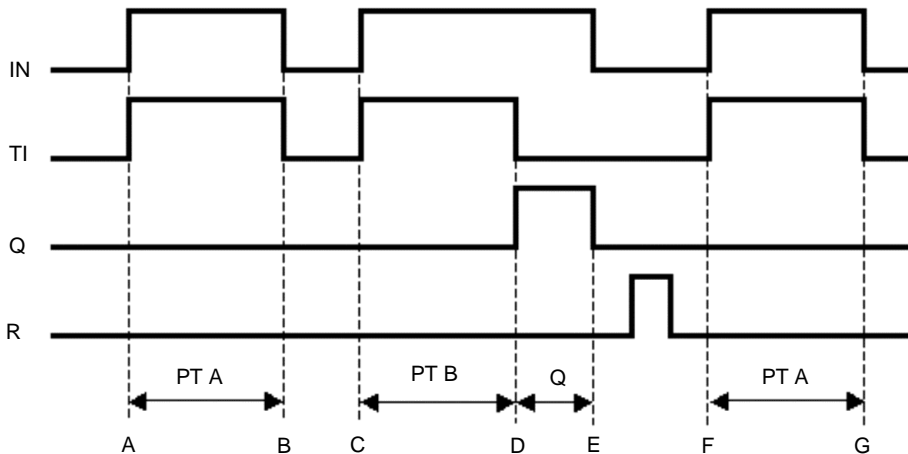
Program Example

TONA



- (1) When the variable of the normally open instruction is turned ON, the elapsed time .ET will increase by the specified time-based units because the TONA instruction is triggered.
 - The timer measurement bit .TI turns ON.
 - The timer output bit .Q turns OFF.
- (2) When the elapsed time .ET increases to equal the preset time .PT, the elapsed time .ET keeps the current value.
 - The timer measurement bit .TI turns OFF.
 - The timer output bit .Q turns ON and allows power to pass.
- (3) When the TONA instruction turns OFF, the elapsed time .ET keeps the current value.
 - The timer measurement bit .TI turns OFF.
 - The timer output bit .Q turns OFF.
- (4) The TONA instruction acts like an accumulator and increases its value. Set the R coil ON to reset the current value.

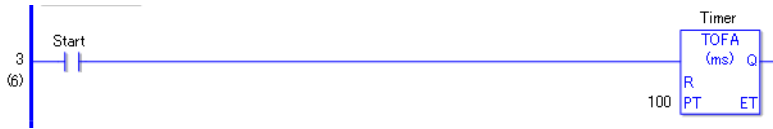
◆ Timing Chart for the TONA Instruction



- Points A, F The timer input bit IN turns ON and the timer measurement bit TI turns ON. The timer starts and the elapsed time ET increases. The timer output bit Q remains OFF.
- Points B, G The timer input bit IN turns OFF, and if the elapsed time ET is less than the preset time PT, the timer output bit Q remains OFF. The elapsed time ET is in the keep state.
- Point C The timer input bit IN turns ON and the timer measurement bit TI turns ON. The timer measurement starts again and the elapsed time ET is added to the kept value. The timer output bit Q remains OFF.

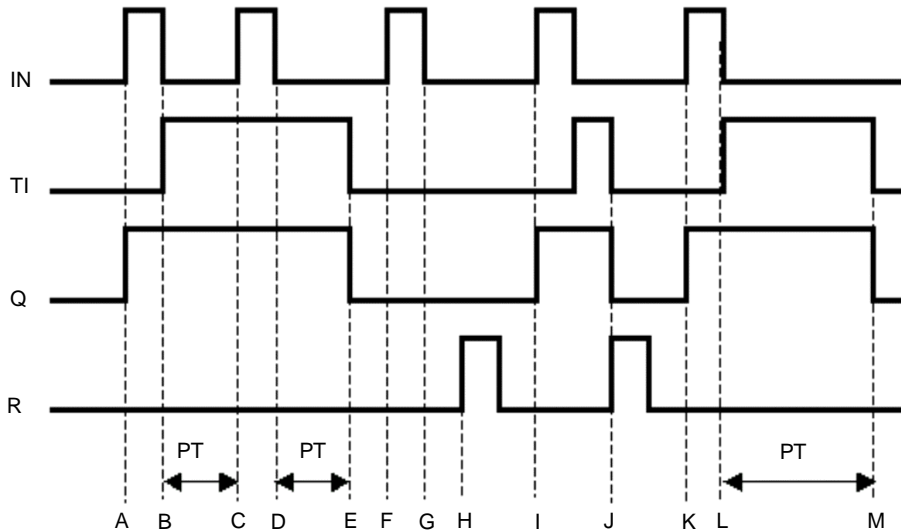
- Point D When the elapsed time ET reaches the preset time PT, the timer measurement bit TI turns OFF.
 The timer output bit Q turns ON.
- Point E The timer input bit IN turns OFF and the timer output bit Q turns OFF. Reset the elapsed time ET to zero using the reset bit (R).

◆ Operational Example of the TOFA Instruction



- (1) When the timer turns OFF, because the TOFA instruction is triggered, the elapsed time .ET increases in the specified time-based units.
 - The timer measurement bit .TI turns ON.
 - The timer output bit .Q turns OFF.
- (2) When the elapsed time .ET increases to equal the preset time .PT, the elapsed time .ET keeps the current value.
 - The timer measurement bit .TI turns OFF.
 - The timer output bit .Q turns ON and allows power to pass.
- (3) When the TONA instruction turns OFF, the elapsed time .ET keeps the current value.
 - The timer measurement bit .TI turns OFF.
 - The timer output bit .Q turns OFF.

◆ Timing Chart for the TOFA Instruction



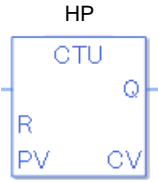
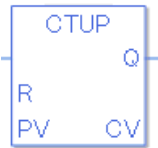
- Point A When IN (input) turns ON, Q (output) turns ON.
- Point B When IN (input) is OFF, TI (timer measurement) turns ON. When TI turns ON, the timer measurement starts.
- Point C When IN (input) turns ON, the timer measurement pauses.
- Point D When IN (input) turns OFF, the paused timer measurement continues.
- Point E When the preset time (PT) value has increased to the point that PT equals ET, TI (timer measurement) and Q (output) turn OFF.
- Points F, G Even if IN (input) turns either on or off, Q (output) and TI (timer) do not turn on.
- Point H Turning ON R resets the timer. The timer is reset when an upward transition is detected.
- Point I When IN (input) turns ON, Q (output) turns ON.
- Point J When R (reset) turns on, Q (output) and TI (timer) are reset. The timer current value ET is also reset and cleared to 0.
- Point K When IN (input) turns ON, Q (output) turns ON.
- Point L When IN (input) turns OFF, TI (timer measurement) turns ON. When TI turns ON, the timer measurement starts.
- Point M When the timer setting (PT) value has increased so that PT equals ET, TI (timer measurement) and Q (output) turn OFF.

(1) If any numeric value out of the setting range is input, an error will occur and the “6706” error code will be set for #L_CalcErrCode. For checking error details, refer to the #L_CalcErrCode. If any value out of the setting range is input, the instruction is not executed.

29.5.5 Counter Instruction

■ CTU and CTUP (Up Counter)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
CTU (Up Counter - Level Sensitive)		Counter	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
CTUP (Up Counter - positive transition)		Counter	2

◆ Explanation of the CTU and CTUP Instructions

Counter variables in CTU and CTUP instructions are structure variables. The following table lists the internal structures.

Counter Variable

Counter Variable	Variables Settings	Description
Variable name.R	Bit Variable	Resets the current value. Clear (0).
Variable name.Q	Bit Variable	Turns ON when the current value reaches the preset value.
Variable name.UP	Bit Variable	Counts up when the variable is ON.
Variable name.QU	Bit Variable	For Up/Down counters, turns ON when the current value reaches the preset value.
Variable name.QD	Bit Variable	For Up/Down counters, turns ON when the current value reaches 0 or less.
Variable name.PV	Integer Variable	Preset value
Variable name.CV	Integer Variable	Current value

When CTU and CTUP instructions are executed, if the counter reset bit variable .R is OFF, regardless of the preset value .PV, the current value .CV will increase by 1. When the current value .CV equals the preset value .PV, the counter output bit variable .Q turns ON. When the counter reset bit variable .R is ON, the current value .CV is reset to 0 and the counter output bit variable .Q also turns OFF.

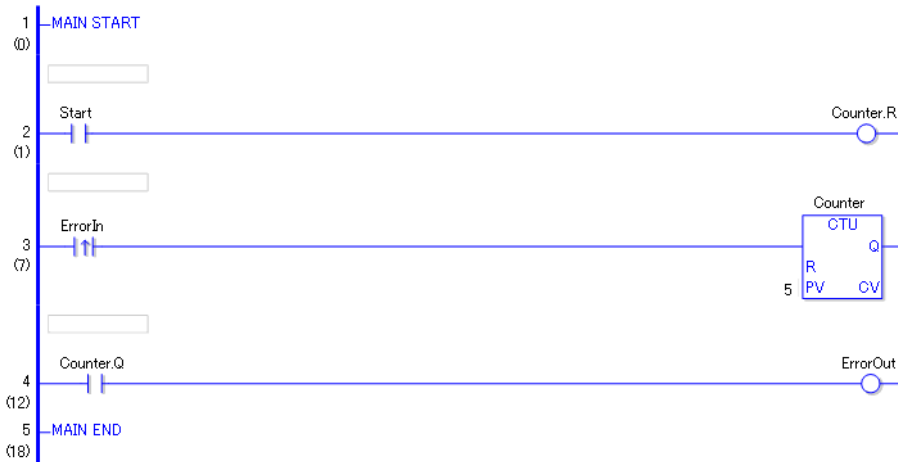
Program Example

CTU

In the following example, if five operation errors are counted within 1 minute, an error will be displayed.

In the program example, the timer instruction is not shown. Only the one-minute timer start trigger for timer start is shown.

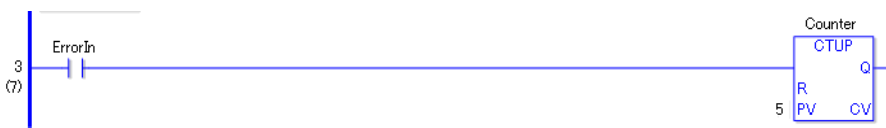
To count operation errors, create a separate error input trigger.



- (1) When the normally open instruction of the one-minute timer turns ON, the OUT instruction assigned to counter .R (reset) turns ON.
When the operation error counter .R (reset) turns ON, the operation error counter .CV of the CTU instruction is cleared to zero.
- (2) When the positive transition normally open instruction in rung 3 turns ON, the operation error counter .CV value (current value) increases by 1.
- (3) When the operation error counter .CV value (current value) equals the .PV value (preset value), the operation error counter .Q of the CTU instruction turns ON , and the OUT instruction in rung 4 outputs the error detection message.

Program Example

CTUP



The difference between CTU and CTUP instructions is whether the .CV value increases as a level counter, or as a positive transition counter.

The difference in program creation is that the positive transition normally open instruction located on rung 3 to detect operation errors is a normally open instruction.

There is no difference in operation other than how the input is determined.

■ CTD and CTDTP (Down Counters)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
CTD (Down Counter - Level Sensitive)		Counter	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
CTDTP (Down Counter - positive transition)		Counter	2

◆ Explanation of the CTD and CTDTP Instructions

Counter variables in CTD and CTDTP instructions are structure variables. The following table lists the internal structures.

Counter Variable

Counter Variable	Variables Settings	Description
Variable name.R	Bit Variable	Resets the current value. Clear (0).
Variable name.Q	Bit Variable	Turns ON when the current value reaches the preset value.
Variable name.UP	Bit Variable	Counts up when the variable is ON.
Variable name.QU	Bit Variable	For Up/Down counters, turns ON when the current value reaches the preset value.
Variable name.QD	Bit Variable	For Up/Down counters, turns ON when the current value reaches 0 or less.
Variable name.PV	Integer Variable	Preset value
Variable name.CV	Integer Variable	Current value

When the CDT and CDTP instructions are ON, if the counter reset bit variable .R is OFF, the current value .CV will decrease by 1.

When the current value .CV is less than 0, the counter output bit .Q turns ON. When the counter reset bit variable .R turns ON, the preset value .PV is copied to the current value variable .CV. And, the counter output variable .Q turns OFF.

NOTE

- When the preset value of the down counter is set to zero or less, output continues to remain on. Be sure to set the preset value to 1 or more.

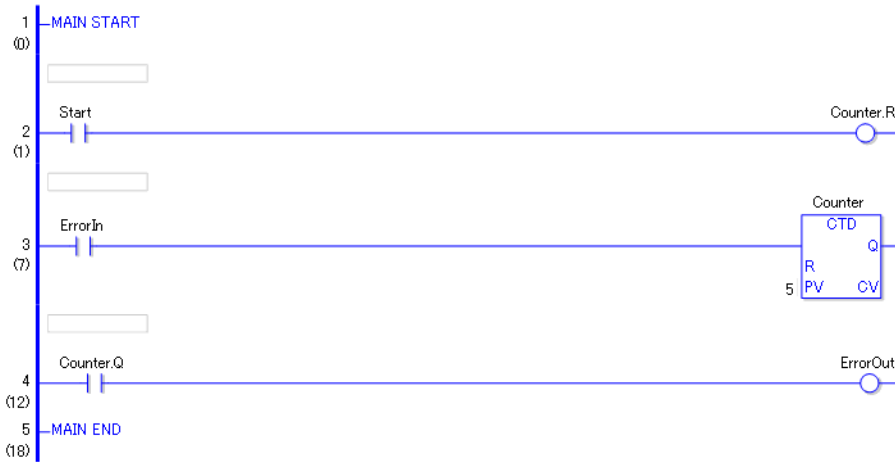
Program Example

CDT

In the following example, if five operation errors are counted within 1 minute, an error will be displayed.

In the program example, the timer instruction is not shown. Only the one-minute timer start trigger for timer start is shown.

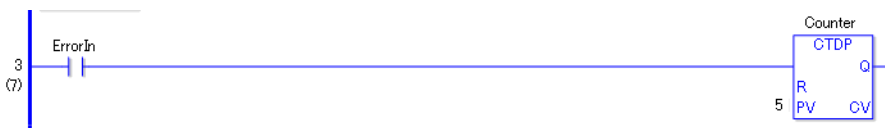
To count operation errors, create a separate error input trigger.



- (1) When the normally open instruction of the one-minute timer turns ON, the OUT instruction assigned to counter .R (reset) turns ON. When the operation error counter .R (reset) turns ON, the CDT instruction's preset value .PV is copied to the current value .CV. In the program example, 5 is copied to the current value .CV.
- (2) When the positive transition normally open instruction turns ON, the operation error counter .CV value (current value) decreases by 1.
- (3) When the value of the operation error counter .CV value (current value) is 0 or less, the operation error counter .Q of the CDT instruction turns ON, and the OUT instruction in rung 4 outputs the error detection message.

Program Example

CTDP



The difference between CTD and CTDP instructions is one counts down when it detects a level change and the other counts down when it detects a positive transition.

The difference in program creation is that the positive transition normally open instruction located on rung 3 to detect operation errors is a normally open instruction.

There is no difference in operation other than how the input is determined.

■ **CTUD and CTUDP (Up/Down Counters)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
CTUD (Up/Down Counter - Level Sensitive)		Counter	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
CTUDP (Up/Down counter - positive transition)		Counter	2

◆ **Explanation of the CTUD and CTUDP Instructions**

Counter variables in CTUD and CTUDP instructions are structure variables. The following table lists the internal structures.

Counter Variable

Counter Variable	Variables Settings	Description
Variable name.R	Bit Variable	Resets the current value. Clear (0).
Variable name.Q	Bit Variable	Turns ON when the current value reaches the preset value.
Variable name.UP	Bit Variable	Counts up when the variable is ON.
Variable name.QU	Bit Variable	For Up/Down counters, turns ON when the current value reaches the preset value.
Variable name.QD	Bit Variable	For Up/Down counters, turns ON when the current value reaches 0 or less.
Variable name.PV	Integer Variable	Preset value
Variable name.CV	Integer Variable	Current value

When the .UP bit of CTUD and CTUDP instructions is ON, they operate the same as CTU instructions. When the .UP bit is OFF, CTUD and CTUDP instructions operate the same as CTD instructions.

When .UP is ON (counts up) and if .CV (current value) is larger than .PV (preset value), .Q turns ON when the current value reaches the preset value and .QU turns ON.

When .UP is OFF (counts down) is OFF and .CV (current value) is 0 or less, then .Q turns ON when the current value reaches the preset value and .QD turns ON.

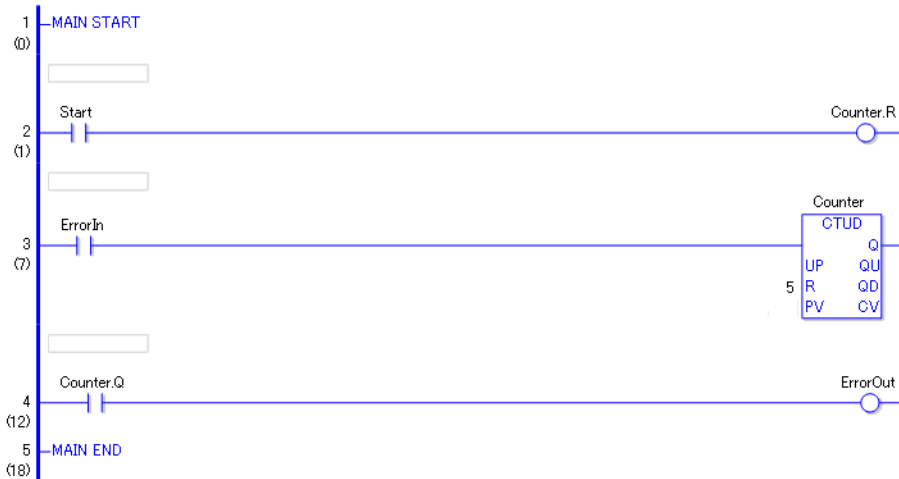
Program Example

CTUD

In the following example, if five operation errors are counted within 1 minute, an error will be displayed.

In the program example, the timer instruction is not shown. Only the one-minute timer start trigger for timer start is shown.

To count operation errors, create a separate error input trigger.



- (1) When the normally open instruction of the one-minute timer turns ON, the OUT instruction assigned to counter .R (reset) turns ON.
When the operation error counter .R (reset) turns ON, if .UP is ON, the CTU instruction is executed, and .CV (current value) is cleared to zero. If .UP is OFF, the CTD instruction is executed, and .PV (preset value) is copied to .CV (current value).
- (2) When the positive transition normally open instruction in rung 3 turns ON, and if .UP is ON, the .CV value increases by 1. If .UP is OFF, the .CV value (current value) decreases by 1.
- (3) When .UP is ON, and the .PV value (preset value) and the .CV value become equal, .Q and .QU turn ON. When .UP is OFF, and the .CV value (current value) is less than 0, Q and .QD turn ON.
- (4) The operation error counter .Q of the CTUD instruction (turns ON when the current value reaches the preset value) turns ON and the OUT instruction outputs the error detection message.

Program Example

CTUDP





The difference between CTUD and CTUDP instructions is whether the .CV value increases or decreases as a level counter, or as a positive transition counter. The difference in program creation is that a positive transition normally open instruction located on rung 3 to detect operation errors is a normally open instruction. There is no difference in operation other than how the input is determined.

29.5.6 R/W Instructions

■ JRD and JRDP (Time Read)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JRD (Time Read - Level Sensitive)		Read	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JRDP (Time Read - positive transition)		Read	2

◆ Explanation of the JRD and JRDP Instructions

Time variables in JRD and JRDP instructions are structure variables. The following table lists the internal structures.

Time Variable

Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD.
VariableName.MIN	Integer Variable	Minutes are input in BCD.
VariableName.SEC	Integer Variable	Seconds are input in BCD.

When JRD and JRDP instructions receive power, the current time will be stored in the variable in D1. The stored time variable can be extracted into hours, minutes and seconds by specifying the structure element. When the time 12:10:45 is stored in the time variable D1, the .HR time is 12, the .MIN time is 10, and the .SEC time is 45.

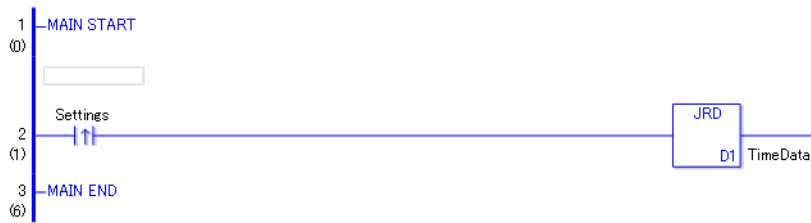
◆ Confirming Execution Results

- (1) If you input a value outside the defined range, an error occurs and the error code (6706) is written to #L_CalcErrCode. When troubleshooting, always check the error code in #L_CalcErrCode.
- (2) #L_CalcZero turns on when the value of D1 is 00:00:00.

Program Example

JRD

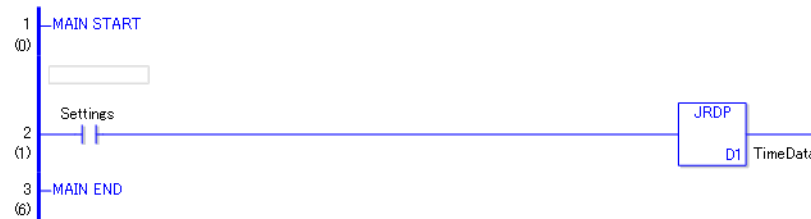
Stores the current time in the time variable.



(1) When the positive transition instruction turns ON, a JRD instruction will execute. When the JRD instruction is executed, the current time is stored in D1.

Program Example



JRDP



(1) When the normally open instruction turns ON, the JRDP instruction will execute. When the JRDP instruction is executed, the current time is stored in D1.

■ JSET and JSETP (Time Set)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JSET (Time Set - Level Sensitive)		Settings	6
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JSETP (Time Set positive transition)		Settings	6

◆ Explanation of the JSET and JSETP Instructions

Time variables used in JSET and JSETP instructions are structure variables. The following table lists the internal structures.

Time Variable

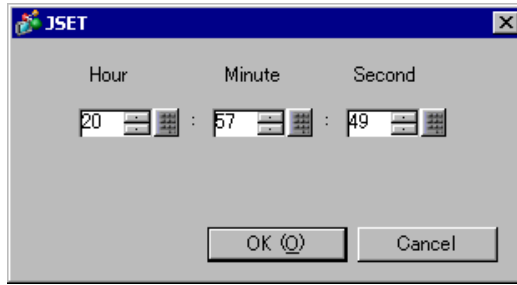
Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD.
VariableName.MIN	Integer Variable	Minutes are input in BCD.
VariableName.SEC	Integer Variable	Seconds are input in BCD.

When JSET and JSETP instructions receive power, the specified time will be stored in the time variable. To set the time, use JSET and JSETP instructions. The time variable in D1 can be extracted into hours, minutes, and seconds by specifying structure elements.

When the current time 12:10:45 is stored in D1, the values 12, 10, and 45 are stored in .HR, .MIN and .SEC, respectively.

◆ **Time Set Dialog Box**

Double-click JSET and JSETP instructions to display a dialog box for setting the time.



In the above dialog box, specify the desired time in hours, minutes and seconds.
Setting Range

Hour	0 - 23
Minute	0 - 59
Second	0 - 59

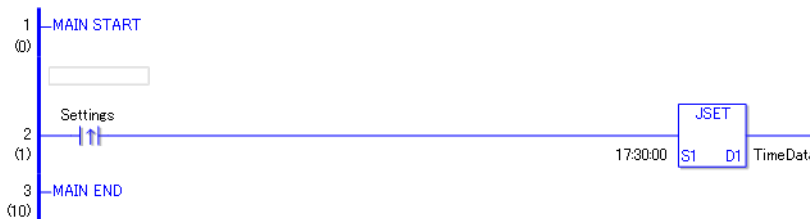
◆ **Confirming Execution Results**

- (1) If you input a value outside the defined range, an error occurs and the error code (6706) is written to #L_CalcErrCode. When troubleshooting, always check the error code in #L_CalcErrCode.
- (2) #L_CalcZero turns on when the value of D1 is 00:00:00.

Program Example

JSET

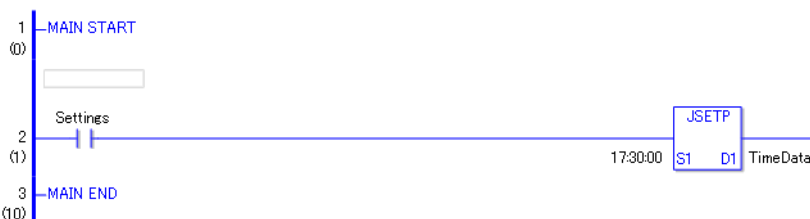
Stores the defined time in the time variable.



- (1) When the positive transition instruction turns ON, the JSET instruction will execute. When the JSET instruction executes, the defined time 17:30:01 is stored in the time variable in D1.

Program Example



JSETP



(1) When the normally open instruction turns ON, the JSETP instruction will execute. When the JSETP instruction executed, the defined time 17:30:00 is stored in the time variable in D1.

■ **NRD and NRDP (Date Read)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NRD (Date Read - Level Sensitive)		Read	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NRDP (Date Read - positive transition)		Read	2

◆ **Explanation of the NRD and NRDP Instructions**

The date variables used in the NRD and NRDP instructions are structure variables. The following table lists the internal structures.

Date Variable

Date Variable	Variables Settings	Description
VariableName.YR	Integer Variable	The year is input in BCD.
VariableName.MO	Integer Variable	The month is input in BCD.
Variable name DAY	Integer Variable	The day is input in BCD.

When NRD and NRDP instructions receive power, the current time is stored in D1. You can extract the year/month/date of the date variable by specifying a particular date element. When the current date 2005/10/20 is stored in D1, 05, 10, and 20 are stored in .YR, .MO, and .DAY, respectively.

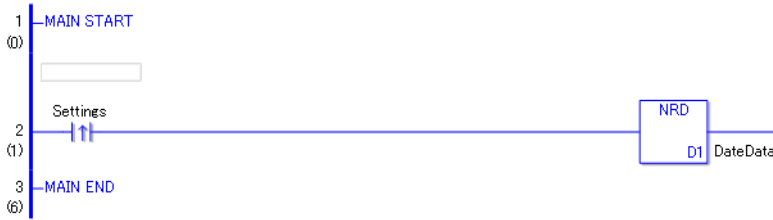
◆ **Confirming Execution Results**

- (1) If any numeric value out of the setting range is input, an error will occur and the “6706” error code is set for #L_CalcErrCode. For checking error details, refer to the #L_CalcErrCode.

Program Example

NRD

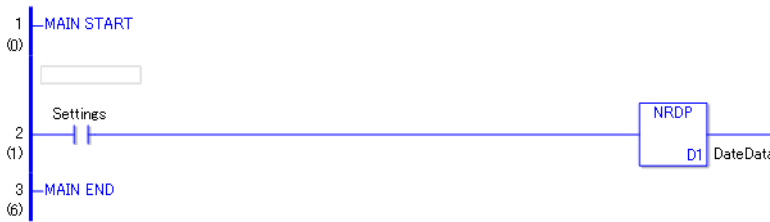
Stores the current date in the date variable.



- (1) When the positive transition instruction turns ON, the NRD instruction is executed. When the NRD instruction is executed, the current date is stored in the date variable in D1.

Program Example



NRDP



- (1) When the normally open instruction turns ON, the NRDP instruction will be executed. When the NRDP instruction is executed, the current date is stored in the date variable in D1.

■ **NSET and NSETP (Date Set)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NSET (Date Set - Level Sensitive)		Settings	5
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NSETP (Date Set - positive transition)		Settings	5

◆ **Explanation of the NSET and NSETP Instructions**

The date variables used in the NSET and NSETP instructions are structure variables. The following table lists the internal structures.

Date Variable

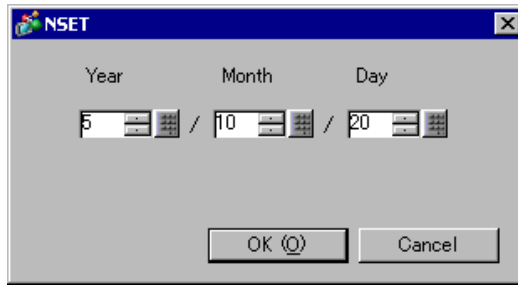
Date Variable	Variables Settings	Description
VariableName.YR	Integer Variable	The year is input in BCD.
VariableName.MO	Integer Variable	The month is input in BCD.
VariableName.DAY	Integer Variable	The day is input in BCD.

When the NSET and NSETP instructions receive power, the specified date will be stored in the date variable. To set the date, use NSET and NSETP instructions. The date variable in D1 can be extracted into hours, minutes, and seconds by specifying structure elements.

When the date 2005/10/20 specified in the JSET instruction is stored in D1, 05, 10, and 20 are stored in .YR, .MO, and .DAY, respectively.

◆ **Date Set Dialog Box**

Double-click the NSET and NSETP instructions to display the dialog box for setting the date.



In the above dialog box, enter the desired date in years, months and days.

Setting Range

Year 0 - 99

Month 1 - 12

Day 1 - 31 (The range depends on the month. Leap years can be specified. Example: February 2008 has 29 days.)

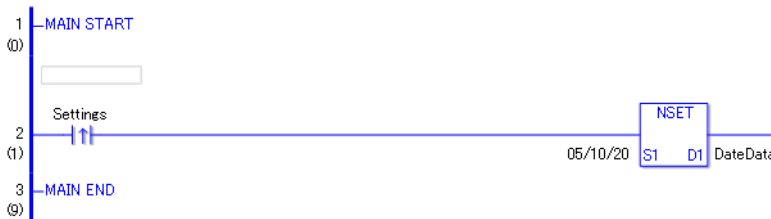
◆ **Confirming Execution Results**

(1) If any numeric value out of the setting range is input, an error will occur and the “6706” error code is set for #L_CalcErrCode. For checking error details, refer to the #L_CalcErrCode.

Program Example

NSET

Stores the setup date in the date variable.



(1) When the positive transition instruction turns ON, the NSET instruction will be executed. When the NSET instruction is executed, the date 10 (month) 20 (day), 2005 specified in the dialog box is stored in the date variable in D1.

Program Example

NSETP

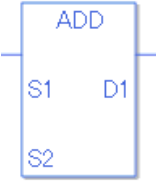
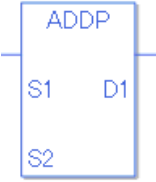


- (1) When the normally open instruction turns ON, the NSETP instruction will be executed. When the NSETP instruction is executed, the date 10 (month) 20 (day), 2005 specified in the dialog box is stored in the date variable in D1.

29.5.7 Operation (Arithmetic)

■ ADD and ADDP (Add)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ADD (Add - Level Sensitive)		Operation	4 to 13
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ADDP (Add - positive transition)		Operation	4 to 13

◆ Operand Settings

The following describes the specifiable content of operands (S1, S2, and D1) for the ADD and ADDP instructions.

The actual number of steps in the ADD and ADDP instructions depends on how operand values are specified. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in ADD and ADDP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



{Data_1 = 1 step} + {Data_2 [0] = 2 steps} + {Result [Specify indirectly] = 3 steps} + {1 steps} = 7 steps.

The last 1 step is for the instruction. Make sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1) and (S2) in the ADD and ADDP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
	U_	.KP / .TR / .TD / .PA / .BA / .ST only	2	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Constant	Integer	-2147483648 to 2147483647	1	O
	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	1	O
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the ADD and ADDP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (only output)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O
	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
Date	.YR/ .MO/ .DAY only	2	O	
Time	.HR/ .MIN/ .SEC only	2	O	
PID	.KP / .TR / .TD / .PA / .BA / .ST only	2	O	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	1	O	
	D_	Modifiers are not specified		1	O
		D_****.B/W [constant]		2	O
		D_****.B/W [address]		3	O
	F_	—	1	O	
	R_	—	1	O	
	T_	.PT / .ET only	2	O	
	C_	.PV / .CV only	2	O	
	N_	.YR / .MO / .DAY only	2	O	
	J_	.HR / .MIN / .SEC only	2	O	
U_	.KP / .TR / .TD / .PA / .BA / .ST only	2	O		
Constant	—	—	—	X	

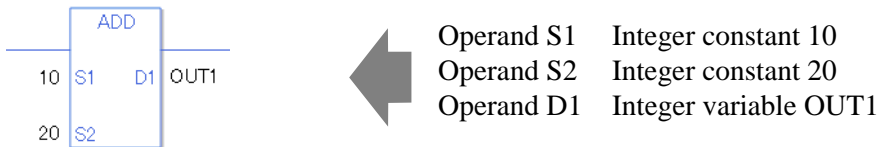
◆ **Explanation of the ADD and ADDP Instructions**

The ADD and ADDP instructions are add instructions. When an ADD instruction is executed, S1 will be added to S2 and the result is stored in D1.

The ADD and ADDP instructions always pass power. When using ADD and ADDP instructions, if variables specified in operands S1, S2, and D1 are not the same type, an error will occur. Specify the same variable type in operands S1, S2, and D1.

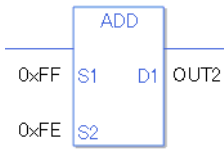
Refer to the following for specifying a constant.

When operand D1 is an integer variable



When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

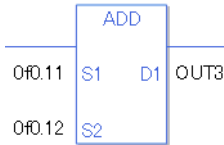
When 0x (zero and lower case "x") is input, the following values will be interpreted as hexadecimal values.



Operand S1 Integer constant 0xFF
 Operand S2 Integer constant 0xFE
 Operand D1 Integer variable OUT2

When operand D1 is a float variable

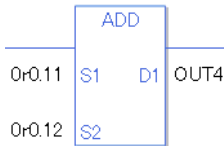
When 0f (zero and lower case "f") is input, the following values will become float values.



Operand S1 Float constant 0.11
 Operand S2 Float constant 0.12
 Operand D1 Float variable OUT3

When operand D1 is a real variable

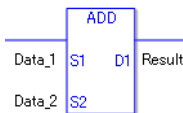
When 0r (zero and lower case "r") is input, the following values will become real values.



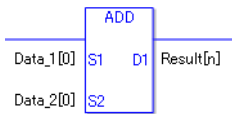
Operand S1 Real constant 0.11
 Operand S2 Real constant 0.12
 Operand D1 Real constant OUT4

When adding the specified array data (integer variable array), specify the array using data [0] or data [N] (N indicates an integer variable).

When all operands S1, S2, and D1 specify the entire array, an error will occur even if the specified variables are the same type.



Data_1 Array size = 5
 Data_2 Array size = 5
 Result Array size = 5
 The operand specification in the left diagram results in an error.



Data_1[0] Array size = 5
 Data_2[0] Array size = 5
 Result[n] Array size = 5
 Arithmetic operations are performed only on individually specified arrays.

◆ **Confirming Execution Results**

- (1) If an overflow occurs as a result of the instruction, the system variable (bit) #L_CalcCarry turns on.
- (2) The instruction will not execute if the value in operand S1 or S2 (infinite or non-numeric value) cannot be recognized. The error code (6706) is written to #L_CalcErrCode. The result in D1 maintains the value from the previous successfully run instruction.
- (3) #L_Error turns on, and the error code (6706) is written to #L_CalcErrCode.
- (4) When the result is 0, the system variable #L_CalcZero turns ON.

(Notes)

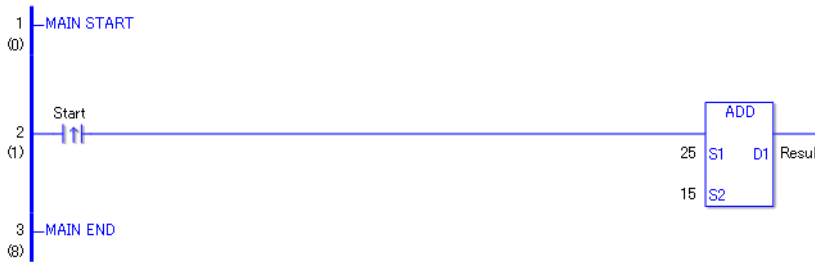
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

ADD

Adds one constant to another and stores the result in the integer variable.

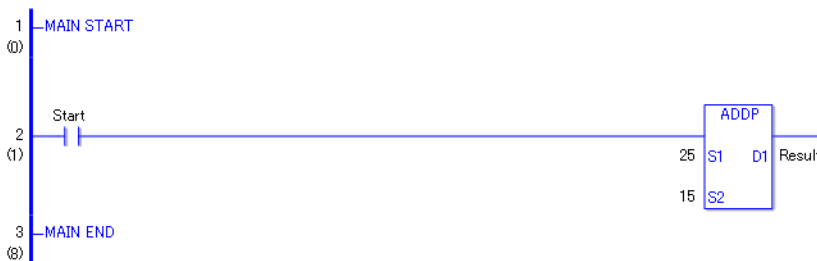


- (1) When the positive transition instruction in the operation turns ON, the ADD instruction will be executed. When the ADD instruction is executed, the result value of 40, obtained from $25 + 15 = 40$, is stored in D1.

When the operation is a normally open instruction, as long as the variable is ON, the ADD instruction is always executed.

Program Example


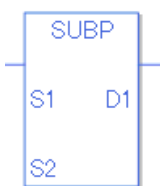
ADDP



(1) When the normally open instruction turns ON, the ADDP instruction will be executed. When the ADDP instruction is executed, the result value of 40, obtained from $25 + 15 = 40$, is stored in D1. Even when the operation is a normally open instruction, only when the upward transition is detected, will the ADDP instruction execute. Therefore, even when the variable of the normally open instruction is always ON, the ADDP instruction is executed only for one scan.

■ **SUB and SUBP (Subtract)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SUB (Subtract - Level Sensitive)		Operation	4 to 13
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SUBP (Subtract - positive transition)		Operation	4 to 13

◆ **Operand Settings**

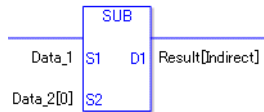
The following describes the specifiable content of the (S1, S2, and D1) operands for the SUB and SUBP instructions.

The actual number of steps in the SUB and SUBP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in SUB and SUBP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



{Data_1 = 1 step} + {Data_2 [0] = 2 steps} + {Result [Specify indirectly] = 3 steps} + {1 steps} = 7 steps.

The last 1 step is for the instruction. Make sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1) and (S2) in the SUB and SUBP instructions

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable].B/W[constant/variable]	4	O
	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
Time	.HR/ .MIN/ .SEC only	2	O	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
U_	.KP / .TR / .TD / .PA / .BA / .ST only	2	O	
Constant	Integer	-2147483648 to 2147483647	1	O
	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	1	O
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the SUB and SUBP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	—	—	—	X

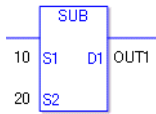
◆ **Explanation of the SUB and SUBP Instructions**

The SUB and SUBP instructions are subtraction instructions. When a SUB instruction is executed, S1 will be subtracted from S2 and the result is stored in D1.

The SUB and SUBP instructions always pass power. When using SUB and SUBP instructions, if the types of variables specified in operands S1, S2, and D1 are not the same, an error will occur. Specify the same variable type for operands S1, S2, and D1.

Refer to the following for specifying a constant.

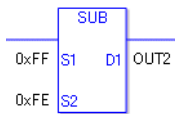
When operand D1 is an integer variable



Operand S1	Integer constant 10
Operand S2	Integer constant 20
Operand D1	Integer constant OUT1

When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

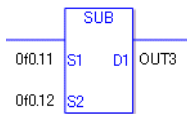
When 0x (zero and lower case "x") is input, the following values will be interpreted as hexadecimal values.



Operand S1	Integer constant 0xFF
Operand S2	Integer constant 0xFE
Operand D1	Integer variable OUT2

When operand D1 is a float variable

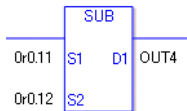
When 0f (zero and lower case "f") is input, the following values will become float values.



Operand S1	Float constant 0.11
Operand S2	Float constant 0.12
Operand D1	Float variable OUT3

When operand D1 is a real variable

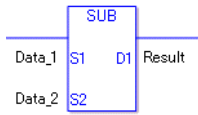
When 0r (zero and lower case "r") is input, the following values will become real values.



Operand S1	Real constant 0.11
Operand S2	Real constant 0.12
Operand D1	Real variable OUT4

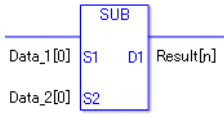
When subtracting specified array data (integer variable array) Specify the array using data [0] or data [N] (N indicates an integer variable).

When all operands S1, S2, and D1 specify the entire array, an error will occur even if the specified variables are the same type.



Data_1 Array size = 5
 Data_2 Array size = 5
 Result Array size = 5

The operand specification in the left diagram results in an error.



Data_1[0] Array size = 5
 Data_2[0] Array size = 5
 Result[n] Array size = 5

Arithmetic operations are performed only on individually specified arrays.

◆ Confirming Execution Results

- (1) If an overflow occurs as a result of the instruction, the system variable (bit) #L_CalcCarry turns on.
- (2) The instruction will not execute if the value in operand S1 or S2 (infinite or non-numeric value) cannot be recognized. For the error check, the error code “6706” is set for the #L_CalcErrCode.
 The output result D1 maintains the value from the previous instruction executed successfully.
- (3) #L_Error turns on, and the error code (6706) is written to #L_CalcErrCode.
- (4) When the result is 0, the system variable #L_CalcZero turns ON.

(Notes)

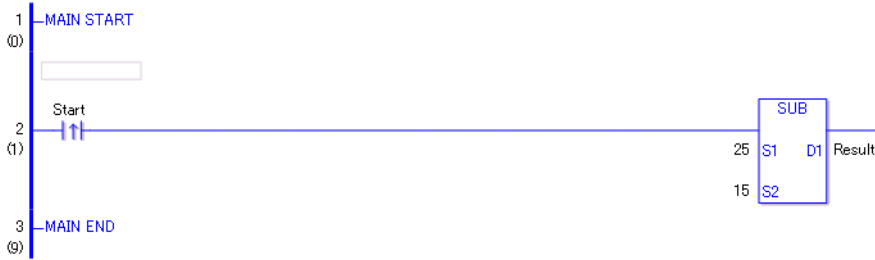
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

SUB

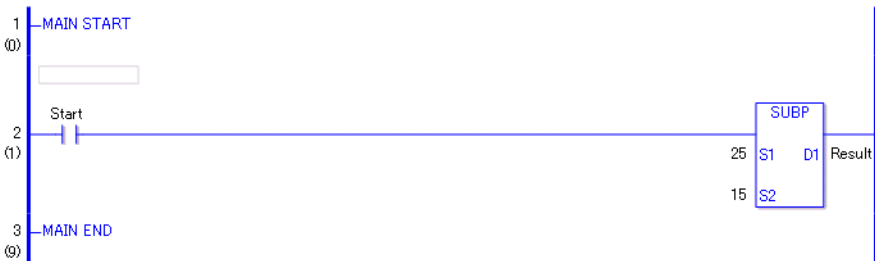
Subtracts one constant from another and stores the result in the integer variable.



- (1) When the positive transition instruction turns ON, the SUB instruction will be executed. When the SUB instruction is executed, the result value of 10, obtained from $25 - 15 = 10$, is stored in D1. When using a normally open instruction, as long as the variable is ON, the SUB instruction is always executed.

Program Example


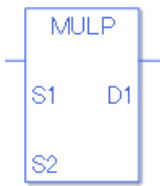
SUBP



- (1) When the normally open instruction turns ON, the SUBP instruction will be executed. When the SUBP instruction is executed, the result value of 10, obtained from $25 - 15 = 10$, is stored in D1. Even when using a normally open instruction, only the upward transition is detected, and the SUBP instruction is executed. Therefore, even when the normally open instruction is always ON, the SUBP instruction is executed only for one scan.

■ MUL and MULP (Multiplication)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
MUL (Multiplication - Level Sensitive)		Operation	4 to 13
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
MULP (Multiplication - positive transition)		Operation	4 to 13

◆ Operand Settings

The following describes the specifiable content of the (S1, S2, and D1) operands for the MUL and MULP instructions.

The actual number of steps in the MUL and MULP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in MUL and MULP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



{Data_1 = 1 step} + {Data_2 [0] = 2 steps} + {Result [Specify indirectly] = 3 steps} + {1 steps} = 7 steps.

The last 1 step is for the instruction. Make sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1) and (S2) in the MUL and MULP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable].B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA / .BA / .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA / .BA / .ST only	2	O	
Constant	Integer	-2147483648 to 2147483647	1	O
	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	1	O
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the MUL and MULP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP / .TR / .TD / .PA / .BA / .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
	U_	.KP / .TR / .TD / .PA / .BA / .ST only	2	O
Constant	—	—	—	X

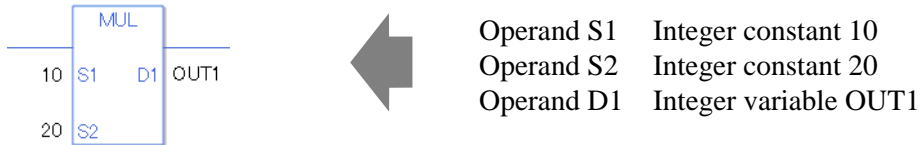
◆ **Explanation of the MUL and MULP Instructions**

The MUL and MULP instructions are multiplication instructions. When a MUL instruction is executed, S1 will be multiplied by S2 and the result is stored in D1.

The MUL and MULP instructions always pass power. When using MUL and MULP instructions, if the types of variables specified in operands S1, S2, and D1 are not the same, an error will occur. Specify the same variable type for operands S1, S2, and D1.

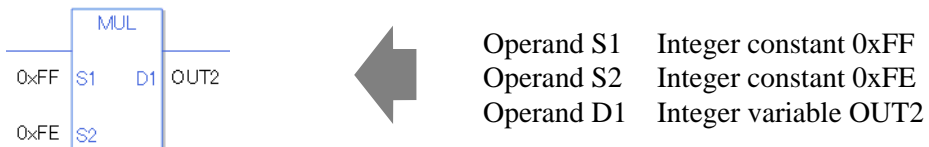
Refer to the following for specifying a constant.

When operand D1 is an integer variable



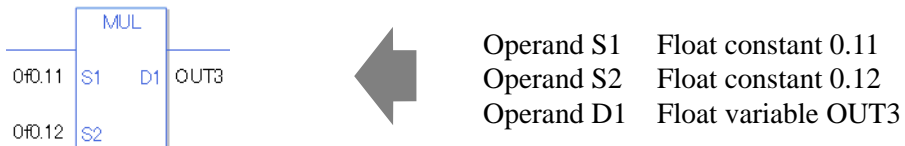
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case "x") is input, the following values will be interpreted as hexadecimal values.



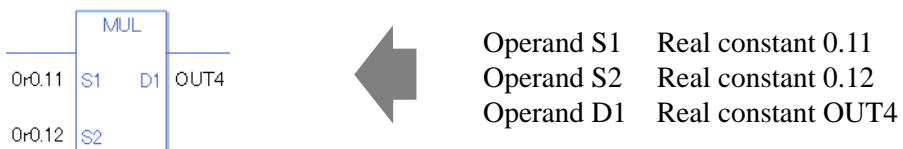
When operand D1 is a float variable

When 0f (zero and lower case "f") is input, the following values will become float values.



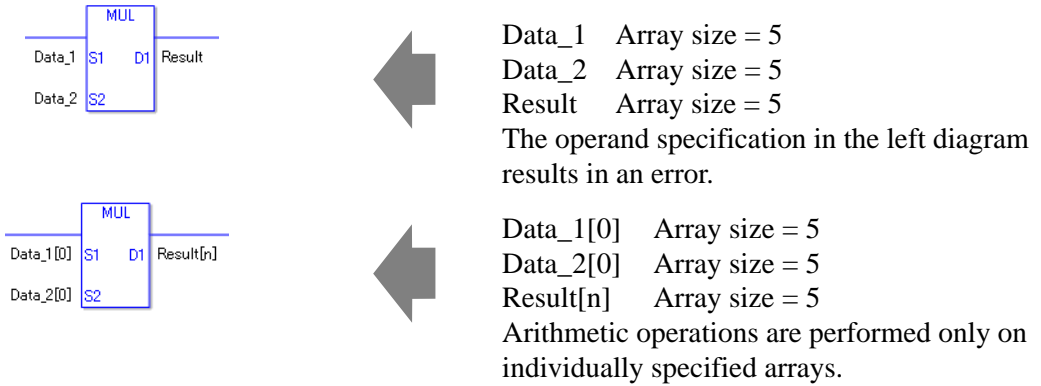
When operand D1 is a real variable

When 0r (zero and lower case "r") is input, the following values will become real values.



When multiplying the specified array data (integer variable array) Specify the array using data [0] or data [N] (N indicates an integer variable).

When all operands S1, S2, and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **Confirming Execution Results**

- (1) If an overflow occurs as a result of the instruction, the system variable (bit) #L_CalcCarry turns on.
- (2) The instruction will not execute if the value in operand S1 or S2 (infinite or non-numeric value) cannot be recognized. For the error check, the error code “6706” is set for the #L_CalcErrCode.
 The output result D1 maintains the value from the previous instruction executed successfully.
- (3) #L_Error turns on, and the error code (6706) is written to #L_CalcErrCode.
- (4) When the execution result is 0, the system variable #L_CalcZero turns ON.

(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

MUL

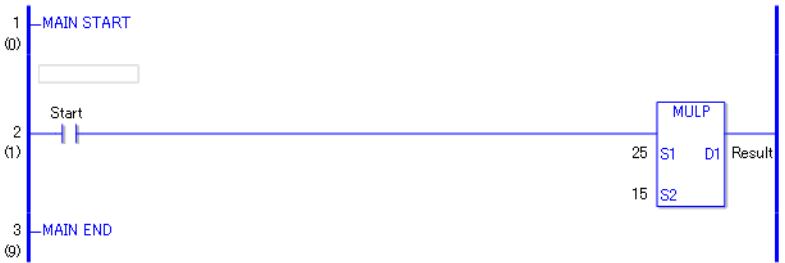
Multiplies one constant by another and stores the result in the integer variable.



- (1) When the positive transition instruction turns ON, the MUL instruction will be executed. When the MUL instruction is executed, the result value 375, obtained from $25 \times 15 = 375$, is stored in D1. When using a normally open instruction, as long as the instruction variable is ON, the MUL instruction is always executed.

Program Example

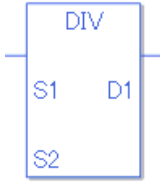
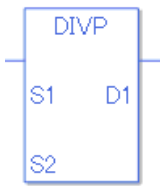
MULP



- (1) When the normally open instruction turns ON, the MULP instruction will be executed. When the MULP instruction is executed, the result value of 10, obtained from $25 \times 15 = 375$, is stored in D1. Even when using a normally open instruction, the MULP instruction executes only when it detects the upward transition. Therefore, even when the variable of the NO instruction is always ON, the MULP instruction is executed only for one scan.

■ **DIV and DIVP (Division)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
DIV (Division - Level Sensitive)		Operation	4 to 13
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
DIVP (Division - positive transition)		Operation	4 to 13

◆ **Operand Settings**

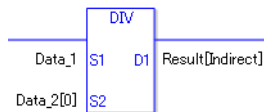
The following describes the specifiable content of the (S1, S2, and D1) operands for DIV and DIVP instructions.

The actual number of steps in the DIV and DIVP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in DIV and DIVP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



{Data_1 = 1 step} + {Data_2 [0] = 2 steps} + {Result [Specify indirectly] = 3 steps} + {1 steps} = 7 steps.

The last 1 step is for the instruction. Make sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1) and (S2) in the DIV and DIVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable].B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA / .BA / .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA / .BA / .ST only	2	O	
Constant	Integer	-2147483648 to 2147483647	1	O
	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	1	O
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the DIV and DIVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP / .TR / .TD / .PA / .BA / .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
U_	.KP / .TR / .TD / .PA / .BA / .ST only	2	O	
Constant	—	—	—	X

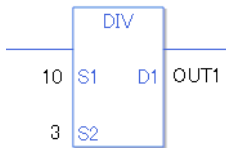
◆ **Explanation of the DIV and DIVP Instructions**

The DIV and DIVP instructions are division instructions. When a DIV instruction is executed, S1 will be divided by S2 and the result is stored in D1.

The DIV and DIVP instructions always pass power. When using the DIV and DIVP instructions, if the types of variables specified in operands S1, S2, and D1 are not the same, an error occurs. Specify the same variable type for operands S1, S2, and D1.

Refer to the following for specifying a constant.

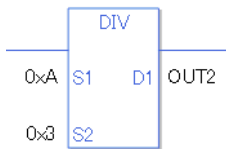
When operand D1 is an integer variable



Operand S1 Integer constant 10
 Operand S2 Integer constant 3
 Operand D1 Integer variable OUT1
 The operation result is rounded off to the nearest integer.

When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

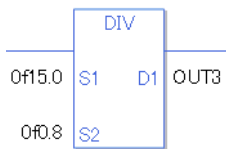
When 0x (zero and lower case "x") is input, the following values will be interpreted as hexadecimal values.



Operand S1 Integer constant 0xA
 Operand S2 Integer constant 0x3
 Operand D1 Integer variable OUT2
 The operation result is rounded off to the nearest integer.
 Example: 0xA (S1) / 0x3 (S2) = 3 (D1)

When operand D1 is a float variable

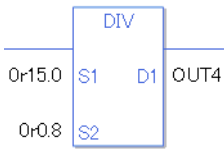
When 0f (zero and lower case "f") is input, the following values will become float values.



Operand S1 Float constant 150
 Operand S2 Float constant 0.8
 Operand D1 Float variable OUT3
 The operation result is a value including the decimal point.
 Example: 0f 15.0 (S1) / 0f 0.8 (S2) = 18.75 (D1)

When operand D1 is a real variable

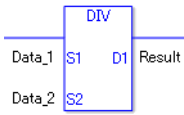
When 0r (zero and lower case "r") is input, the following values will become real values.



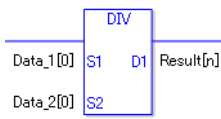
Operand S1 Float constant 150
 Operand S2 Float constant 0.8
 D1 Operand Float variable OUT4
 The operation result is a value including the decimal point.
 For example 0r15.0 (S1) / 0r0.8 (S2) = 18.75 (D1)

When dividing specified array data (integer variable array) Specify the array using data [0] or data [N] (N indicates an integer variable).

When all operands S1, S2, and D1 specify the entire array, an error will occur even if the specified variables are the same type.



Data_1 Array size = 5
 Data_2 Array size = 5
 Result Array size = 5
 The operand specification in the left diagram results in an error.



Data_1[0] Array size = 5
 Data_2[0] Array size = 5
 Result[n] Array size = 5
 Arithmetic operations are performed only on individually specified arrays.

◆ **Confirming Execution Results**

- (1) If an overflow occurs as a result of the instruction, the system variable (bit) #L_CalcCarry turns on.
- (2) The instruction will not execute if the value in operand S1 or S2 (infinite or non-numeric value) cannot be recognized. For the error check, the error code "6706" is set for the #L_CalcErrCode.
 The output result D1 maintains the value from the previous instruction executed successfully.
- (3) #L_Error turns on, and the error code (6706) is written to #L_CalcErrCode.
- (4) When the execution result is 0, the system variable #L_CalcZero turns ON.

(Notes)

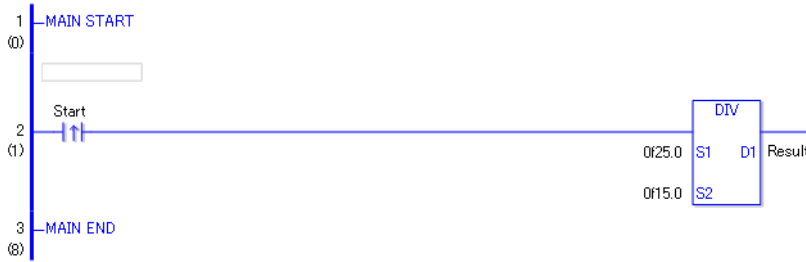
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

DIV

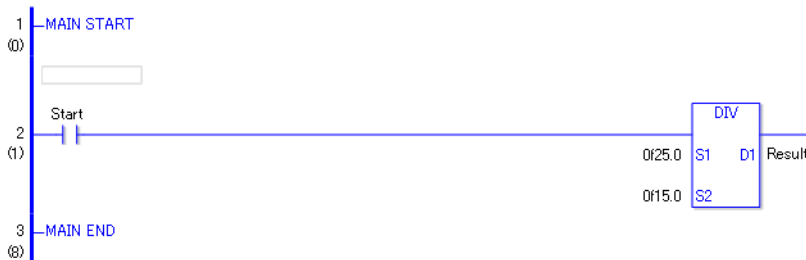
Divides one constant by another and stores the result in the float variable.



- (1) When the positive transition instruction turns ON, the DIV instruction will be executed. When the DIV instruction is executed, the result value of 1.66666..., obtained from $25/15 = 1.66666\dots$, is stored in the result data (float variable) in D1. When the value cannot be divided, it is rounded off to the nearest digit. When using a normally open instruction, as long as the variable for the instruction is ON, the DIV instruction is always executed.

Program Example

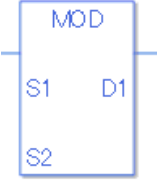
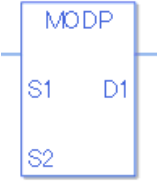
DIVP



- (1) When the normally open instruction turns ON, the DIVP instruction will be executed. When the DIVP instruction is executed, the result value of 1.66666..., obtained from $25 / 15 = 1.66666\dots$, is stored in the result data (float variable) in D1. When the value cannot be divided, it is rounded off to the nearest digit. Even when using a normally open instruction, only the upward transition is detected, and the DIVP instruction is executed. Therefore, even when the instruction is always ON, the DIVP instruction is executed only for one scan.

■ **MOD and MODP (Modules)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
MOD (Modules - Level Sensitive)		Operation	4 to 13
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
MODP (Modules - positive transition)		Operation	4 to 13

◆ **Operand Settings**

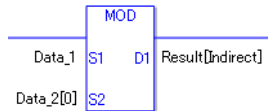
The following describes the specifiable content of operands (S1, S2, and D1) for the MOD and MODP instructions.

The actual number of steps in the MOD and MODP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in MOD and MODP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



{Data_1 = 1 step} + {Data_2 [0] = 2 steps} + {Result [Specify indirectly] = 3 steps} + {1 steps} = 7 steps.

The last 1 step is for the instruction. Make sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1) and (S2) in the MOD and MODP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable].B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA / .BA / .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA / .BA / .ST only	2	O	
Constant	Integer	-2147483648 to 2147483647	1	O
	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the MOD and MODP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

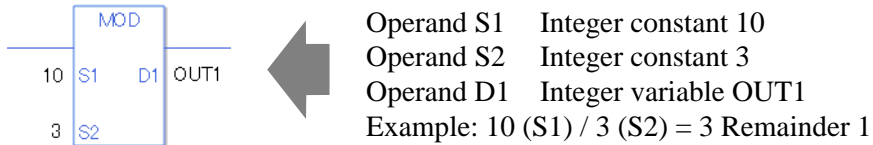
Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP / .TR / .TD / .PA / .BA / .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
U_	.KP / .TR / .TD / .PA / .BA / .ST only	2	O	
Constant	—	—	—	X

◆ **Explanation of the MOD and MODP Instructions**

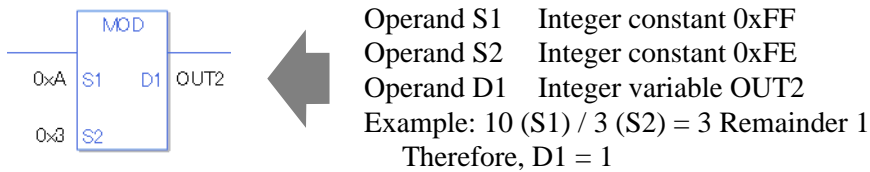
The MOD and MODP instructions are modules operations. When a MOD instruction is executed, S1 will be divided by S2 and the value of the remainder is stored in D1. The MOD and MODP instructions are always conducted. When using the MOD and MODP instructions, if the types of variables specified in operands S1, S2, and D1 are not the same, an error will occur. Specify the same variable type for operands S1, S2, and D1. Refer to the following when specifying a constant.

When operand D1 is an integer variable



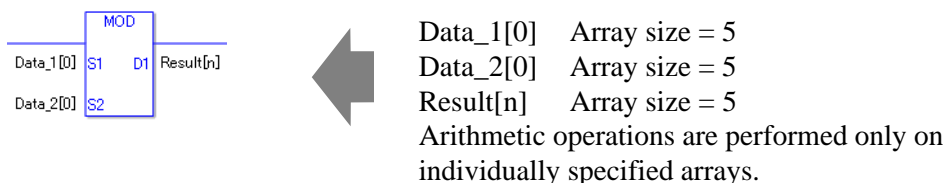
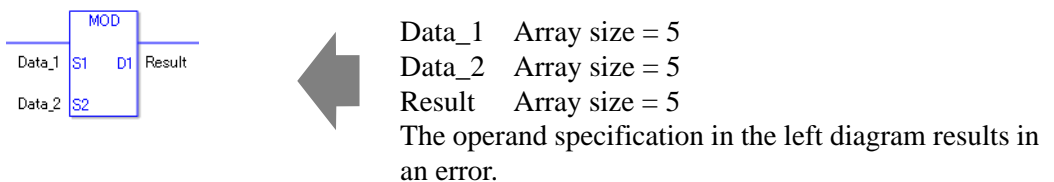
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case "x") is input, the following values will be interpreted as hexadecimal values.



When performing a module operation for specified array data (integer variable array) Specify an array using data [0] or data [N] (N indicates an integer variable).

When all operands S1, S2, and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **Confirming Execution Results**

- (1) If an overflow occurs as a result of the instruction, the system variable (bit) #L_CalcCarry turns on.
- (2) #L_Error turns on, and the error code (6706) is written to #L_CalcErrCode.
- (3) When the execution result is 0, the system variable #L_CalcZero turns ON.

(Notes)

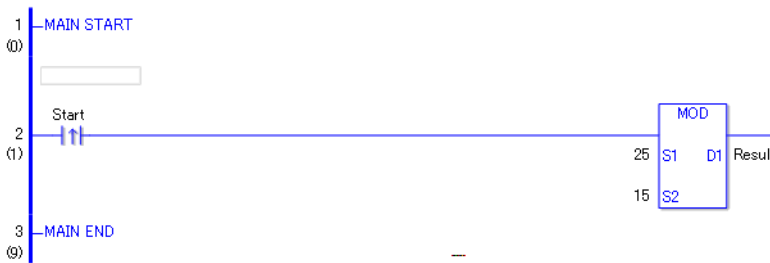
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

MOD

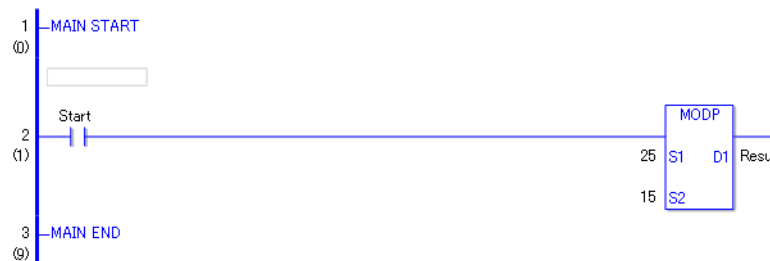
Performs modules operation on two constants and stores the result in the integer variable.



- (1) When the positive transition instruction turns ON, the MOD instruction will be executed. When the MOD instruction is executed, the result value of 10, obtained from $25/15 = 1$ (remainder 10), is stored in D1. When using a normally open instruction, as long as the operation is ON, the MOD instruction is always executed.

Program Example



MODP



- (1) When the normally open instruction start turns ON, the MODP instruction will be executed. When the MODP instruction is executed, the result value of 10, obtained from $25 / 15 = 1$ (remainder 10), is stored in D1. Even when using a normally open instruction, the MODP instruction executes only when it detects the upward transition. Therefore, even when the NO instruction is always ON, the MODP instruction is executed only for one scan.

■ **INC and INCP (Increment)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
INC (Increment - Level Sensitive)		Operation	2 to 4
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
INCP (Increment - positive transition)		Operation	2 to 4

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the INC and INCP instructions.

Name	Type	Condition	Instruction Step Count	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	2	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	2	O
Symbol	Bit	—	—	X
	Word	—	2	O

Continued

Name	Type	Condition	Instruction Step Count	Possible: O Not Possible: X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	2	O
		Specify integer variable[constant]	3	O
		Specify integer variable [Variable]	4	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	3	O
	Counter	.PV/ .CV only	3	O
	Date	.YR/ .MO/ .DAY only	3	O
	Time	.HR/ .MIN/ .SEC only	3	O
PID	.KP / .TR / .TD / .PA / .BA / .ST only	3	O	

Continued

Name	Type	Condition	Instruction Step Count	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	1	O	
	D_	Modifiers are not specified		1	O
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT / .ET only	3	O	
	C_	.PV / .CV only	3	O	
	N_	.YR / .MO / .DAY only	3	O	
	J_	.HR / .MIN / .SEC only	3	O	
U_	.KP / .TR / .TD / .PA / .BA / .ST only	3	O		
Constant	Integer	-2147483648 to 2147483647	—	X	
	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	—	X	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X	

◆ **Explanation of the INC and INCP Instructions**

The INC and INCP instructions are incremental instructions. When an INC instruction is executed, 1 is added to D1.

The INC and INCP instructions always pass power.

◆ **Confirming Execution Results**

- (1) If an overflow occurs as a result of the instruction, the system variable (bit) #L_CalcCarry turns on.
- (2) When the execution result is 0, the system variable #L_CalcZero turns ON.

(Notes)

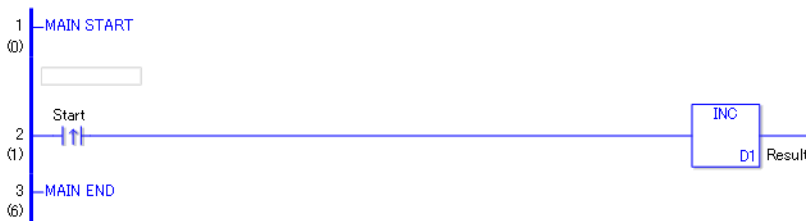
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

INC

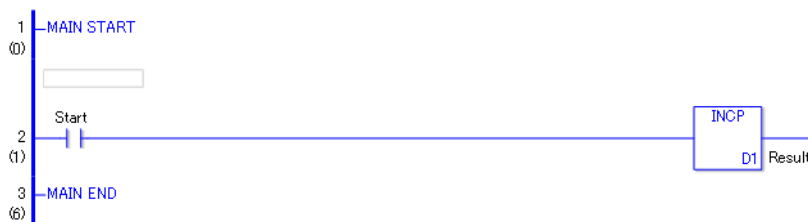
Every time the INC instruction turns on, 1 is added.



- (1) When the positive transition instruction turns on, it passes power and the INC instruction adds 1 to the result data D1 (integer variable). When using a normally open instruction, as long as the instruction is passing power, the INC instruction continually executes adding 1 at each scan.

Program Example



INCP



- (1) When the normally open instruction turns ON, the INCP instruction will be executed. When the INCP instruction is executed, 1 is added to the result data (integer variable) in D1. Even when using a normally open instruction, the INCP instruction executes only when it detects the upward transition. Therefore, even when the instruction is always ON, the INCP instruction is executed only for one scan and 1 is added to the result data (integer variable).

■ DEC and DECP (Decrement)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
DEC (Decrement - Level Sensitive)		Operation	2 to 4
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
DECP (Decrement - positive transition)		Operation	2 to 4

◆ Operand Settings

The following describes the specifiable content of Operand (D1) in the DEC and DECP instructions.

Name	Type	Condition	Instruction Step Count	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	2	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	2	O
Symbol	Bit	—	—	X
	Word	—	2	O

Continued

Name	Type	Condition	Instruction Step Count	Possible: O Not Possible: X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	2	O
		Specify integer variable[constant]	3	O
		Specify integer variable [Variable]	4	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	3	O
	Counter	.PV/ .CV only	3	O
	Date	.YR/ .MO/ .DAY only	3	O
	Time	.HR/ .MIN/ .SEC only	3	O
PID	.KP / .TR / .TD / .PA / .BA / .ST only	3	O	

Continued

Name	Type	Condition	Instruction Step Count	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	1	O	
	D_	Modifiers are not specified		1	O
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT / .ET only	3	O	
	C_	.PV / .CV only	3	O	
	N_	.YR / .MO / .DAY only	3	O	
	J_	.HR / .MIN / .SEC only	3	O	
U_	.KP / .TR / .TD / .PA / .BA / .ST only	3	O		
Constant	Integer	-2147483648 to 2147483647	—	X	
	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	—	X	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X	

◆ **Explanation of the DEC and DECP Instructions**

DEC and DECP instructions are decrement instructions. When a DEC instruction is run, it subtracts 1 from D1.

The DEC and DECP instructions always pass power.

◆ **Confirming Execution Results**

(1) If an overflow occurs as a result of the instruction, the system variable (bit) #L_CalcCarry turns on.

(2) When the execution result is 0, the system variable #L_CalcZero turns ON.

(Notes)

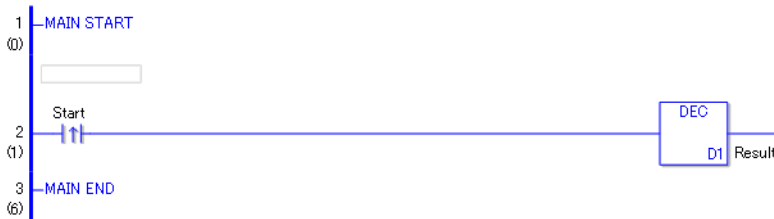
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

DEC

Every time the DEC instruction turns on, 1 is subtracted.



(1) When the positive transition instruction turns on and passes power to the DEC instruction, the DEC instruction subtracts 1 from the result data D1 (integer variable). When using a normally open instruction, as long as the instruction is passing power, the DEC instruction is continually run and subtracts 1 from D1.

Program Example

DECP

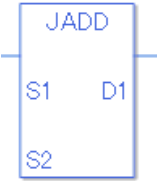
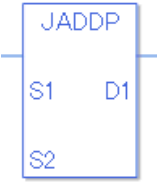


- (1) When the normally open instruction turns ON, the DECP instruction will be executed. When the DECP instruction is executed, 1 is subtracted from D1(integer variable). Even when using a normally open instruction, the DECP instruction executes only when it detects an upward transition. Therefore, even when the operation is continuously ON, the INCP instruction executes for only one scan and 1 is subtracted from D1(integer variable).

29.5.8 Operation (Time)

■ JADD and JADDP (Time Addition)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JADD (Time Addition - Level Sensitive)		Operation	4
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JADDP (Time Addition - positive transition)		Operation	4

◆ Operand Settings

The following describes the specifiable content of Operands (S1, S2, and D1) in the JADD and JADDP instructions.

Name	Type	Condition	Instruction Step Count	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X

Continued

Name	Type	Condition	Instruction Step Count	Possible: O Not Possible: X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	—	X
		Specify integer variable [Variable]	—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	Other than .HR / .MIN / .SEC	4	O
	PID	.KP / .TR / .TD / .PA / .BA / .ST only	—	X

Continued

Name	Type	Condition	Instruction Step Count	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT / .ET only	—	X	
	C_	.PV / .CV only	—	X	
	N_	.YR / .MO / .DAY only	—	X	
	J_	Other than .HR / .MIN / .SEC	4	O	
U_	.KP / .TR / .TD / .PA / .BA / .ST only	—	X		
Constant	Integer	-2147483648 to 2147483647	—	X	
	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	—	X	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X	

◆ **Explanation of the JADD and JADDP Instructions**

The JADD and JADDP instructions are time addition instructions. When a JADD instruction is executed, the time variable in operand S1 will be added to the time variable in S2, and the result of the addition is stored in the time variable in operand D1. The JADD and JADDP instructions always pass power.

Time Variable

Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD
VariableName.MIN	Integer Variable	Minutes are input in BCD
VariableName.SEC	Integer Variable	Seconds are input in BCD

In the JADD instruction, you cannot run time add operations on individual time variable elements (.HR .MIN .SEC).

The time variables and each element thereof are saved as BCD data.

◆ **Confirming Execution Results**

- (1) If the result reaches 00:00' 00" after the instruction, an overflow will occur. The #L_CalcCarry for the system variable (bit) turns on.
- (2) If the operation result is 00 (h):00 (min):00 (s), the system variable #L_CalcZero turns ON.

(Notes)

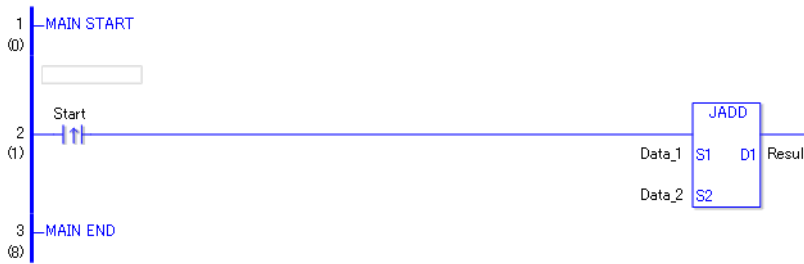
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

JADD

When the positive transition instruction is turned ON, time addition will be performed.

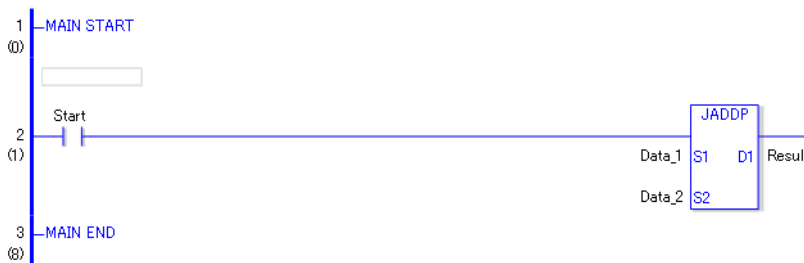


(1) When the positive transition instruction turns on and passes power to the JADD instruction, the JADD instruction adds S1 (Data_1, Time variable) and S2 (Data_2, Time variable) and stores the result in D1 (Time variable). When using a normally open instruction, as long as the instruction is passing power, the JADD instruction continually executes at each scan, performing the time add operation.

Example: When Data_1 in operand S1 is 12:10:45, and Data_2 in operand S2 is 6:55:20, if a JADD instruction is executed, the result is 19:06:05, and 19:06:05 is stored in the result data in operand D1.

Program Example

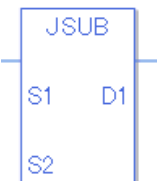
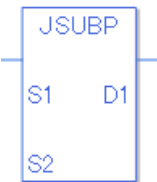
JADDP



(1) When the Normally Open instruction turns ON, the JADDP instruction will be executed. When the JADDP instruction is executed, Data_1 (time variable) in operand S1 is added to Data_2 (time variable) in operand S2, and the result of the addition is stored in operand D1. Even when using a Normally Open instruction, only the upward transition is detected, and the JADDP instruction is executed. Therefore, even when the variable of the NO instruction is always ON, the JADDP instruction is executed only for one scan.

■ JSUB and JSUBP (Time Subtraction)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JSUB (Time Subtraction - Level Sensitive)		Operation	4
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JSUBP (Time Subtraction - positive transition)		Operation	4

◆ Operand Settings

The following describes the specifiable content of Operands (S1, S2, and D1) in the JSUB and JSUBP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	—	X
		Specify integer variable [Variable]	—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
Time	Other than .HR / .MIN / .SEC	4	O	
PID	.KP / .TR / .TD / .PA / .BA / .ST only	—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT / .ET only	—	X	
	C_	.PV / .CV only	—	X	
	N_	.YR / .MO / .DAY only	—	X	
	J_	Other than .HR / .MIN / .SEC	4	O	
U_	.KP / .TR / .TD / .PA / .BA / .ST only	—	X		
Constant	Integer	-2147483648 to 2147483647	—	X	
	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	—	X	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X	

◆ **Explanation of the JSUB and JSUBP Instructions**

The JSUB and JSUBP instructions are time subtraction instructions. When a JSUB instruction is executed, the time variable in operand S2 will be subtracted from the time variable in operand S1, and the result of the subtraction is stored in the time variable in operand D1. The JSUB and JSUBP instructions always pass power.

Time Variable

Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD.
VariableName.MIN	Integer Variable	Minutes are input in BCD.
VariableName.SEC	Integer Variable	Seconds are input in BCD.

In the JSUB instruction, you cannot run time subtract operations on individual time variable elements (.HR .MIN .SEC).

The time variables and each element thereof are saved as BCD data.

◆ **Confirming Execution Results**

(1) If the result does not reach 00:00' 00" after the instruction, an overflow will occur. The #L_CalcCarry for the system variable (bit) turns on.

(2) If the operation result is 00 (h):00 (min):00 (s), the system variable #L_CalcZero turns ON.

(Notes)

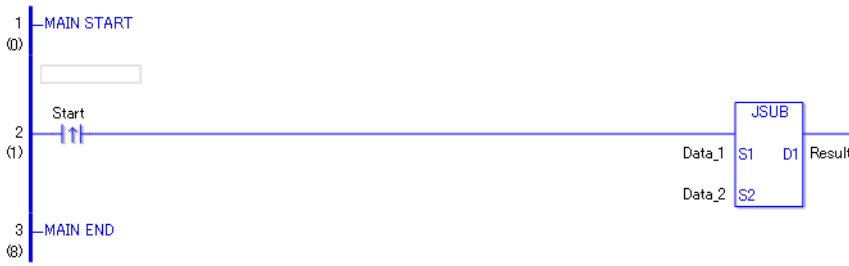
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

JSUB

When the positive transition instruction is turned ON, time subtraction will be performed.

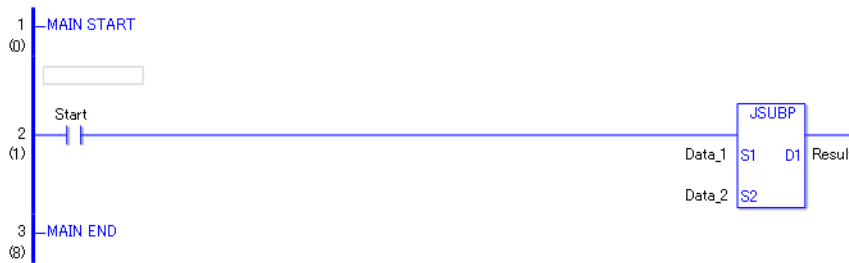


(1) When the positive transition instruction turns on and passes power to the JSUB instruction, the JSUB instruction subtracts S2 (Data_2, Time variable) from S1 (Data_1, Time variable), and stores the result in D1 (Time variable). When using a normally open instruction, on every scan the JSUB instruction receives power, it runs the date/time calculation.

Example: When Data_1 in operand S1 is 12:10:45, and Data_2 in operand S2 is 6:55:20, if a JSUB instruction is executed, the result is 5:15:25, and 5:15:25 is stored in operand D1.

Program Example

JSUBP





(1) When the Normally Open instruction turns ON, the JSUBP instruction will be executed. When the JSUBP instruction is executed, Data_2 (time variable) in operand S2 is subtracted from Data_1 (time variable) in operand S1, and the result of the subtraction is stored in operand D1. Even when using a Normally Open instruction, only the upward transition is detected, and the JSUBP instruction is executed. Therefore, even when the variable of the NO instruction is always ON, the JSUBP instruction is executed only for one scan.

29.5.9 Operation (Logical)

■ AND and ANDP (Logical AND)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
AND (Logical AND - Level Sensitive)		Operation	4 to 13
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ANDP (Logical AND - positive transition)		Operation	4 to 13

◆ Operand Settings

The following describes the specifiable content of Operands (S1, S2, and D1) in the AND and ANDP instructions.

The actual number of steps in the AND and ANDP instructions depends on the specification method of the operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in the S2 operand + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in AND and ANDP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



{Data_1 = 1 step} + {Data_2 [0] = 2 steps} + {Result [Specify indirectly] = 3 steps} + {1 steps} = 7 steps.

The last 1 step is for the instruction. Make sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1 and S2) in the AND and ANDP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/ W[constant] Specify integer array(entire array)	2	O
		Specify integer variable[variable] or Specify integer variable B/ W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA / .BA / .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA / .BA / .ST only	2	O	
Constant	Integer	-2147483648 to 2147483647	1	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the AND and ANDP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant] Specify integer array(entire array)	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT / .ET only	2	O
	C_	.PV / .CV only	2	O
	N_	.YR / .MO / .DAY only	2	O
	J_	.HR / .MIN / .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	-2147483648 to 2147483647	—	X

◆ **Explanation of the AND and ANDP Instructions**

The AND and ANDP instructions are logical AND instructions. When the AND instruction is executed, S1 and S2 will be logically ANDed and the result is stored in D1.

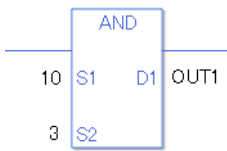
The AND and ANDP instructions always pass power. When using the AND and ANDP instructions, if the types of variables specified in the S1, S2, and D1 operands are not the same type, an error will occur. Specify the same variable type for the S1, S2, and D1 operands.

Refer to the following for specifying a constant.

S1	Operator	S2	D1
OFF	AND	OFF	OFF
ON		OFF	OFF
OFF		ON	OFF
ON		ON	ON

When an AND instruction is executed, the D1 bit will be turned ON only when S1 and S2 are ON. Otherwise, the D1 bit is OFF.

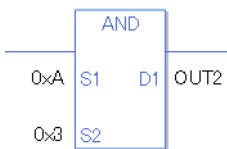
When operand D1 is an integer variable



Operand S1 Integer constant 10
 Operand S2 Integer constant 3
 Operand D1 Integer variable OUT1

When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values will be interpreted as hexadecimal values.



Operand S1 Integer constant 0xA
 Operand S2 Integer constant 0x3
 Operand D1 Integer variable OUT2

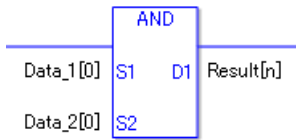
When Calculating Data in a Specified Array (Integer Variable Array)

Specifying the entire array



Data_1 Array size = 5
 Data_2 Array size = 5
 Result Array size = 5
 Logical operations are performed on all arrays.

Individually Specifying Array Variables



Data_1[0] Array size = 5
 Data_2[0] Array size = 5
 Result[n] Array size = 5
 Logical operations are performed on individual variables in the arrays.

◆ **Confirming Execution Results**

(1) When the execution result is 0, the system variable #L_CalcZero turns ON.

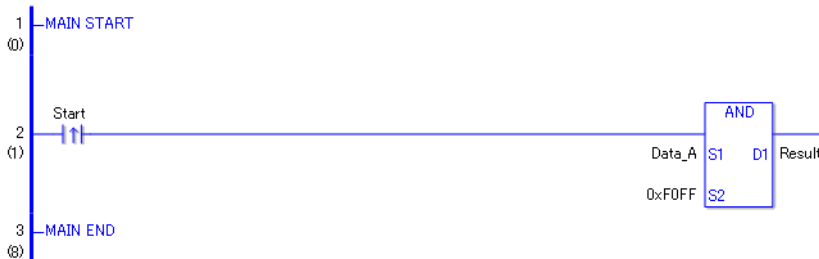
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

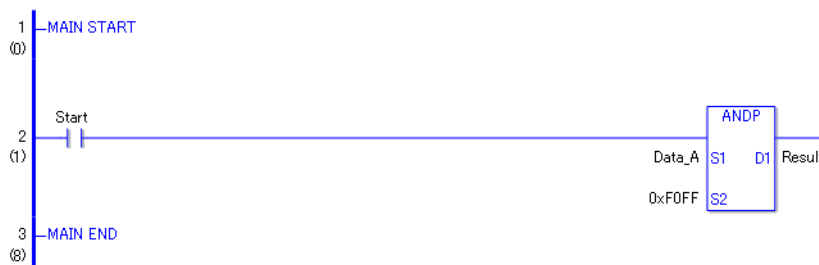
AND



(1) When the positive transition instruction turns on and passes power to the AND instruction, it does a logical AND between Data_A and F0FF and stores the result in D1 (integer variable). When using a normally open instruction, as long as the instruction is passing power, the AND instruction continually executes at each scan, performing the logical AND operation.

Program Example

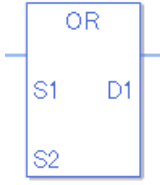

ANDP



- (1) When the normally open instruction turns ON and passes power to the ANDP instruction, it does a logical AND between Data_A and F0FF and stores the result in D1 (integer variable). Even when using a normally open instruction, the ANDP instruction executes only when it detects the upward transition. As a result, even if the instruction is always on, ANDP executes only at the first scan.

■ OR and ORP (Logical OR)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
OR (Logical OR - Level Sensitive)		Operation	4 to 13
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ORP (Logical OR - positive transition)		Operation	4 to 13

◆ Operand Settings

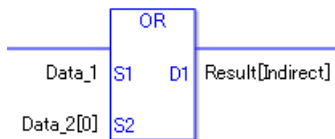
The following describes the specifiable content of Operands (S1, S2, and D1) in the OR and ORP instructions.

The actual number of steps in the OR and ORP instructions depends on the specification method of the operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in OR and ORP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



{Data_1 = 1 step} + {Data_2 [0] = 2 steps} + {Result [Specify indirectly]= 3 steps} + {1 step} = 7 steps

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1 and S2) in the OR and ORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant] Specify integer array(entire array)	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	- 2147483648 to 2147483647	1	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the OR and ORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant] Specify integer array(entire array)	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	- 2147483648 to 2147483647	—	X

◆ **Explanation of the OR and ORP Instructions**

The OR and ORP instructions are logic OR instructions. When an OR instruction is executed, S1 and S2 are logically ORed and the result is stored in D1.

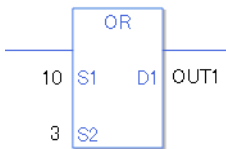
The OR and ORP instructions always pass power. When using the OR and ORP instructions, if the variables specified in operands S1, S2, and D1 are not the same type, an error will occur. Specify the same variable type in operands S1, S2, and D1.

Refer to the following for specifying a constant.

S1	Operator	S2	D1
OFF	OR	OFF	OFF
ON		OFF	ON
OFF		ON	ON
ON		ON	ON

When an OR instruction is executed, the D1 bit will be turned ON only when S1 and S2 are ON. Otherwise, the D1 bit is OFF.

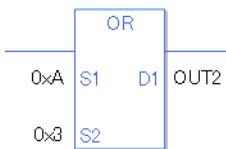
When operand D1 is an integer variable



Operand S1 Integer constant 10
 Operand S2 Integer constant 3
 Operand D1 Integer variable OUT1

When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

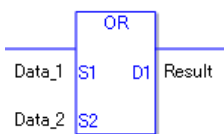
When 0x (zero and lower case x) is input, the following values will be interpreted as hexadecimal values.



Operand S1 Integer constant 0xA
 Operand S2 Integer constant 0x3
 Operand D1 Integer variable OUT2

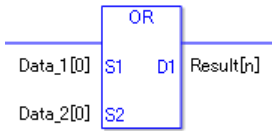
When Calculating Data in a Specified Array (Integer Variable Array)

Specifying the entire array



Data_1 Array size = 5
 Data_2 Array size = 5
 Result Array size = 5
 Logical operations are performed on all arrays.

Individually Specifying Array Variables



Data_1 [0] Array size = 5
 Data_2 [0] Array size = 5
 Result [n] Array size = 5
 Logical operations are performed on individual variables in the arrays.

◆ **Confirming Execution Results**

(1) When the execution result is 0, the system variable #L_CalcZero turns ON.

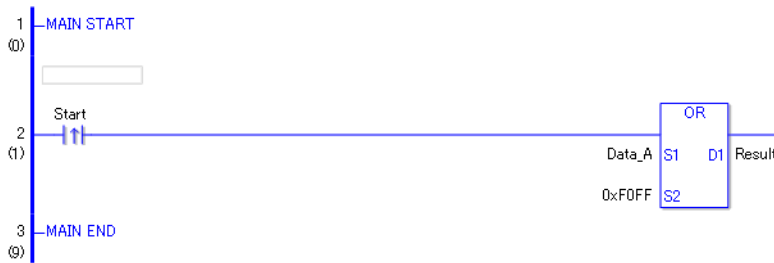
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

OR

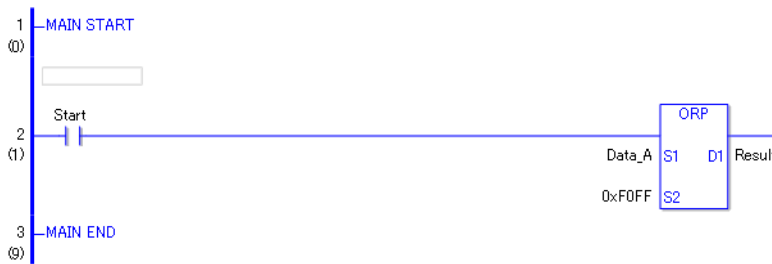


(1) When the positive transition instruction start turns ON, the OR instruction will be executed. When the OR instruction is executed, the result value obtained by ORing Data_A with F0FF is stored in D1.

When using a normally open instruction, as long as the instruction variable is ON, an OR instruction is always executed.

Program Example



ORP



- (1) When the normally open instruction turns ON, the ORP instruction will be executed. When the ORP instruction is executed, the result value obtained after Data_A is ORed with F0FF is stored in D1. Even when using a normally open instruction, the ORP instruction executes only when it detects the upward transition. Therefore, even when the variable of the NO instruction is always ON, the ORP instruction is executed only for one scan.

■ XOR and XORP (Logical XOR)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
XOR (Logical XOR - Level Sensitive)		Operation	4 to 13
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
XORP (Logical XOR - positive transition)		Operation	4 to 13

◆ Operand Settings

The following describes the specifiable content of Operands (S1, S2, and D1) in the XOR and XORP instructions.

The actual number of steps in the XOR and XORP instructions depends on the specification method of the operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in the XOR and XORP instructions

(For the number of steps in an operand, refer to operand settings on the next page.)



{Data_1 = 1 step} + {Data_2 [0] = 2 steps} + {Result [Specify indirectly]= 3 steps} + {1 step} = 7 steps

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1 and S2) in the XOR and XORP instructions

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/ W[constant] Specify integer array(entire array)	2	O
		Specify integer variable[variable] or Specify integer variable B/ W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	- 2147483648 to 2147483647	1	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the XOR and XORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant] Specify integer array(entire array)	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	- 2147483648 to 2147483647	—	X

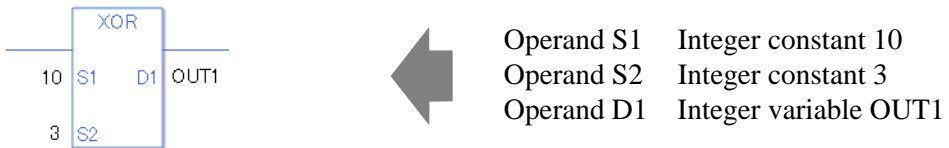
◆ **Explanation of the XOR and XORP Instructions**

XOR and XORP instructions are exclusive OR instructions. When an XOR instruction runs, it runs a logical XOR operation between S1 and S2 stores the result in D1. The XOR and XORP instructions always pass power. When using XOR and XORP, specify the same variable data type in operands S1, S2, and D1 to avoid errors. Refer to the following for specifying a constant.

S1	Operator	S2	D1
OFF	XOR	OFF	OFF
ON		OFF	ON
OFF		ON	ON
ON		ON	OFF

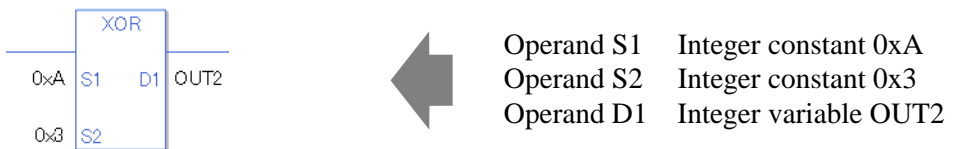
When an XOR instruction is executed, the D1 bit will be turned ON only when either S1 or S2 is ON. Otherwise, the D1 bit is OFF.

When operand D1 is an integer variable



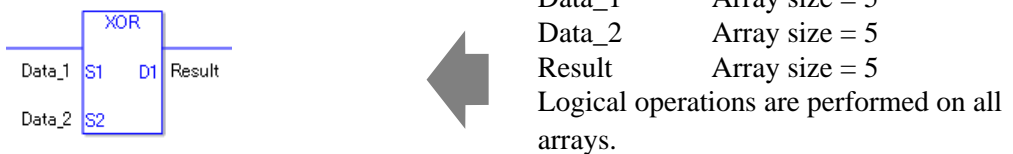
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values will be interpreted as hexadecimal values.

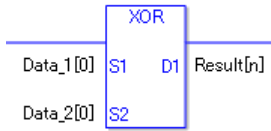


When Calculating Data in a Specified Array (Integer Variable Array)

Specifying the entire array



Individually Specifying Array Variables



Data_1 [0] Array size = 5
 Data_2 [0] Array size = 5
 Result [n] Array size = 5
 Logical operations are performed on individual variables in the arrays.

◆ **Confirming Execution Results**

(1) When the execution result is 0, the system variable #L_CalcZero turns ON.

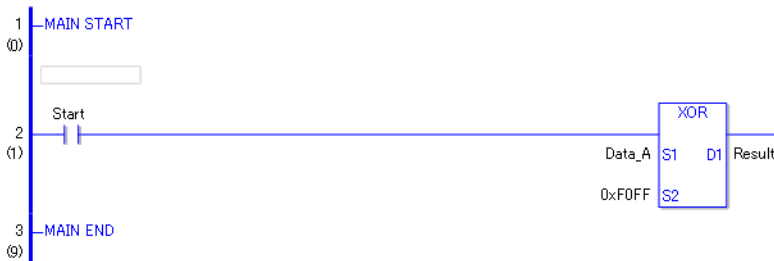
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

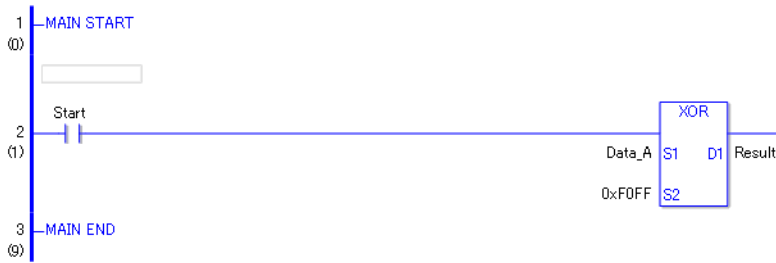
XOR



- (1) When the positive transition instruction turns ON, the XOR instruction will be executed. When the XOR instruction is executed, the result value obtained by XORing Data_A with F0FF is stored in D1.
- When using a normally open instruction, as long as the instruction variable is ON, the XOR instruction is always executed.

Program Example



XORP



- (1) When the normally open instruction turns ON, the XORP instruction will be executed. When the XORP instruction is executed, the result value obtained after Data_A is XORed with F0FF is stored in D1. Even when using a normally open instruction, the XORP instruction executes only when it detects the upward transition. Therefore, even when the variable of the NO instruction is always ON, the XORP instruction is executed only for one scan.

■ **NOT and NOTP (Logical NOT)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NOT (Logical NOT - Level Sensitive)		Operation	3 to 9
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NOTP (Logical NOT - positive transition)		Operation	3 to 9

◆ **Operand Settings**

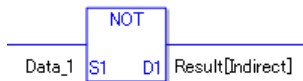
The following describes the specifiable content of operands S1 and D1 for the NOT and NOTP instructions.

The actual number of steps in the NOT and NOTP instructions depends on the specification method of the operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in the NOT and NOTP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{Result [Specify indirectly] = 3 \text{ steps}\} + \{1 \text{ step}\} = 5 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the NOT and NOTP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant] Specify integer array(entire array)	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	- 2147483648 to 2147483647	1	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the NOT and NOTP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant] Specify integer array(entire array)	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	- 2147483648 to 2147483647	—	X

◆ **Explanation of the NOT and NOTP Instructions**

The NOT and NOTP instructions are logical invert instructions. When a NOT instruction is run, S1 is logically inverted and the result is stored in D1. NOT and NOTP instructions always pass power. When using NOT and NOTP instructions, specify the same variable data type in operands S1 and D1 to avoid errors.

Refer to the following for specifying a constant.

S1	Operator	D1
OFF	NOT	ON
ON		OFF

When a NOT instruction is executed, if the S1 bit is OFF, the D1 bit turns ON. If the S1 bit is ON, the D1 bit turns OFF.

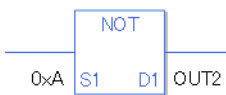
When operand D1 is an integer variable



Operand S1 Integer constant 10
 Operand D1 Integer variable OUT1

When operand D1 is an integer variable and you want to input a hexadecimal value in operand S1.

When 0x (zero and lower case x) is input, the following values will be interpreted as hexadecimal values.



Operand S1 Integer constant 0xA
 Operand D1 Integer variable OUT2

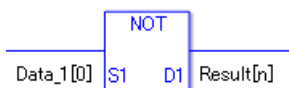
When Calculating Data in a Specified Array (Integer Variable Array)

Specifying the entire array



Data_1 Array size = 5
 Result Array size = 5
 Logical operations are performed on all arrays.

Individually Specifying Array Variables



Data_1 [0] Array size = 5
 Result [0] Array size = 5
 Logical operations are performed on individual variables in the arrays.

◆ **Confirming Execution Results**

(1) When the execution result is 0, the system variable #L_CalcZero turns ON.

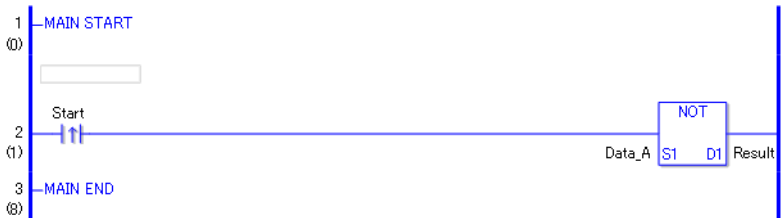
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

NOT



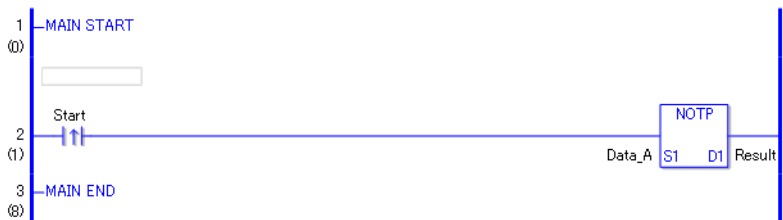
(1) When the positive transition instruction turns ON, the NOT instruction will be executed.

When the NOT instruction is executed, the result value obtained by logically inverting Data_A is stored in D1.

When using a normally open instruction, as long as the instruction variable is ON, the NOT instruction is always executed.

Program Example

NOTP



(1) When the normally open instruction turns ON, the NOTP instruction will be executed.



When the NOTP is executed, the result value obtained after Data_A is logically inverted is stored in D1.

Even when using a normally open instruction, the NOTP instruction executes only when it detects the upward transition. Therefore, even when the variable of the NO instruction is always ON, the NOTP instruction is executed only for one scan.

29.5.10 Operation (Transfer)

■ MOV and MOVP (Transfer)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
MOV (Transfer - Level Sensitive)		Transfer	3 to 9
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
MOVP (Transfer - positive transition)		Transfer	3 to 9

◆ Operand Settings

The following describes the specifiable content of Operands (S1 and D1) in the MOV and MOVP instructions.

The actual number of steps in the MOV and MOVP instructions depends on the specification method of the operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in the MOV and MOVP instructions
(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{Result \text{ [Specify indirectly]} = 3 \text{ steps}\} + \{1 \text{ step}\} = 5 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the MOV and MOVP instructions

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable].B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable [variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	- 2147483648 to 2147483647	1	O
	Float	±1.175494351e-38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the MOV and MOVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (output only)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable [variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	—	—	—	X

◆ **Explanation of the MOV and MOVP Instructions**

The MOV and MOVP instructions are transfer instructions. When the MOV instruction is executed, the value in S1 is stored in D1.

The MOV and MOVP instructions always pass power. When using the MOV and MOVP instructions, if the variables specified in operands S1 and D1 are not the same type, an error will occur. Specify the same variable type in operands S1 and D1.

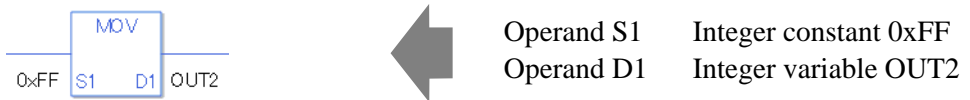
Refer to the following for specifying a constant.

When operand D1 is an integer variable



When operand D1 is an integer variable and you want to input a hexadecimal value in operand S1.

When 0x (zero and lower case x) is input, the following values will be interpreted as hexadecimal values.



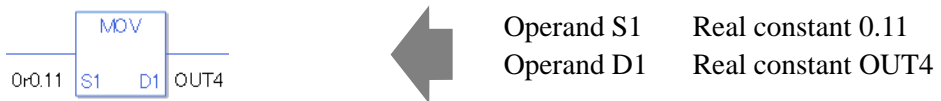
When operand D1 is a float variable

When 0f (zero and lower case "f") is input, the following values are interpreted as float values.



When operand D1 is a real variable

When 0r (zero and lower case "r") is input, the following values are interpreted as real values.

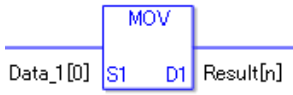


When transferring data in a specified array (integer variable array) Specify the array using data [0] or data [N] (N indicates an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the variables are the same type.



Data_1 Array size = 5
 Result Array size = 5
 The operand specification in the left diagram results in an error.



Data_1 [0] Array size = 5
 Result [n] Array size = 5
 The figure to the left shows that the operand is specified successfully (no error).

◆ **Confirming Execution Results**

- (1) If a numeric value cannot be indicated by operand S1 (when the execution result exceeds the range), the instruction will not be executed.
 #L_Error turns ON and an error code (6706) is set in #L_CalcErrCode.
 The output result D1 keeps its previous value with which the instruction was executed successfully.

(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

MOV

Stores the constant in the integer variable.



- (1) When the positive transition instruction turns ON, the MOV instruction will be executed.
 When the MOV instruction is executed, the constant 10 is stored in D1.
 When using a normally open instruction, as long as the instruction variable is ON, the MOV instruction is always executed.

Program Example

MOVP

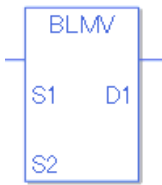
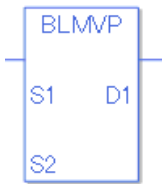


(1) When the normally open instruction turns ON, the MOVP instruction is executed. When the MOVP instruction is executed, the constant 10 is stored in D1.

Even when using a normally open instruction, the MOVP instruction executes only when it detects the upward transition. Therefore, even when the variable of the NO instruction is always ON, the MOVP instruction is executed only for one scan.

■ BLMV and BLMVP (Block Transfer)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BLMV (Block Transfer - Level Sensitive)		Transfer	6 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BLMVP (Block Transfer - positive transition)		Transfer	6 to 10

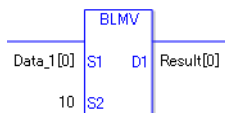
◆ Operand Settings

The following describes the specifiable content of Operands (S1, S2, and D1) in the BLMV and BLMVP instructions.

The actual number of steps in the BLMV and BLMVP instructions depends on the specification method of the operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in the BLMV and BLMVP instructions
 (For the number of steps in an operand, refer to the operand settings on the next page.)



{Data_1 [0] = 2 steps} + {10 = 1 step} + {Result [0] = 2 steps} + {1 step} = 6 steps

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1 and D1) in the BLMV and BLMVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	2	O
		Specify bit array ([variable])	3	O
	Integer (not including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	2	O
		Specify float variable [variable]	3	O
	Real	—	—	X
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
Date	.YR/ .MO/ .DAY only	—	X	
Time	.HR/ .MIN/ .SEC only	—	X	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	1	O
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	Integer	- 2147483648 to 2147483647	—	X
	Float	±1.175494351e-38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (S2) in for the BLMV and BLMVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X

Continued

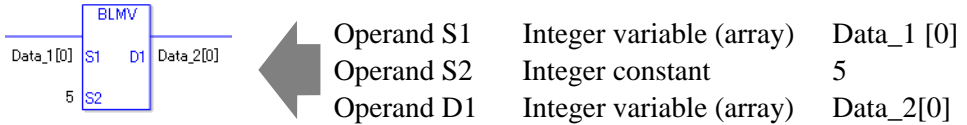
Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	—	1 to 4096	1	O

◆ **Explanation of the BLMV and BLMVP Instructions**

The BLMV and BLMVP instructions are block transfer instructions. When the BLMV instruction is executed, the number of data elements indicated in S2 are copied from S1 to D1. The BLMV and BLMVP instructions always pass power. When using the BLMV and BLMVP instructions, if the variables specified in operands S1 and D1 are not the same type, an error will occur. Specify the same variable type in the operands S1 and D1.

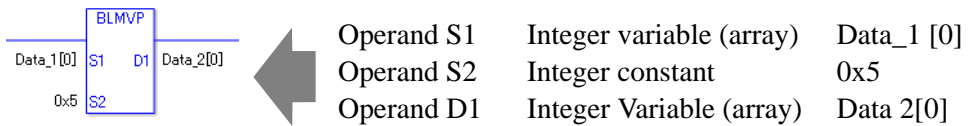
Refer to the following for specifying a constant.

When operand D1 is an integer variable



When operand D1 is an integer variable and you want to input a hexadecimal value in operand S2

When 0x (zero and lower case "x") is input, the following values are interpreted as hexadecimal values.



◆ **Confirming Execution Results**

(1) When the range of the array is exceeded (when the execution result exceeds the range), an instruction will not be executed. #L_Error turns ON and an error code is set in #L_CalcErrCode. The output result D1 keeps the last result of a successful operation.

(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

BLMV

Copies 1 through 5 from Data_1 to Data_2.



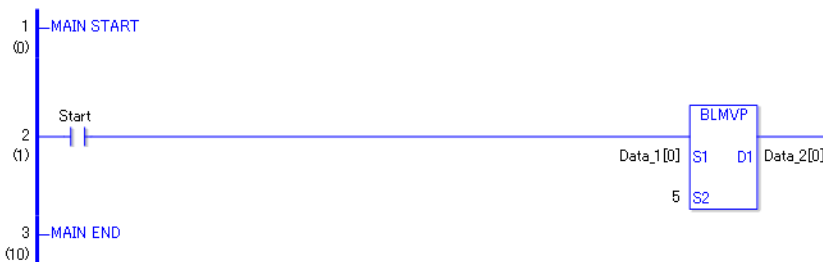
(1) When the positive transition instruction turns ON, the BLMV instruction will be executed. When the BLMV instruction is executed, numbers 0 through 4 in data 1, stored in D1, are copied to numbers 0 through 4 in data 2.

When the start is a normally open instruction, as long as the start is ON, the BLMV instruction is always executed.

Array Variable Name	Data 1	5 Executed Instructions	Data 2	
Element	Data_1 [0]	→	Data_2 [0]	
	Data_1 [1]	→	Data_2 [1]	
	Data_1 [2]	→	Data_2 [2]	
	Data_1 [3]	→	Data_2 [3]	
	Data_1 [4]	→	Data_2 [4]	
	Data_1 [5]	→	Data_2 [5]	
	Data_1 [6]			Data_2 [6]
	Data_1 [7]			Data_2 [7]
	Data_1 [8]			Data_2 [8]
	Data_1 [9]			Data_2 [9]
	Data_1 [10]			Data_2 [10]

Program Example



BLMVP



- (1) When the normally open instruction turns ON, the BLMVP instruction will be executed. When the BLMVP instruction is executed, numbers 0 through 4 in data 1, stored in D1, are copied to numbers 0 through 4 in data 2. Even when using a normally open instruction, only the start ON up edge is detected, and the BLMVP instruction is executed. Therefore, even when the variable of the NO instruction is always ON, the BLMVP instruction is executed only for one scan.

■ **FLMV and FLMVP (Multipoint Transfer)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
FLMV (Multipoint Transfer - Level Sensitive)		Transfer	4 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
FLMVP (Multipoint Transfer - positive transition)		Transfer	4 to 10

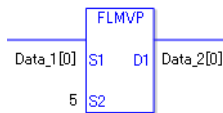
◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2, and D1) in the FLMV and FLMVP instructions.

The actual number of steps in the FLMV and FLMVP instructions depends on the specification method of the operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in the FLMV and FLMVP instructions
(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{ \text{Data}_1 [0] = 2 \text{ steps} \} + \{ 5 = 1 \text{ step} \} + \{ \text{Data}_2 [0] = 2 \text{ steps} \} + \{ 1 \text{ step} \} = 6 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the FLMV and FLMVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable [variable]	3	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (S2) in the FLMV and FLMVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	—	X
		Specify integer variable [Variable]	—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
J_	.HR/ .MIN/ .SEC only	2	O	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	—	1 to 4096 (Maximum number of arrays)	1	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the FLMV and FLMVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (not including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	2	O
		Specify float variable [variable]	3	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	—	X
	Counter	.PV/.CV only	—	X
	Date	.YR/.MO/.DAY only	—	X
	Time	.HR/.MIN/.SEC only	—	X
	PID	.KP/.TR/.TD/.PA/.BA/.ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/.CV only	—	X
	N_	.YR/.MO/.DAY only	—	X
	J_	.HR/.MIN/.SEC only	—	X
	U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	X
Constant	Integer	- 2147483648 to 2147483647	—	X
	Float	±1.175494351e-38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	X

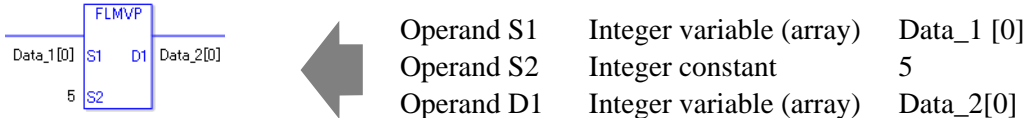
◆ **Explanation of the FLMV and FLMVP Instructions**

FLMV and FLMVP instructions are multiple point transfer instructions. When a FLMV instruction is run, the value in S1 is copied to S2 number of addresses, beginning with the address in D1.

FLMV and FLMVP instructions always pass power. When using FLMV and FLMVP instructions, specify the same data type in operands S1 and D1 to avoid errors.

Refer to the following for specifying a constant.

When operand D1 is an integer variable



When operand D1 is an integer variable and you want to input hexadecimal values in operand S2.

When 0x (zero and lower case x) is input, the following values will be interpreted as hexadecimal values.



◆ **Confirming Execution Results**

(1) When the range of the array is exceeded (when the execution result exceeds the range), an instruction will not be executed. #L_Error turns ON and an error code is set in #L_CalcErrCode. The output result D1 keeps its previous value with which the instruction was executed successfully.

(Notes)

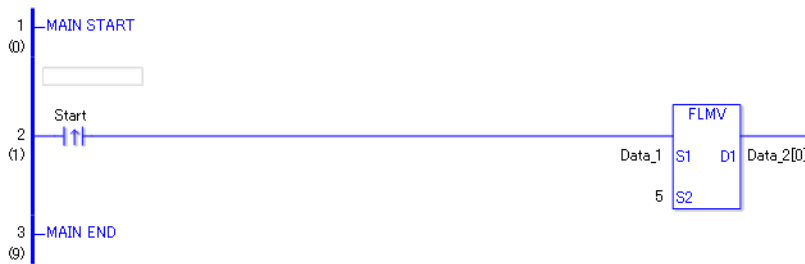
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

FLMV

Copies the data in data 1 to elements 0 through 4 in data 2.



(1) When the positive transition instruction turns ON, the FLMV instruction will be executed.

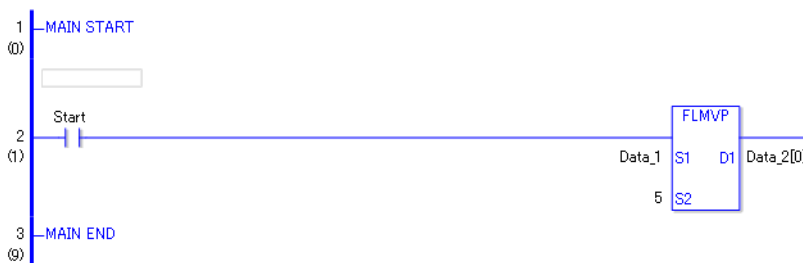
When the FLMV instruction is executed, data 1, stored in D1, is copied to elements 0 through 4 in data 2.

When the start is a normally open instruction, as long as the start is ON, the FLMV instruction is always executed.

Array Variable Name	Data 1	5 Executed Instructions	Data 2
Element	Data 1	→	Data_2 [0]
		→	Data_2 [1]
		→	Data_2 [2]
		→	Data_2 [3]
		→	Data_2 [4]
			Data_2 [5]
			Data_2 [6]
			Data_2 [7]
			Data_2 [8]
			Data_2 [9]
			Data_2 [10]

Program Example

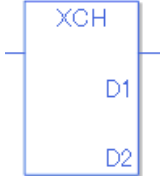
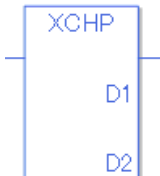
FLMVP



- (1) When the normally open instruction turns ON, the FLMVP instruction will be executed. When the FLMVP instruction is executed, the data in data 1, stored in D1, is copied to numbers 0 through 4 in data 2. Even when using a normally open instruction, the FLMVP instruction executes only when it detects the upward transition. Therefore, even when the variable of the NO instruction is always ON, the FLMVP instruction is executed only for one scan.

■ XCH and XCHP (Exchange)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
XCH (Exchange - Level Sensitive)		Transfer	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
XCHP (Exchange - positive transition)		Transfer	3 to 7

◆ Operand Settings

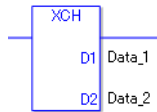
The following describes the specifiable content of Operands (D1 and D2) in the XCH and XCHP instructions.

The actual number of steps in the XCH and XCHP instructions depends on the specification method of the operand. The following describes how to calculate the number of steps.

Number of steps in operand D1 + Number of steps in operand D2 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in the XCH and XCHP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{\text{Data}_1 = 1 \text{ step}\} + \{\text{Data}_2 = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (D1 and D2) in the XCH and XCHP instructions

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (Output included)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the XCH and XCHP Instructions**

XCH and XCHP instructions are exchange instructions. When a XCH instruction is run, the data in operands D1 and D2 are switched.

The XCH and XCHP instructions always pass power. When using XCH and XCHP instructions, specify the same data type in operands D1 and D2 to avoid errors.

◆ **Confirming Execution Results**

(1) When the result exceeds the array size, the instruction will not run, the system variable #L_Error turns on, and the error code is written to #L_CalcErrCode. D1 and D2 revert to values from the previous successfully run instruction.

(Notes)

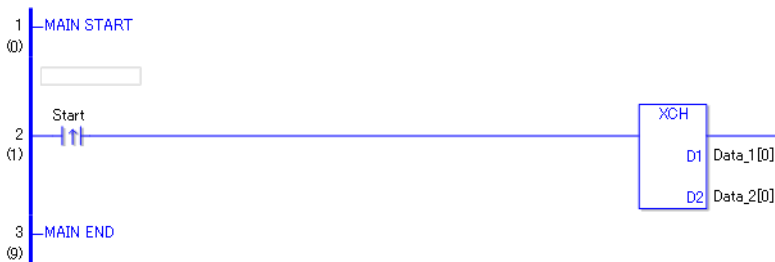
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

XCH

Exchanges the contents of data 1 and data 2.



(1) When the positive transition instruction turns ON, the XCH instruction will be executed.

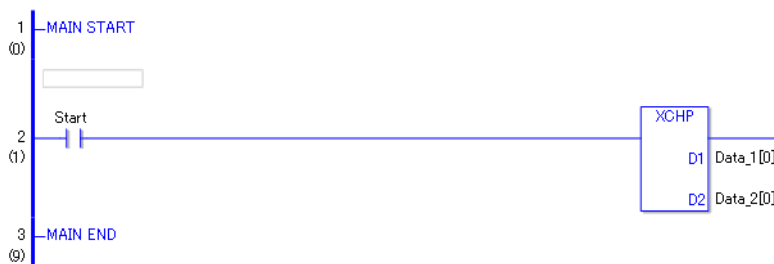
When the XCH instruction is executed, the summary of the Data_1[0] in D1 and the Data_2[0] in D2 will be switched.

When using a normally open instruction, as long as the instruction variable is ON, the XCH instruction is always executed.

Array Variable Name	Data 1	Instruction Execution	Data 2
Element	Data_1[0]	↔	Data_2 [0]
	Data_1 [1]		Data_2 [1]
	Data_1 [2]		Data_2 [2]
	Data_1 [3]		Data_2 [3]
	Data_1 [4]		Data_2 [4]

Program Example

XCHP

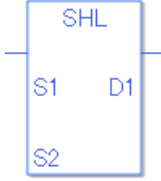
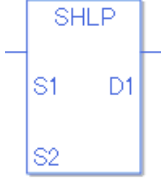


- (1) When the positive transition instruction turns ON, the XCHP instruction will be executed. When the XCHP instruction is executed, the summary of the Data_1[0] in D1 and the Data_2[0] in D2 will be switched. Even when the operation is a normally open instruction, the XCHP instruction will execute only when an upward transition is detected. Therefore, even when the variable of the normally open instruction is always ON, the XCHP instruction executes for only one scan.

29.5.11 Calculation Instruction (Shift Instruction)

■ SHL and SHLP (Shift Left)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SHL (Shift Left - Level Sensitive)		Shift	4 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SHLP (Shift Left - positive transition)		Shift	4 to 10

◆ Operand Settings

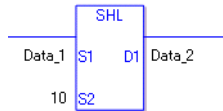
The following table lists the specifiable content of operands S1, S2, and D1 for the SHL and SHLP instructions.

The actual number of steps in the SHL and SHLP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in SHL and SHLP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{Data_2 = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in SHL and SHLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or entire array	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (S2) in SHL and SHLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	0 to 131071	1	O
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in SHL and SHLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (Output included)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or entire array	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the SHL and SHLP Instructions**

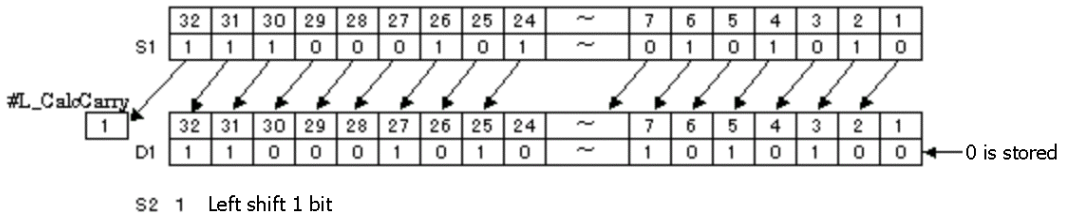
When an SHL or SHLP instruction is executed, the S1 bits are shifted to the left S2 number of bits. Every time 1 bit is shifted, the leftmost bit (the most significant bit) is lost. 0 is stored in the rightmost empty bit. The results are stored in D1.

The SHL and SHLP instructions always pass power. When using SHL and SHLP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

- S1: Shift address Specifies the address to shift.
- S2: Number of bits to shift Specifies the number of bits to shift.
- D1: Store address Specifies the address to store the shift result.

For example, When 1 bit is shifted left



When operand D1 is an integer variable



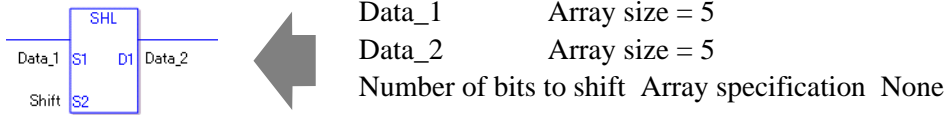
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values will be interpreted as hexadecimal values.

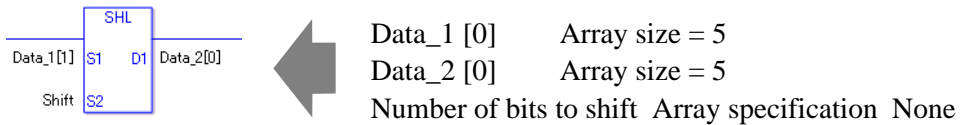


Use the same format when shifting data in a specified array (integer variable array) and when specifying an array element.

An error will occur if the formats are different.



If the S1 and D1 arrays are the same size, S1 is treated like a single giant integer. Bits are shifted one element to the next element. The topmost bits of each element are not lost. However, the topmost bit in the last element is lost. Specify S2 as 0 or higher (32 x Array Size - 1).



If both S1 and D1 are not in an array, this instruction shifts the 32 bits in S1. Specify a value between 0 and 31 for S2.

◆ **System Variables Indicating Execution Results**

- When the execution result is 0, #L_CalcZero turns on.
- If an overflow occurs as a result of the shift operation, the last overflowed bit is stored in #L_CalcCarry.
- When the execution results in an error, the error information is stored in #L_Status.
- When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

- When checking the result using system variables, make sure the check takes place after the instruction has been executed.
- When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

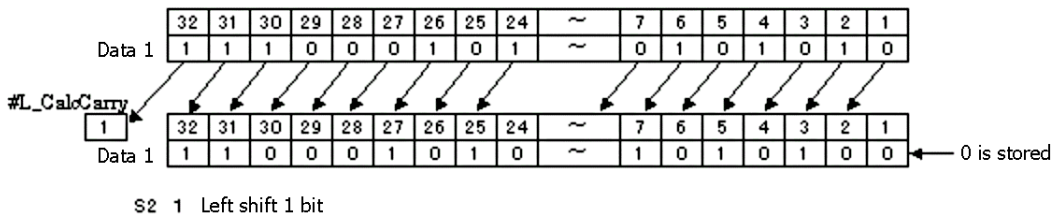
Program Example

SHL

Determines whether the most significant bit is ON or OFF.

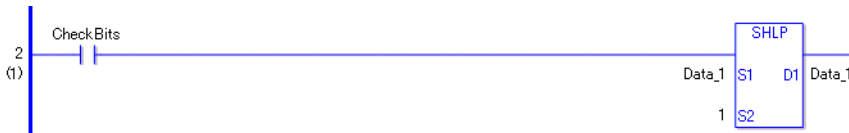


- (1) When the positive transition instruction turns ON, an SHL instruction will be executed. When the SHL instruction is executed, the result from shifting 1 bit to the left is stored in D1.
 - (2) When 1 bit is shifted to the left, you can check whether the most significant bit before data shifting is ON or OFF from the state of #L_CalcCarry.
- (Note) When using a normally open instruction, the SHL instruction is always executed as long as the normally open instruction is ON.



Program Example

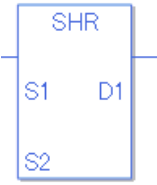
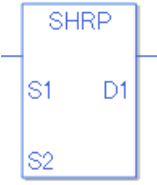
SHLP



SHLP and SHL instructions have different ways of detecting when to execute. In SHLP instructions, even when using a normally open instruction, only the upward transition is detected and the SHLP instruction is executed. Therefore, the SHLP instruction is executed only for one scan, even when the normally open instruction remains ON.

■ **SHR and SHRP (Shift Right)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SHR (Shift Right - Level Sensitive)		Shift	4 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SHRP (Shift Right - positive transition)		Shift	4 to 10

◆ **Operand Settings**

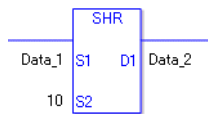
The following describes the specifiable content of operands S1, S2, and D1 for the SHR and SHRP instructions.

The number of steps in the SHR and SHRP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in SHR and SHRP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{Data_2 = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in SHR and SHRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or entire array	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (S2) in SHR and SHRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	0 to 131071	1	O
	Float	±1.175494351e-38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in SHR and SHRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (Output included)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or entire array	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

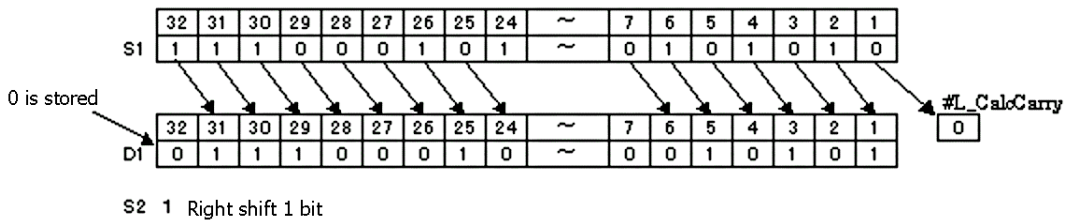
◆ **Explanation of SHR and SHRP Instructions**

When the SHR or SHRP instruction is executed, the S1 bits are shifted to the right S2 number of bits. Every time 1 bit is shifted, the rightmost bit (least significant bit) is lost. 0 is stored in the empty topmost bit positions. The result is stored in D1.

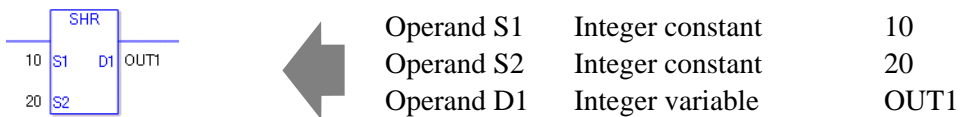
The SHR and SHRP instructions always pass power. When using SHR and SHRP instructions, an error will occur if the variables specified in operands S1 and D1 operands are not the same type. Specify the same variable type in the S1 and D1 operands. Refer to the following for specifying a constant.

- S1: Shift address Specifies the address to shift.
- S2: Number of bits to shift Specifies the number of bits to shift.
- D1: Store address Specifies the address to store the shift result.

For example, when 1 bit is shifted to the right

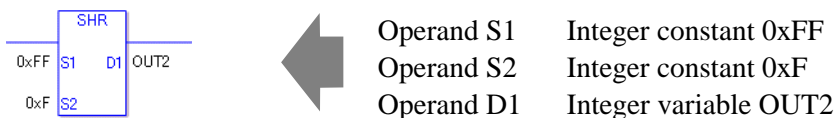


When operand D1 is an integer variable



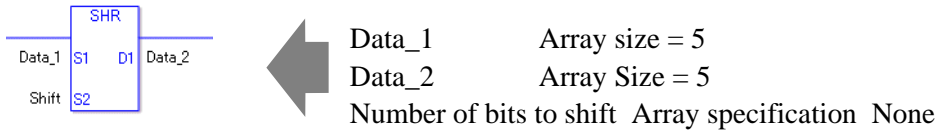
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values will be interpreted as hexadecimal values.



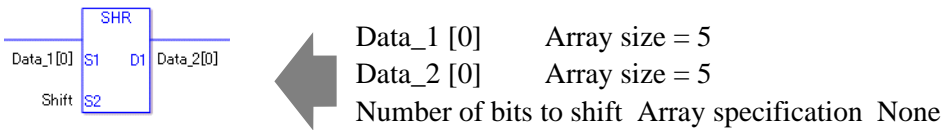
Use the same format when shifting data in a specified array (integer variable array) and when specifying an array element.

An error will occur if the formats are different.



If the S1 and D1 arrays are the same size, S1 is treated like a single giant integer. Bits are shifted one element to the next element.

The bottom bit of each element is not lost, except for the bottom bit in the last element. Specify S2 as 0 or higher, up to (32 x Array Size - 1).



If both S1 and D1 are not arrays, 32 bits are shifted. Specify S2 between 0 and 31.

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

If an overflow occurs as a result of the shift operation, the last overflowed bit is stored in #L_CalcCarry.

When the execution results in an error, the error information is stored in #L_Status.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

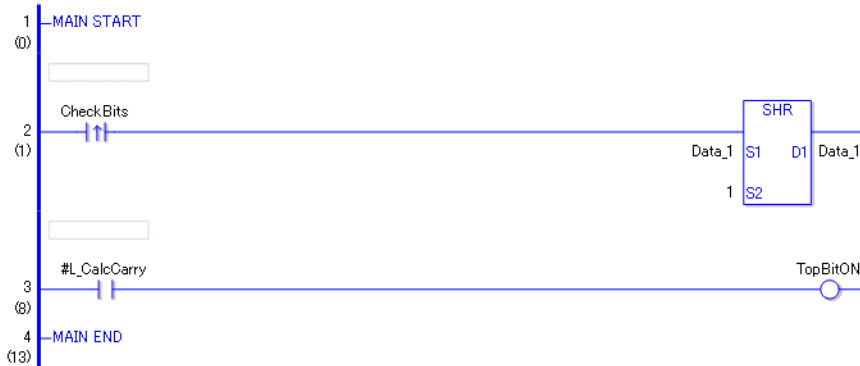
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

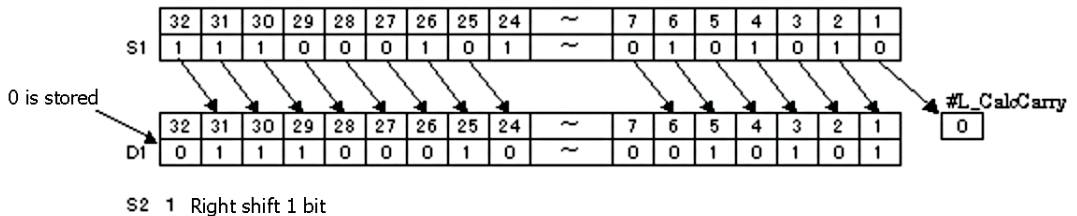
Program Example

SHR

Determines whether the least significant bit is ON or OFF.

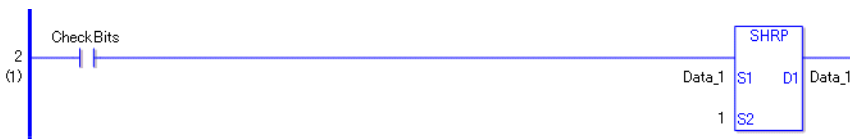


- (1) When the variable of the positive transition instruction turns ON, the SHR instruction is run. The SHR instruction shifts bits in Data_1 one bit to the right and stores the result in D1.
- (2) After the bit shift operation is complete, you can check the previous value of the least significant bit in Data_1 by using the #L_CalcCarry system variable.
(Supplementary) When using a normally open instruction, the SHR instruction is always executed as long as the bit remains ON.



Program Example

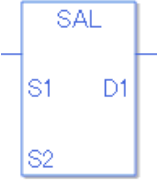
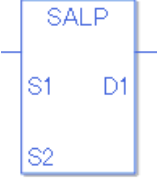
SHRP



When to run the instruction is different between SHRP and SHR instructions. In the SHRP instruction, even when using a normally open instruction, only the upward transition of the bit is detected, and the SHRP instruction is executed. Even if the bit of the normally open instruction remains ON, the SHRP instruction is executed only for one scan.

■ **SAL and SALP (Arithmetic Shift Left)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SAL (Arithmetic Shift Left - Level Sensitive)		Shift	4 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SALP (Arithmetic Shift Left - positive transition)		Shift	4 to 10

◆ **Operand Settings**

The following table lists the specifiable content of operands S1, S2, and D1 for the SAL and SALP instructions.

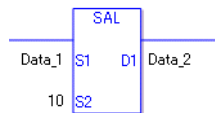
The number of steps in the SAL and SALP instructions depends on the specified operand.

The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in SAL and SALP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{Data_2 = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in SAL and SALP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/.CV only	2	O
	Date	.YR/.MO/.DAY only	2	O
	Time	.HR/.MIN/.SEC only	2	O
	PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/.CV only	2	O
	N_	.YR/.MO/.DAY only	2	O
	J_	.HR/.MIN/.SEC only	2	O
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	O	
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (S2) in SAL and SALP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	0 to 31	1	O
	Float	±1.175494351e-38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in SAL and SALP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (Output included)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	- 2147483648 to 2147483647	—	X
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error information is stored in #L_Status.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

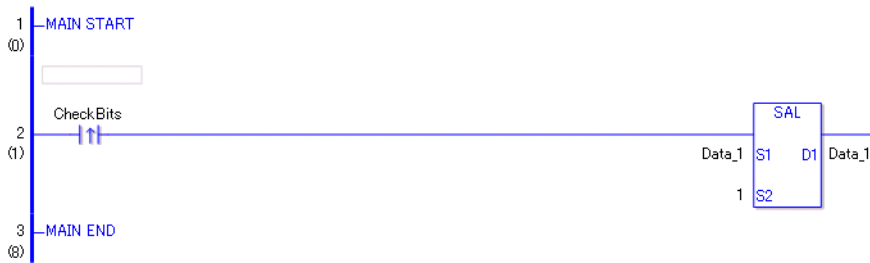
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

SAL

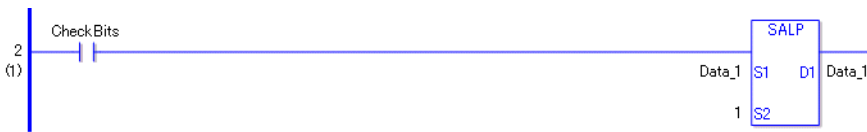


(1) When the positive transition instruction turns ON, the SAL instruction is executed. When the SAL instruction is executed, the result of the bit shift is stored in D1. The most significant bit is not shifted, and zero is stored in the least significant bit.

(Supplementary) When using a normally open instruction, the SAL instruction is always executed as long as the normally open bit is ON.

Program Example



SALP



The SALP and SAL instructions have different ways of detecting when to execute. In the SALP instruction, Even when using a normally open instruction, the SLAP instruction executes only when it detects the upward transition.As a result, if the bit remains ON, the SALP instruction is executed only for one scan.

■ **SAR and SARP (Arithmetic Shift Right)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SAR (Arithmetic Shift Right - Level Sensitive)		Shift	4 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SARP (Arithmetic Shift Right - positive transition)		Shift	4 to 10

◆ **Operand Settings**

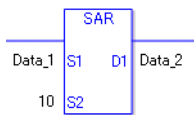
The following table lists the specifiable content of operands S1, S2, and D1 for the SAR and SARP instructions.

The actual number of steps in the SAR and SARP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in SAR and SARP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{Data_2 = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in SAR and SARP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (S2) in SAR and SARP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	0 to 31	1	O
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in SAR and SARP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (Output included)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the SAR and SARP Instructions**

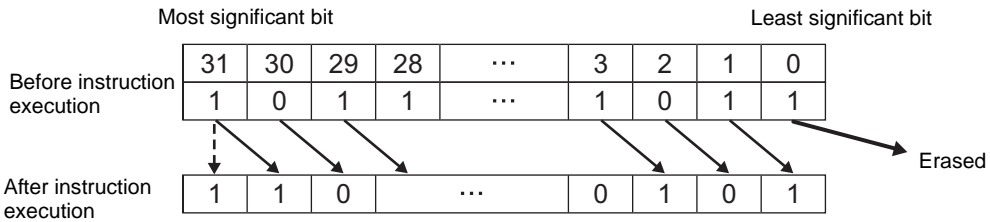
When the SAR or SARP instruction is executed, the S1 bits are shifted to the right of the S2 number of bits. For each bit shift, the bottom-most bit (the least significant bit) is lost, and the most significant bit is stored in the topmost empty bit. The result is stored in D1. The SAR and SARP instructions always pass power. When using the SAR and SARP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type.

Specify the same variable type in the S1 and D1 operands.

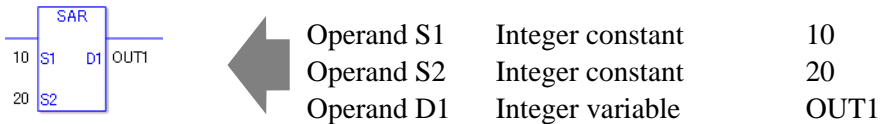
Refer to the following for specifying a constant.

- S1: Shift address Specifies the address to shift.
- S2: Number of bits to shift Specifies the number of bits to shift.
- D1: Store address Specifies the address to store the shift result.

For example, When 1 bit is shifted to the right

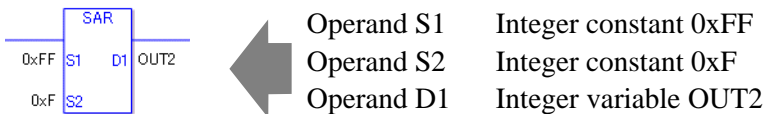


When operand D1 is an integer variable

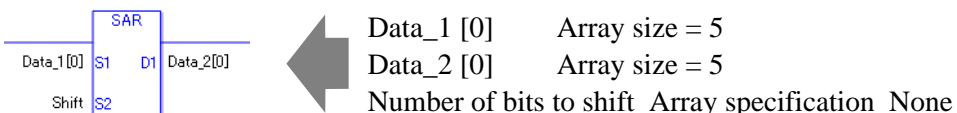


When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values will be interpreted as hexadecimal values.



When specifying an array variable, specify an array element.



31 array element bits are shifted. For S2, specify a value between 0 and 31.

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error information is stored in #L_Status.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

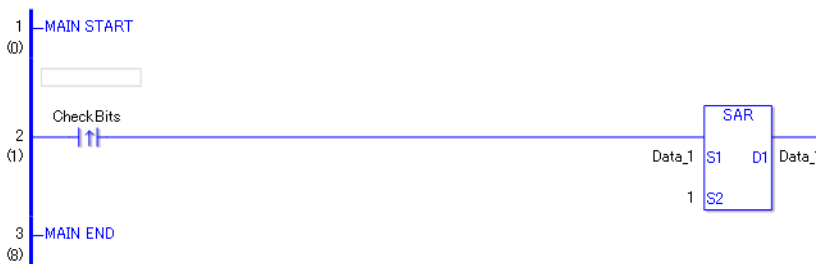
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

SAR



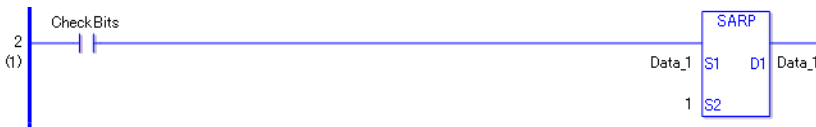
(1) When the positive transition instruction turns ON, the SAR instruction will be executed.

When the SAR instruction is executed, the 1 bit to the right is stored in D1. The most significant bit is not shifted but is also copied to D1. For every bit that shifts, the most significant bit is copied to the topmost empty bit.

(Supplementary) When using a normally open instruction, the SAR instruction is always executed as long as the bit is ON.

Program Example

SALP

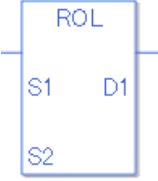
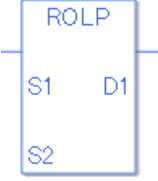


The SARP and SAR instructions have different ways of detecting when to execute. In the SARP instruction, even when using a normally open instruction, the SARP instruction is executed only when a positive transition is detected. As a result, even if the bit remains ON, the SARP instruction is executed only for one scan.

29.5.12 Operation (Rotation Instruction)

■ ROL and ROLP (Rotate Left)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ROL (Rotate Left - Level Sensitive)		Rotate	4 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ROLP (Rotate Left - positive transition)		Rotate	4 to 10

◆ Operand Settings

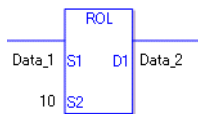
The following table lists the specifiable content of operands S1, S2, and D1 for the ROL and ROLP instructions.

The actual number of steps in the ROL and ROLP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in ROL and ROLP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{Data_2 = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in ROL and ROLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or entire array	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
		—	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/.CV only	2	O
	Date	.YR/.MO/.DAY only	2	O
	Time	.HR/.MIN/.SEC only	2	O
	PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/.CV only	2	O
	N_	.YR/.MO/.DAY only	2	O
	J_	.HR/.MIN/.SEC only	2	O
	U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (S2) in ROL and ROLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	0 to 131071	1	O
	Float	±1.175494351e-38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in ROL and ROLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (Output included)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or entire array	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X

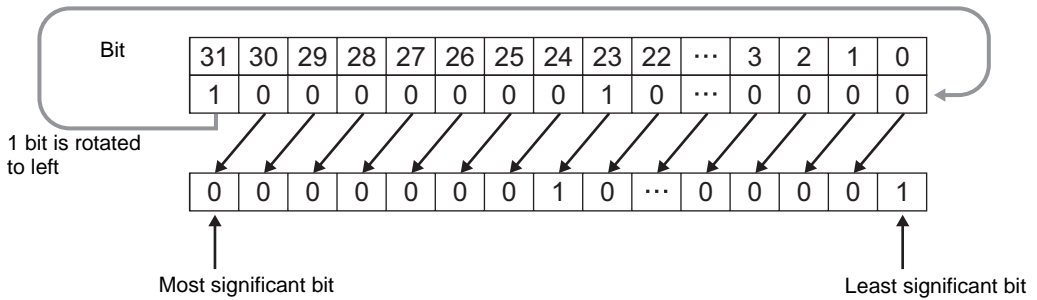
◆ **Explanation of the ROL and ROLP Instructions**

When the ROL or ROLP instruction is executed, the S1 bits are rotated to the left S2 number of bits. Every time 1 bit is rotated, the topmost bit (the most significant bit) is rotated to the bottom-most bit (least significant bit). The result is stored in D1. The ROL and ROLP instructions always pass power. When using the ROL and ROLP instructions, if the variables specified in operands S1 and D1 are not the same type, an error will occur. Specify the same variable type in the S1 and D1 operands.

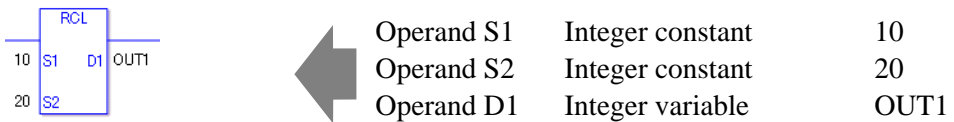
Refer to the following for specifying a constant.

- S1: Rotation address Specifies an address to rotate bits.
- S2: Number of bits to rotate Specifies the number of bits to rotate.
- D1: Storage device Specifies an address for storing the results after rotating bits.

For example, When 1 bit is rotated to left

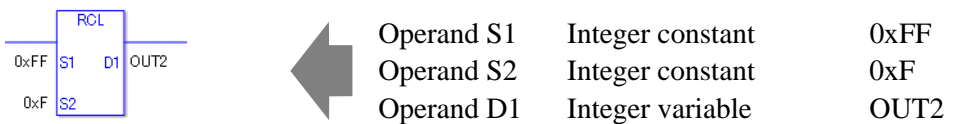


When operand D1 is an integer variable



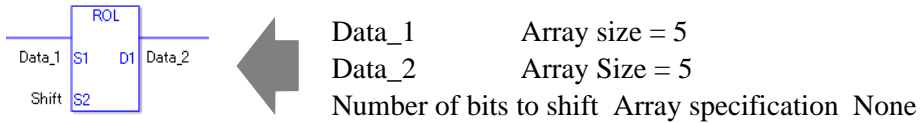
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal values.



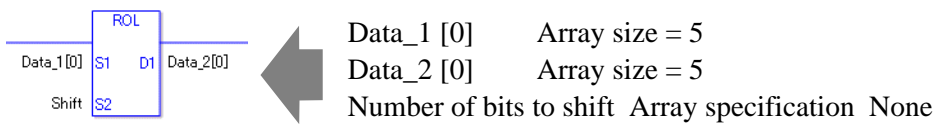
Use the same format when rotating data in a specified array (integer variable array) and when specifying an array element.

An error will occur if the formats are different.



If the arrays of S1 and D1 are the same size, S1 is treated as if it's a giant integer. Bits are rotated from one element to the next.

The entire array is rotated, not bits within each element. Specify S2 as 0 or higher, up to (32 x Array Size - 1).



If both S1 and D1 are not arrays, 32 bits are rotated. For S2, specify a value between 0 and 31.

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

If an overflow results from the rotation instruction, the overflowed bit is stored in #L_CalcCarry.

When the execution results in an error, the error information is stored in #L_Status.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

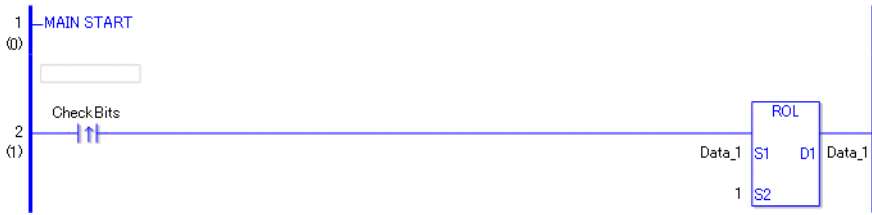
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

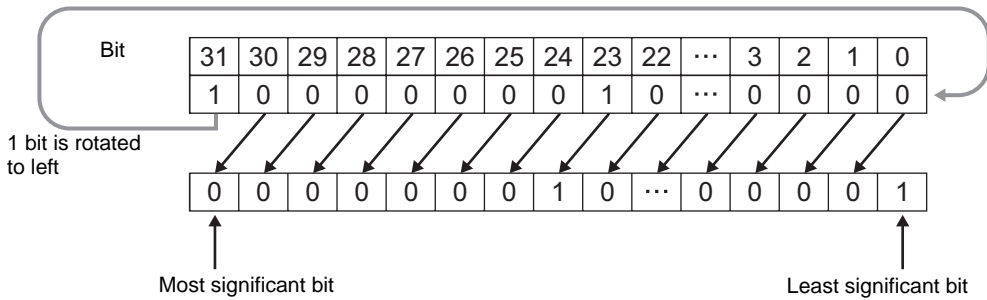
When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

ROL

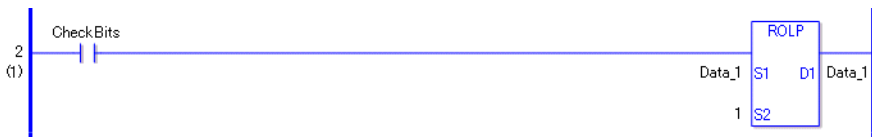


(1) When a positive transition instruction turns ON, the ROL instruction is executed. When the ROL instruction is executed, the result of rotating 1 bit is stored in D1.
 (Supplementary) When using a normally open instruction, the ROL instruction is always executed as long as the bit is ON.



Program Example

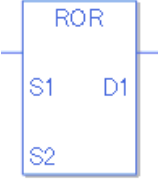
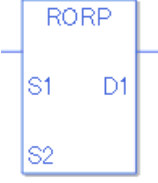
ROLP



The ROLP and ROL instructions have different ways of detecting when to execute. In the ROLP instruction, even when using a normally open instruction, only the upward transition of the bit is detected, and the ROLP instruction is executed. As a result, even if the bit remains ON, the ROLP instruction is executed only for one scan.

■ ROR and RORP (Rotate Right)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ROR (Rotate Right - Level Sensitive)	 A rectangular ladder symbol with 'ROR' at the top. Below it are two columns: 'S1' on the left and 'D1' on the right. At the bottom is 'S2'. Horizontal lines extend from the left and right sides.	Rotate	4 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
RORP (Rotate Right - positive transition)	 A rectangular ladder symbol with 'RORP' at the top. Below it are two columns: 'S1' on the left and 'D1' on the right. At the bottom is 'S2'. Horizontal lines extend from the left and right sides.	Rotate	4 to 10

◆ Operand Settings

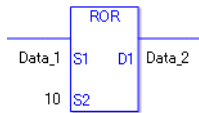
The following table lists the specifiable content of operands S1, S2, and D1 for ROR and RORP instructions..

The number of steps in the ROR and RORP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in the ROR and RORP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{Data_2 = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in ROR and RORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or entire array	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (S2) in ROR and RORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	0 to 131071	1	O
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in ROR and RORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (Output included)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or entire array	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

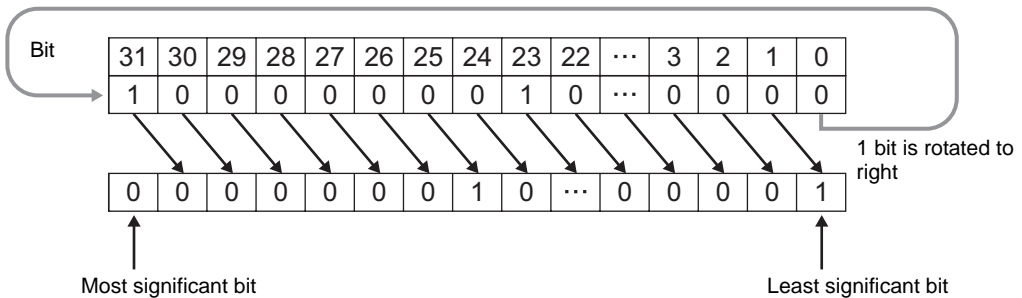
◆ **Explanation of the ROR and RORP Instructions**

When the ROR or RORP instruction is executed, the S1 bits are rotated to the right S2 number of bits. Every time 1 bit is rotated, the information of the bottom-most bit (the least significant bit) is stored in the topmost empty bit.

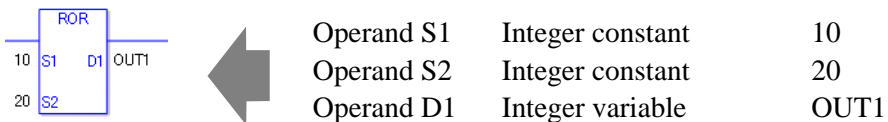
The result is stored in D1. The ROR and RORP instructions always pass power. When using the ROR and RORP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in the S1 and D1 operands. Refer to the following for specifying a constant.

- S1: Rotation address Specifies an address to rotate bits.
- S2: Number of bits to rotate Specifies the number of bits to rotate.
- D1: Storage device Specifies an address for storing the results after rotating bits.

For example, When 1 bit is rotated to right

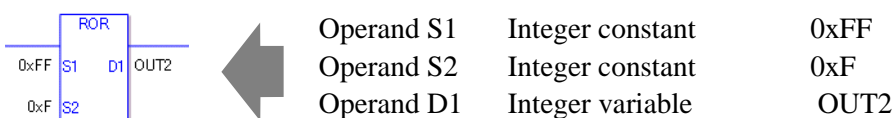


When operand D1 is an integer variable



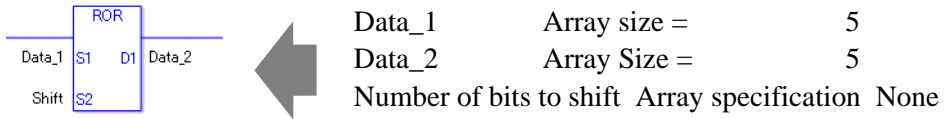
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal values.



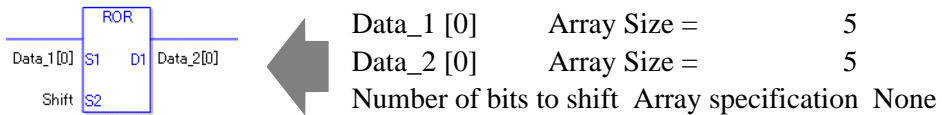
Use the same format when rotating data in a specified array (integer variable array) and when specifying an array element.

An error will occur if the formats are different.



If the arrays of S1 and D1 are the same size, S1 is treated as if it's a giant integer. Bits are rotated from one element to the next.

The entire array is rotated, not just bits in each element. For S2, specify a value from 0 to (32 x Array Size - 1).



If both S1 and D1 are not arrays, 32 bits are rotated. For S2, specify a value between 0 and 31.

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

If an overflow results from the rotation instruction, the overflowed bit is stored in #L_CalcCarry.

When the execution results in an error, the error information is stored in #L_Status.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

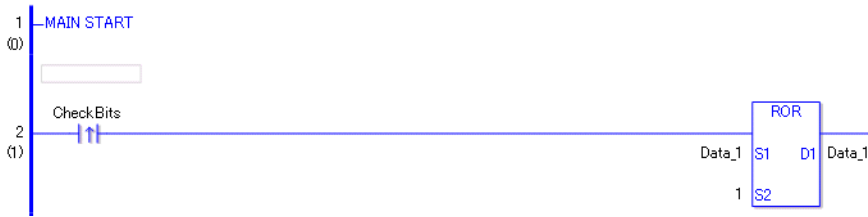
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

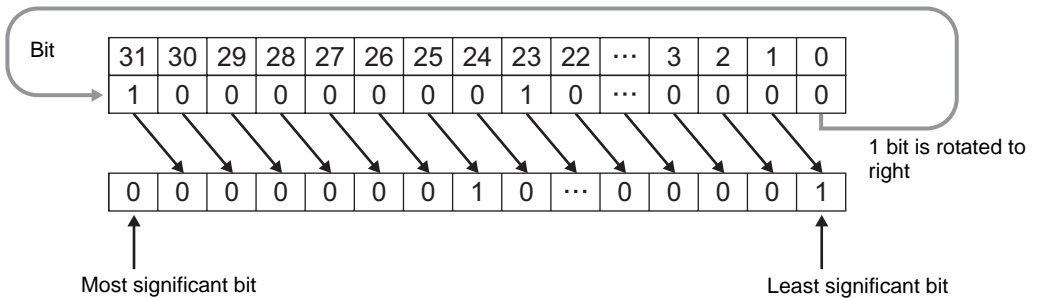
When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

ROR

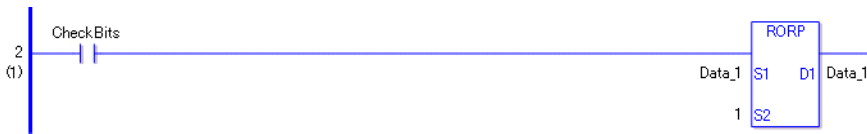


(1) When the positive transition instruction turns ON, the ROR instruction is executed. When the ROR instruction is executed, the result of rotating 1 bit to the right is stored in D1. (Supplementary) When using a normally open instruction, the ROR instruction is always executed as long as the bit is ON.



Program Example

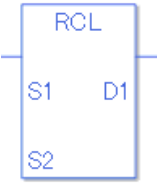
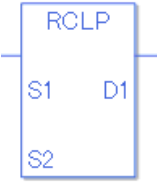
RORP



The RORP and ROR instructions have different ways of detecting when to execute. For RORP, even when using a normally open instruction, only the upward transition is detected, and the RORP instruction is executed. Therefore, the RORP instruction is executed only for one scan, even when the bit confirmation continues to turn ON.

■ RCL and RCLP (Rotate Left with Carry Over)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
RCL (Rotate Left with Carry-over - Level Sensitive)		Rotate	4 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
RCLP (Rotate Left with Carry-over - positive transition)		Rotate	4 to 10

◆ Operand Settings

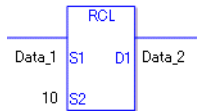
The following table lists the specifiable content of operands S1, S2, and D1 for the RCL and RCLP instructions.

The actual number of steps in the RCL and RCLP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in the RCL and RCLP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{Data_2 = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in RCL and RCLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (S2) in RCL and RCLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	0 to 32	1	O
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in RCL and RCLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (Output included)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

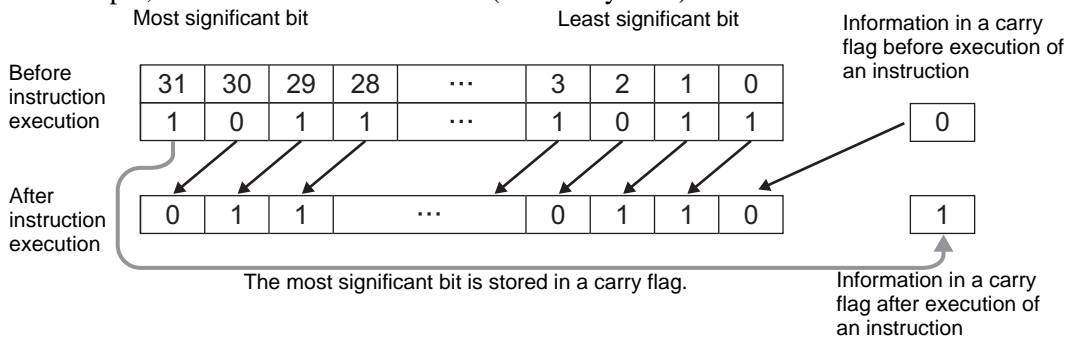
◆ **Explanation of the RCL and RCLP Instructions**

When the RCL or RCLP instruction is executed, the S1 bits are rotated to the left S2 number of bits. The topmost bit (the most significant bit) is stored in a carry flag, and the carry flag (1 or 0) is rotated to the bottom-most bit (the least significant bit).

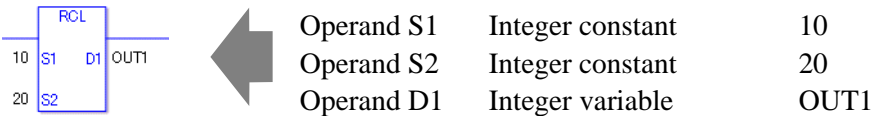
The result is stored in D1. The RCL and RCLP instructions always pass power. When using the RCL and RCLP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in the S1 and D1 operands. Refer to the following for specifying a constant.

- S1: Rotation address Specifies an address to rotate bits.
- S2: Number of bits to rotate Specifies the number of bits to rotate.
- D1: Storage device Specifies an address for storing the results after rotating bits.

For example, When 1 bit is rotated to left (with carry over)

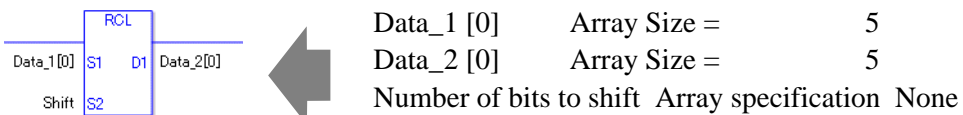
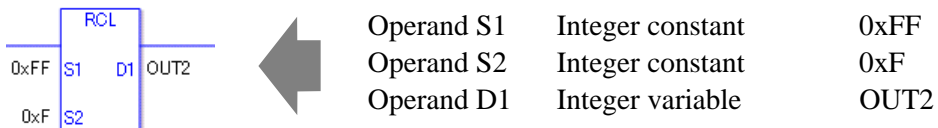


When operand D1 is an integer variable



When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal values.



If both S1 and D1 are not an array, 32 bits are rotated with carry over.
 For S2, specify a value between 0 and 32.

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

If an overflow results from the rotation instruction, the overflowed bit is stored in #L_CalcCarry.

When the execution results in an error, the error information is stored in #L_Status.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

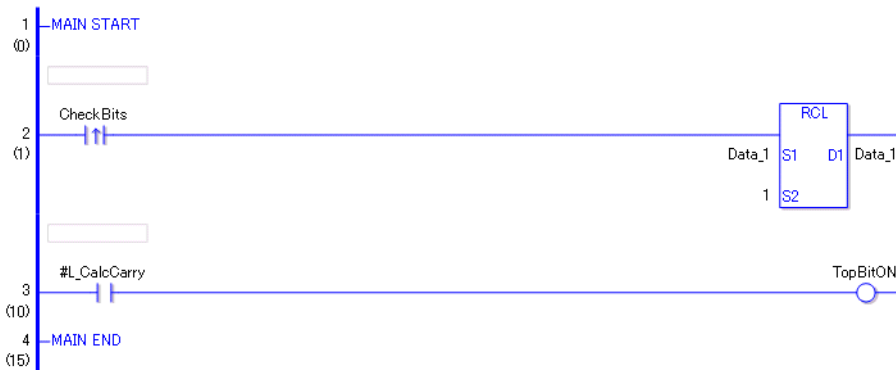
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

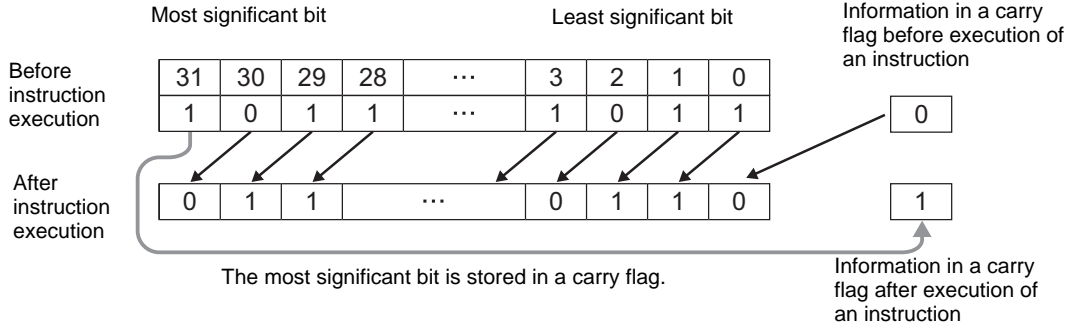
Program Example

RCL



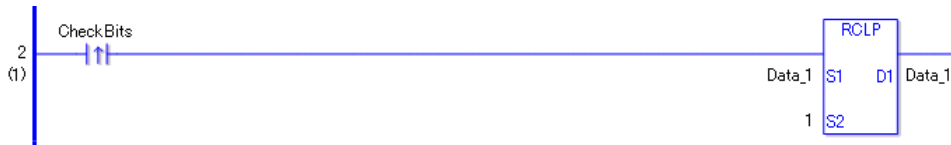
- (1) When the positive transition instruction turns ON, the RCL instruction is executed. When the RCL instruction is executed, the result from rotating 1 bit with carry over is stored in D1.
- (2) When 1 bit is shifted to the left with carry over, you can use #L_CalcCarry to check the value of the most significant bit before the rotate operation.

(Supplementary) When using a normally open instruction, the RCL instruction is always executed as long as the bit is ON.



Program Example

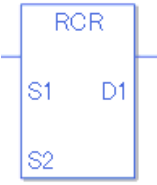

RCLP



The RCLP and RCL instructions have different ways of detecting when to execute. In the RCLP instruction, Even when using a normally open instruction, the RCLP instruction executes only when it detects the upward transition. Therefore, the RCLP instruction is executed only for one scan, even when the bit remains ON.

■ **RCR and RCRP (Rotate right with carry over)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
RCR (Rotate Right with Carry-over - Level Sensitive)		Rotate	4 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
RCRP (Rotate Right with Carry-over - positive transition)		Rotate	4 to 10

◆ **Operand Settings**

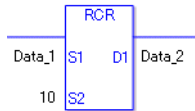
The following table lists the specifiable content of operands S1, S2, and D1 for the RCR and RCRP instructions.

The actual number of steps in the RCR and RCRP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Converting the number of steps in RCR and RCRP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{Data_2 = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in RCR and RCRP instructions

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (S2) in RCR and RCRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	0 to 32	1	O
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in RCR and RCRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (Output included)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
Specify float variable[variable]		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

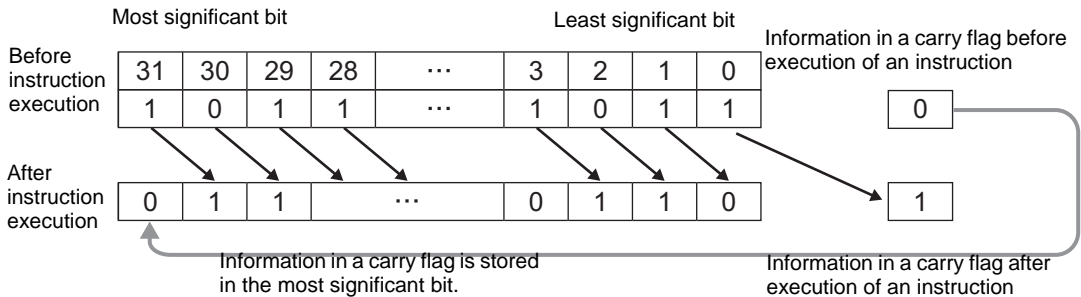
◆ **Explanation of the RCR and RCRP Instructions**

When the RCR or RCRP instruction is executed, the S1 bits are rotated to the right S2 number of bits. The bottom-most bit (the least most bit) is stored in a carry flag and the carry flag (1 or 0) is rotated to the topmost bit (the most significant bit).

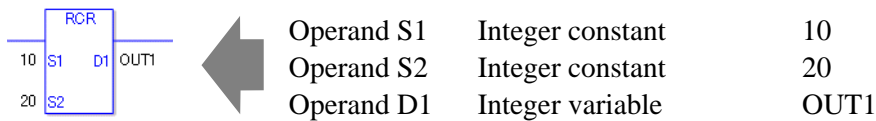
The result is stored in D1. The RCR and RCRP instructions always pass power. When using the RCR and RCRP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in the S1 and D1 operands. Refer to the following for specifying a constant.

- S1: Rotation address Specifies an address to rotate bits.
- S2: Number of bits to rotate Specifies the number of bits to rotate.
- D1: Storage device Specifies an address for storing the results after rotating bits.

For example, When 1 bit is rotated to right (with carry over)

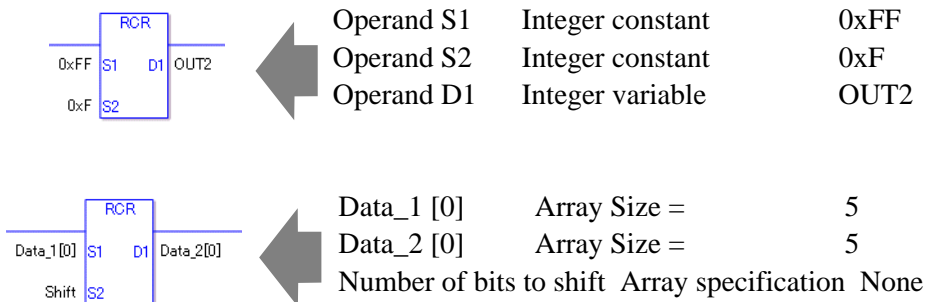


When operand D1 is an integer variable



When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal values.



If both S1 and D1 are not an array, 32 bits are rotated with carry over. For S2, specify a value between 0 and 32.

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

If an overflow results from the rotation instruction, the overflowed bit is stored in #L_CalcCarry.

When the execution results in an error, the error information is stored in #L_Status.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

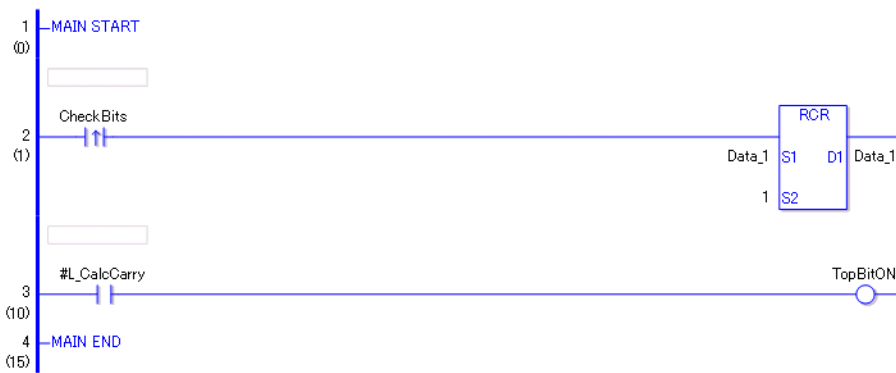
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

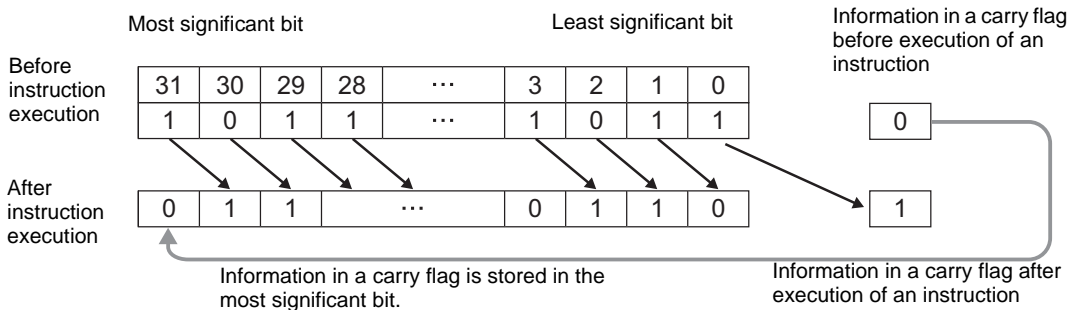
RCR



(1) When the positive transition instruction turns ON, the RCR instruction will be executed. When the RCR instruction is executed, the result of rotating 1 bit with carry over is stored in D1.

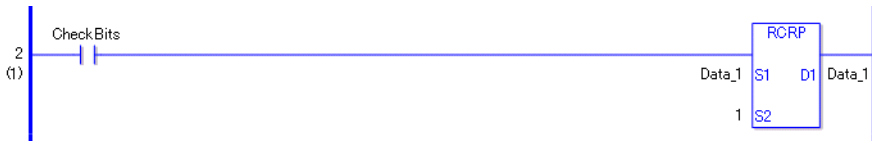
(2) When 1 bit is shifted to the right with carry over, you can use #L_CalcCarry to check the value of the least significant bit before rotation.

(Supplementary) When using a normally open instruction, as long as the bit is ON, the RCR instruction is always executed.



Program Example

RCRP

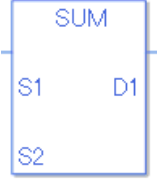
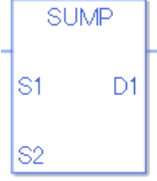


The RCRP and RCR instructions have different ways of detecting when to execute. In the RCRP instruction, even when using a normally open instruction, only the upward transition is detected, and the RCRP instruction is executed. Therefore, the RCRP instruction is executed only for one scan, even when the bit remains ON.

29.5.13 Function Instruction

■ SUM/SUMP (Total)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SUM (Total - Level Sensitive)		Operation	6 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SUMP (Total - positive transition)		Operation	6 to 10

◆ Operand Settings

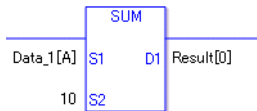
The following describes the specifiable content of operands S1, S2, D1 for the SUM/SUMP instructions.

The actual number of steps in the SUM/SUMP instructions depends on the operand specification method. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Calculate the number of steps in the SUM/SUMP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 [0] = 2 \text{ steps}\} + \{10 = 1 \text{ step}\} + \{Result [0] = 2 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand S1 in the SUM/SUMP instructions

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (not including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand S2 in the SUM/SUMP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	—	1 to 4096	1	O

◆ **Operand Settings**

The following describes the specifiable content of Operand D1 in the SUM/SUMP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (Output included)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	2	O
Specify float variable[variable]		3	O	

Continued

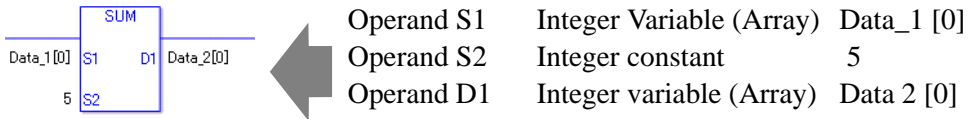
Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the SUB and SUMP Instructions**

The SUM/SUMP instructions both calculate sums. When the SUM instruction is executed, S2 array elements beginning at address S1 are totaled and the result is saved to D1. The SUM/SUMP instructions always pass power. If the variables designated to operands S1 and D1 are not the same type, an error will occur when using SUM/SUMP instructions. Designate the same variable type in operands S1 and D1.

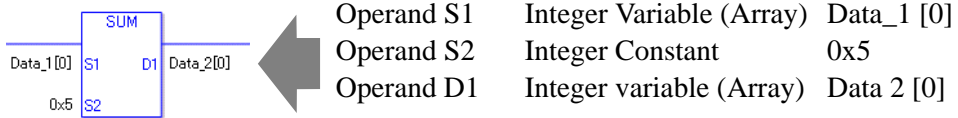
Refer to the following for specifying a constant.

When operand D1 is an integer variable



When operand D1 is an integer variable and you want to input hexadecimal values in operand S2.

When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal values.



Confirming Execution Results

(1) The instruction will not execute if the value in operand S1 or S2 (infinite or non-numeric value) cannot be recognized. For the error check, the error code “6706” is set for the #L_CalcErrCode.

The output result D1 maintains the value from the previous instruction executed successfully.

◆ **System Variables Indicating Execution Results**

When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

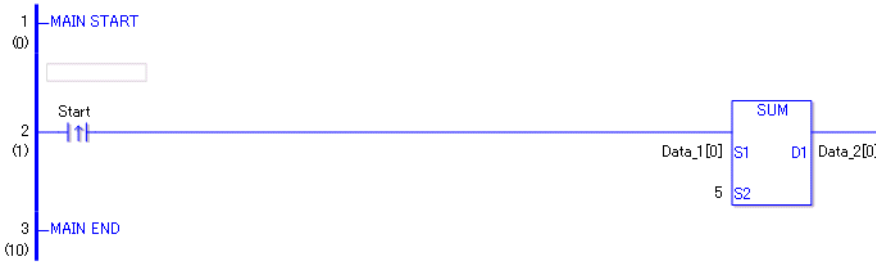
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

SUM

Totals 1 through 5 in Data_1 and saves the total in Data_2.

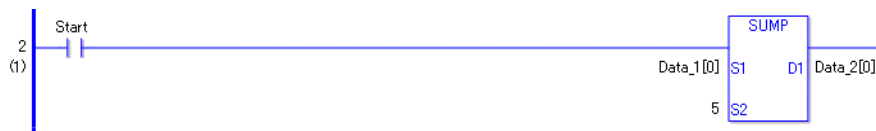


(1) When the positive transition instruction turns on and passes power to the SUM instruction, it totals 5 data elements in Data_1 beginning with element 0 and stores the result in D1 (Data_2). When using a normally open instruction, as long as the instruction is passing power, the SUM instruction continually executes at each scan, performing the sum operation.

Array Variable Name	Data 1	5 Executed Instructions	Save in	Data 2
Element	Data_1 [0]	+	→	Data_2 [0]
	Data_1 [1]	+		Data_2 [1]
	Data_1 [2]	+		Data_2 [2]
	Data_1 [3]	+		Data_2 [3]
	Data_1 [4]	+		Data_2 [4]
	Data_1 [5]			Data_2 [5]
	Data_1 [6]			Data_2 [6]
	Data_1 [7]			Data_2 [7]
	Data_1 [8]			Data_2 [8]
	Data_1 [9]			Data_2 [9]
	Data_1 [10]			Data_2 [10]

Program Example



SUMP



(1) The SUMP and SUM instructions differ in how they detect the instruction start. The SUMP instruction only detects the upward transition and executes the SUMP instruction even when using a normally open instruction. Even if the variable of the NO instruction stays ON, the SUMP instruction is executed only once (on the first scan).

■ **AVE/AVEP (Average)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
AVE (Average - Level Sensitive)		Operation	6 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
AVEP (Average - positive transition)		Operation	6 to 10

◆ **Operand Settings**

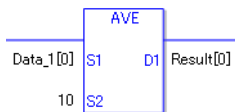
The following describes the specifiable content of operands S1, S2 and D1 for the AVE/AVEP instructions.

The actual number of steps in the AVE/AVEP instructions depends on the operand specification method. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Calculate the number of steps in the AVE/AVEP Instructions

(For the number of steps in the operand, refer to the operand settings in the next section.)



$$\{ \text{Data}_1 [0] = 2 \text{ steps} \} + \{ 10 = 1 \text{ step} \} + \{ \text{Result} [0] = 2 \text{ steps} \} + \{ 1 \text{ step} \} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand S1 in the AVE/AVEP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (not including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	2	O
Specify float variable[variable]		3	O	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand S2 in the AVE/AVEP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	—	1 to 4096	1	O

◆ **Operand Settings**

The following describes the specifiable content of Operand D1 in the AVE/AVEP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (Output included)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]	2	O	
		Specify integer variable [Variable]	3	O	
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X	
	Float	—		—	X
		Specify float variable[constant]	2	O	
Specify float variable[variable]		3	O		

Continued

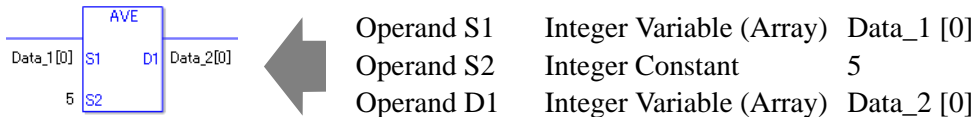
Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the AVE and AVEP Instructions**

The AVE/AVEP instructions both calculate the average. When the AVE instruction is executed, S2 array elements beginning at address S1 are averaged and the result is saved in D1. The AVE/AVEP instructions always pass power. If the variables designated to operands S1 and D1 are not the same type, an error will occur when using the AVE/AVEP instructions. Designate the same variable type in operands S1 and D1.

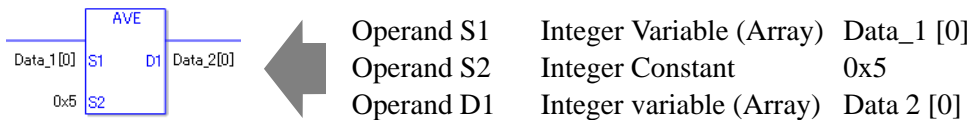
Refer to the following for specifying a constant.

When operand D1 is an integer variable



When operand D1 is an integer variable and you want to input hexadecimal values in operand S2.

When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal values.



Confirming Execution Results

(1) The instruction will not execute if the value in operand S1 or S2 (infinite or non-numeric value) cannot be recognized. For the error check, the error code “6706” is set for the #L_CalcErrCode.

The output result D1 maintains the value from the previous instruction executed successfully.

◆ **System Variables Indicating Execution Results**

When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

If There are no items to be calculated, the total is zero and the result is zero.

Program Example

AVE

Averages 1 through 5 in Data_1 and saves the result in Data_2.

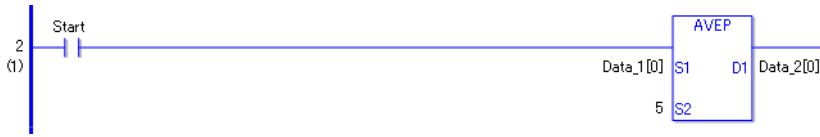


- (1) When the positive transition instruction turns ON, the AVE instruction will be executed. When the AVE instruction is executed, the average of array elements 0 through 4 of Data 1 are calculated and the result at D1 is stored in Data_2. When using a normally-open instruction, as long as the instruction variable is ON, the AVE instruction is always executed.

Array Variable Name	Data 1	5 Executed Instructions	Save in	Data 2
Element	Data_1 [0]	+	→	Data_2 [0]
	Data_1 [1]	+		Data_2 [1]
	Data_1 [2]	+ divide by 5		Data_2 [2]
	Data_1 [3]	+		Data_2 [3]
	Data_1 [4]	+		Data_2 [4]
	Data_1 [5]			Data_2 [5]
	Data_1 [6]			Data_2 [6]
	Data_1 [7]			Data_2 [7]
	Data_1 [8]			Data_2 [8]
	Data_1 [9]			Data_2 [9]
	Data_1 [10]			Data_2 [10]

Program Example



AVEP



- (1) AVEP and AVE instructions differ in how they detect the instruction start. The AVEP only detects the upward transition and executes the AVEP instruction even when using a normally open instruction. Even if the variable of the NO instruction stays ON, the AVEP instruction is executed only once (on the first scan).

■ **SQRT/SQ RTP (Square Root)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SQRT (Square Root - Level Sensitive)		Operation	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SQ RTP (Square Root - positive transition)		Operation	3 to 7

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1 and D1) in the SQRT/SQ RTP instructions.

The actual number of steps in the SQRT/SQ RTP instructions depends on the operand specification method. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in the instruction.

For example, Calculate the number of steps in the SQRT/SQ RTP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{\text{Data}_1[0] = 2 \text{ steps}\} + \{\text{Result}[N] = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1 and D1) in the SQRTP/SQRTP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (not including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	—	X
		Specify integer variable [Variable]	—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	Float Variable	1	O
		Specify float variable[constant]	2	O
Specify float variable[variable]		3	O	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	Real Variable	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant (Cannot use for D1)	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O

◆ **Explanation of the SQRT and SQRTP Instructions**

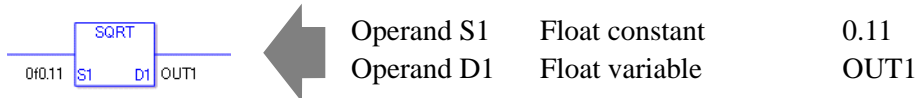
The SQRT/SQRTP instructions both calculate the square root. When the SQRT instruction is executed, the square root of S1 is calculated and the value is saved in D1.

The SQRT/SQRTP instructions always pass power. If the variables designated in operands S1 and D1 are not the same type, an error will occur when using the SQRT/SQRTP instructions. Designate the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

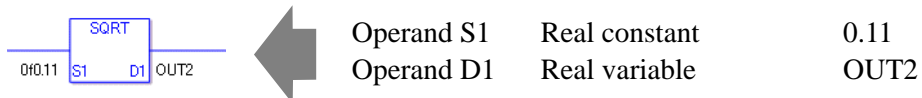
When operand D1 is a float variable

When 0f (zero and lower case "f") is input, the following values become float values.



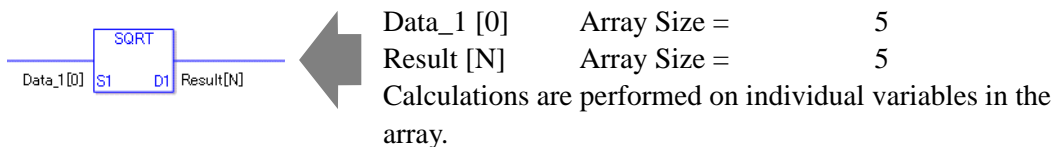
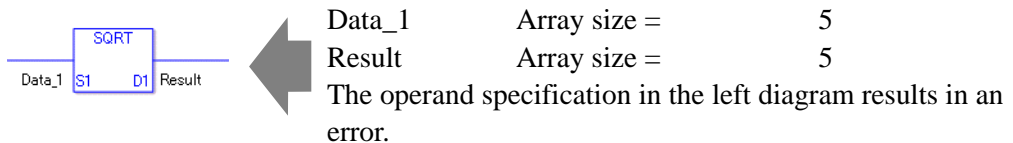
When operand D1 is a real variable

When 0r (zero and lower case "r") is input, the following values are interpreted as real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the variables are the same type.



Confirming Execution Results

(1) The instruction will not execute if the value in operand S1 or S2 (infinite or non-numeric value) cannot be recognized. For the error check, the error code “6706” is set for the #L_CalcErrCode.

The output result D1 maintains the value from the previous instruction executed successfully.

◆ System Variables Indicating Execution Results

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

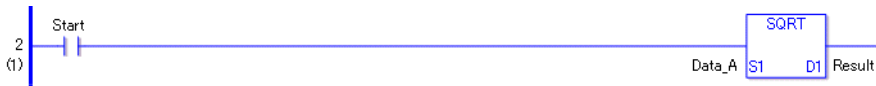
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

SQRT



(1) When the positive transition instruction turns ON, the SQRT instruction will be executed.

When the SQRT instruction is executed, the square root of Data_A is stored in the calculation result (real/float variable) in D1.

When using a normally-open instruction, as long as the instruction variable is ON, the SQRT instruction is always executed.

Program Example



SQRTP



(1) The SQRTP and SQRT instructions differ in how they detect the instruction start. The SQRTP only detects the upward transition and executes the SQRTP instruction even when using a normally open instruction. Even if the variable of the NO instruction stays ON, the SQRTP instruction is executed only once (on the first scan).

■ BCNT/BCNTP (Bit Count)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BCNT (Bit Count - Level Sensitive)		Operation	3 to 9
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BCNTP (Bit Count - positive transition)		Operation	3 to 9

◆ Operand Settings

The following describes the specifiable content of operands S1 and D1 for the BCNT/BCNTP instructions.

The actual number of steps in the BCNT/BCNTP instructions depends on the operand specification method. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

For example, Calculate the number of steps in the BCNT/BCNTP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



{Data_1 [1] = 2 steps} + {Result [Specify indirectly] = 3 steps} + {1 step} = 6 steps

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table list the configurable conditions for Operands (S1 and D1) in the BCNT/BCNTP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format *(Note 1) S1 = IO Enabled D1 = IO Disabled	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format *(Note 1) S1 = IO Enabled D1 = IO Disabled	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format *(Notes 2) D1 = Disabled	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_*(Notes 2)	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	

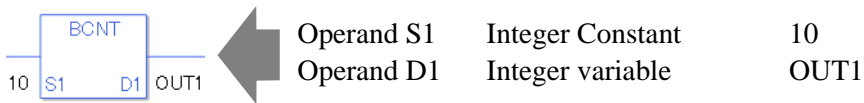
Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Constant *(Notes 3) D1 = Disabled	Integer *(Notes 3)	- 2147483648 to 2147483647	1	O
	Float	±1.175494351e-38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the BCNT and BCNTP Instructions**

The BCNT/BCNTP instructions both count bits. When the BCNT instruction is executed, the ON bits in S1 data are counted and the number of ON bits is saved in D1. The BCNT/BCNTP instructions always pass power. If the variables designated to operands S1 and D1 are not the same type, an error will occur when using the BCNT/BCNTP instructions. Specify the same variable type in operands S1 and D1. Refer to the following for specifying a constant.

When operand D1 is an integer variable



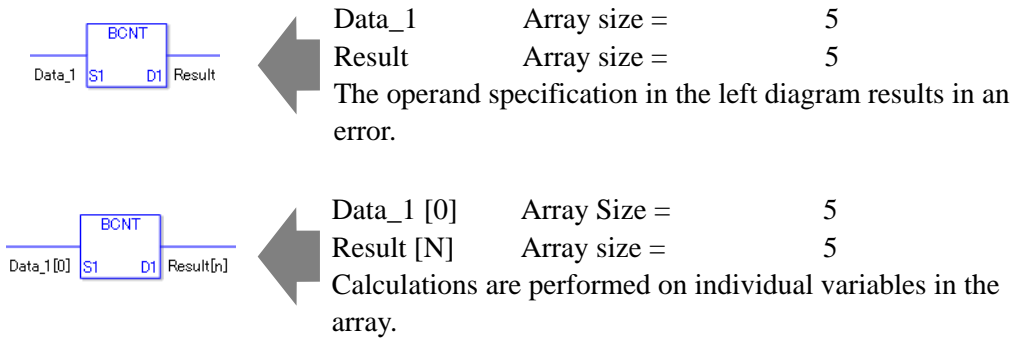
When operand D1 is an integer variable and you want to input hexadecimal values in operand S2.

When 0x (zero and lower case x) is input, the following values will be interpreted as hexadecimal values.



When calculating data in a specified array (integer variable array), use Data [0] or Data [N] (N is an integer variable) to specify the array.

When operands S1 and D1 specify the entire array, an error will occur even if the variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.
 When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

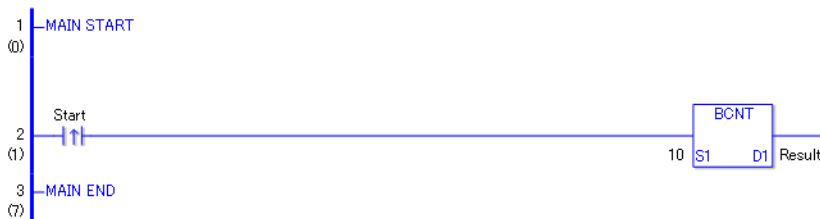
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

BCNT

Counts the number of bits that are ON, and saves the number in an integer variable.



When the positive transition instruction turns ON, the BCNT instruction will be executed. When the BCNT instruction is executed, the ON bits in the value 10 (binary 1010) are counted and the result of 2 is saved in the result data. The result data is configured in D1. When using a normally-open instruction, as long as the instruction variable is ON, the BCNT instruction is always executed.

Program Example

BCNTP



- (1) The BCNTP and BCNT instructions differ in how they detect the instruction start, The BCNTP only detects the upward transition and executes the BCNTP instruction even when using a normally open instruction. Even if the variable of the NO instruction stays ON, the BCNTP instruction is executed only once (on the first scan).

■ **PID**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
PID (PID - Level Sensitive)	<p style="text-align: center;">HP</p>	Operation	10 to 18

◆ **Explanation of the PID Instruction**

The PID variable in the PID instruction is a structure variable. You cannot allocate variables other than PID variables (address format: U_) to operand HP. For the internal structure of the PID variable designated to operand HP, refer to the following table.

PID Variable

PID Variable	Variables Settings	Description
VariableName.Q	Bit Variable	PID Instruction Processing Completion Flag
VariableName.PF	Bit Variable	Processing Deadband Range Flag
VariableName.UO	Bit Variable	Output Values over the Upper Limit
VariableName.TO	Bit Variable	Output Values over the Lower Limit
VariableName.IF	Bit Variable	Integral Setting
VariableName.KP	Integer Variable	Proportional Constant
VariableName.TR	Integer Variable	Integral Calculus Time
VariableName.TD	Integer Variable	Differential Calculus Time
VariableName.PA	Integer Variable	Processing Deadband Range
VariableName.BA	Integer Variable	Bias (Offset)
VariableName.ST	Integer Variable	Frequency in Sampling

Other operands are as follows.

S1: Setpoint

S2: Preset Value

S3: Tieback Value (The set value is output when an instruction is disabled)

D1: Output Value

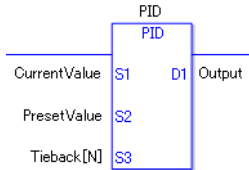
◆ **Operand Settings**

The following describes the specifiable content of operands S1, S2, S3, and D1 for the PID instruction. The actual number of steps in the PID instruction depends on the operand specification method. The following describes how to calculate the number of steps.

Number steps in HP operand + Number of steps in operand S1 + Number of steps in operand S2 + Number of steps in operand S3 + Number of steps in operand D1 + 5 = Total number of steps in one instruction

For example, Calculate the number of steps in the PID instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



{PID control = 1 step (PID variable in HP operand is fixed to 1 step)} + {Current value = 1 step} + {Setting = 1 step} + {Tieback value [N] = 3 steps} + {Output = 1 step} + {5 steps} = 12 steps

The last 5 steps are included in the PID instruction. Be sure to add 5 steps.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2, S3, and D1) in the PID instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format *(Note 1) S1, S2, S3 = IO possible D1 = IO Disabled	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Address Format *(Notes 2) D1 = Not possible	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_*(Notes 2)	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	2	O
		D_****.B/W [address]	3	O
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant *(Notes 3) D1, S2 = Not possible	Integer *(Notes 3)	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Basic Function**

The PID instruction compares the measured values (current values) and the set values (target values). The measured values are based on analog input and temperature input. The instruction then adjusts the output values to even the gap between the current values and the target values. You can combine P control, I control, and D control in PID control. Specify each below mentioned parameter to be controlled.

The output value calculated by the PID control is generally expressed by the following formula.

$$CV = KP(E + \text{Reset} \int_0^t (E)dt + \text{Rate} \frac{d(E)}{dt})$$

- KP: Proportional constant
- E: Deviation (SP-PV or PV-SP)
- Reset: Integral Cycles
- Rate: Differential calculus time

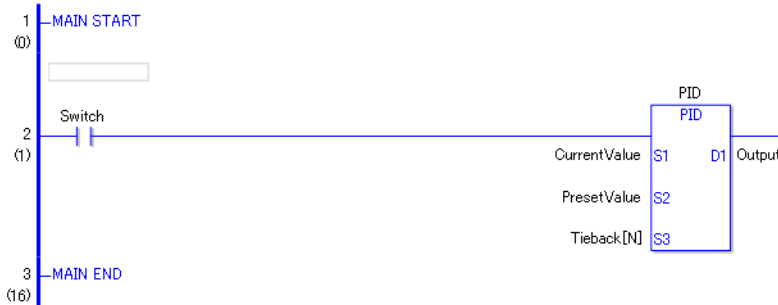
Using the [Tuning] tab that will be described later, adjust the sampling time to reduce the effect of noise on deviation. The following formula shows the result of filtering on the deviation.

$$EF_n = EF_{n-1} + \frac{T_{Loop}}{T_{Filter}} (E_n - EF_{n-1})$$

- EF: Result of filtering on deviation
- T_{loop}: Frequency data
- T_{Filter}: Sampling Frequency
- E: Deviation (SP-PV or PV-SP)

◆ **Function Summary**

When the PID instruction is enabled, the PID is calculated and the operation volume is adjusted and output (calculated). When the instruction is disabled as below, it outputs the Tieback value. The Tieback value is specified in S3. Input the constant 0 if no output is necessary when the instruction is disabled.



To use the PID instruction in a logic program, allocate variables to the PID variable operand (HP) and the integer variable operands (S1, S2, S3 and D1) first.

PID Variable

When you allocate a variable to the PID instruction operand HP, a member is automatically allocated to the variable.

PID Variable

PID Variable	Variables Settings	Description
VariableName.Q	Bit Variable	PID Instruction Processing Completion Flag
VariableName.PF	Bit Variable	Processing Deadband Range Flag
VariableName.UO	Bit Variable	Output Values over the Upper Limit
VariableName.TO	Bit Variable	Output Values over the Lower Limit
VariableName.IF	Bit Variable	Integral Setting
VariableName.KP	Integer Variable	Proportional Constant
VariableName.TR	Integer Variable	Integral Calculus Time
VariableName.TD	Integer Variable	Differential Calculus Time
VariableName.PA	Integer Variable	Processing Deadband Range
VariableName.BA	Integer Variable	Bias (Offset)
VariableName.ST	Integer Variable	Frequency in Sampling

- Values assigned to a proportional constant, integral calculus time, and differential calculus time look different when input in "PID Monitor" from when input to each of the PID variables in a program. When inputting the values in the program, multiply the values by 1000 for the proportional constant, the integral calculus times and the differential calculus times.

For example Proportional constant 0.1 x 1000 -> 100 -> 100

(Notes)

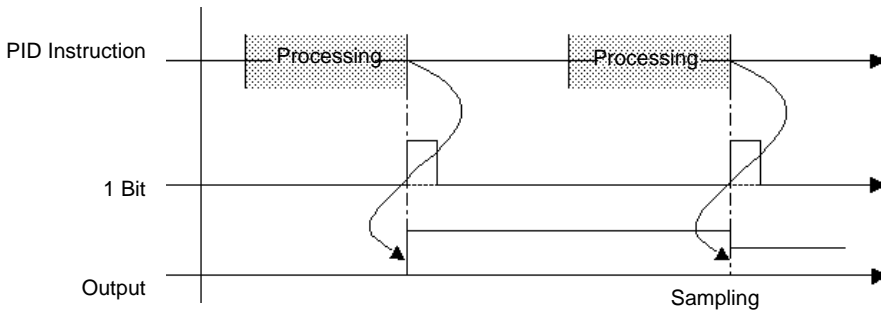
All the PID variables are keep-type variables. Up to 8 PID instructions are allowed per project.

1 PID instruction can be specified for 1 PID variable.

◆ **Explanation of the PID Variable Members**

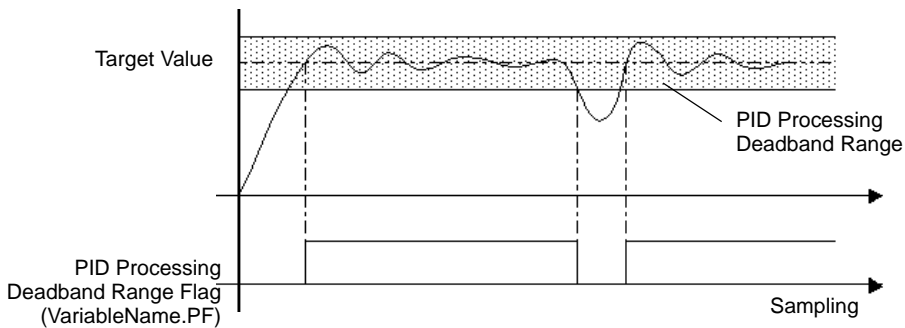
PID Instruction Processing Completion Flag (VariableName.Q)

Upon the value being output to the operand D1 after the processing, .Q turns on. The completion flag of the PID instruction turns on while 1 scan is being executed.



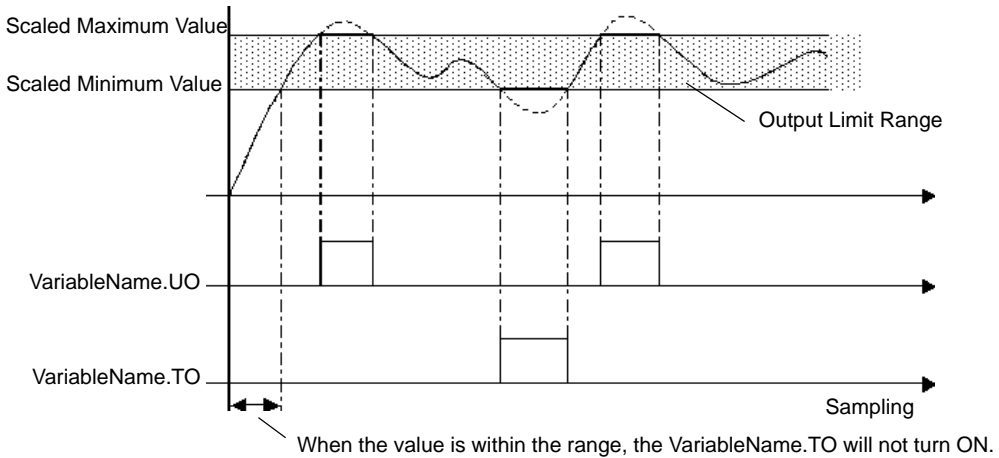
Processing Deadband Range Flag (VariableName.PF)

The flag turns on when the current value reaches the target value within the range specified by specifying PID variables (process invalid range VariableName.PF) and turns off when the current value becomes out of range.



Output Values over the Upper/Lower Limits (VariableName.UO, VariableName.TO)

Double-click the PID instruction to display a dialog box for specifying the PID variable output range. If the calculated result exceeds the specified output value, the VariableName.UO turns ON. When the result is below the specified lower limit, the VariableName.TO turns ON. The PID continues even when the status bits turn ON and the calculated value is output as either the specified upper or lower limit.



Integral Setting (variable name .IF)

Double-click the PID instruction to display a dialog box for setting a range to execute the PID instruction. If the result is out of the integral setting specified, the .IF turns on. The integral setting of each status executes integral calculation only within the range.

Proportional Constant (VariableName.KP)

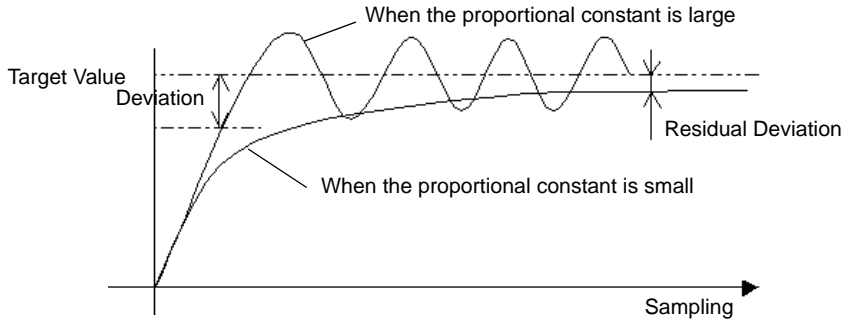
Specify a proportional constant (VariableName.KP) to output a value corresponding to the deviation between the target and current values.

A smaller proportional constant produces a smaller output value to reach the target value, and eliminates overshoot but may increase the residual deviation. A larger proportional constant produces a larger output value to reach the target value and reduces the time to reach the target, but may result in hunting.

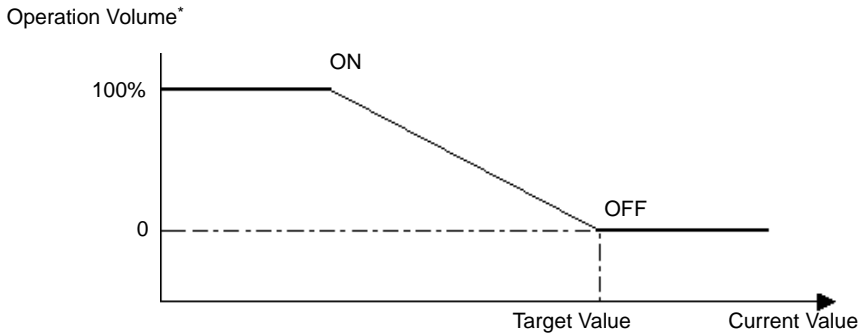
Settings range from 0.01 to 1000.00 Internal data are integer variables. Decimals cannot be used.

To set 0.01, specify $0.01 \times 1000 = 10$.

Specify variable.KP as the value multiplied by 1000.



(Note) In the proportional control, the operation volume will be the maximum 100% if the current value is smaller than the target value. The operation value will be 0% if the target value and the current value match (no deviation).



* Operation Volume: Output per unit time

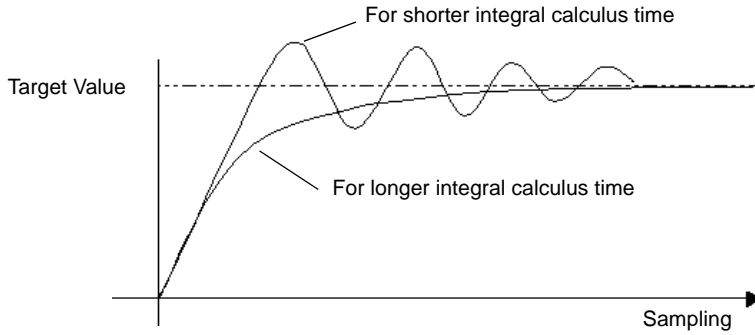
Integral Calculus Time (VariableName.TR)

By setting the integral calculus time (.TR), you can eliminate a deviation to the target value. As the operation volume approaches the target value using proportional control, the deviation becomes so small that the operation volume cannot obtain enough values to offset the deviation. The slight deviation is called a residual deviation. The deviation can be eliminated by the integral control. The integral control adjusts the deviation by increasing the operation volume when the deviation accumulated timewise reaches a certain size. As the integral calculus time becomes short, the operation volume to reach the target value becomes larger, causing overshoot and hunting and reaching the target in a shorter time. Likewise, as the integral calculus time becomes longer, the operation volume to reach the target value becomes smaller, reducing overshoot and hunting, but it takes longer to reach the target. The integral calculus time specifies an interval time (in seconds) for executing integral processing.

Settings range from 0.100 to 3000.000 Internal data are integer variables. Decimals cannot be used.

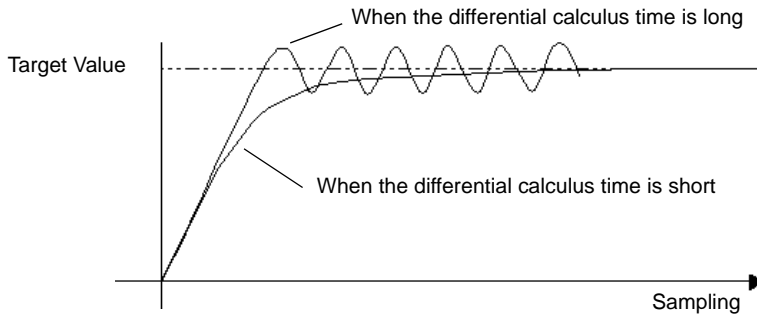
To set 0.1, specify $0.1 \times 1000 = 100$.

Specify variable.TR as the value multiplied by 1000.



Differential Calculus Time (VariableName.TD)

By setting the differential calculus time (.TD), you can respond to any change quickly. The proportional control and integral control require a certain amount of time (time constant) and cannot respond immediately to external disturbances. It takes time to return to the original target value. The differential control responds promptly and assigns a large operation volume when the gap between the current and previous deviations is large compared to the external disturbance. A longer differential calculus time requires shorter time to recover from the effects of external disturbances, but results in overshoot and frequent hunting. A shorter differential calculus time reduces overshoot and hunting but takes more time to recover from the effects of external disturbances.



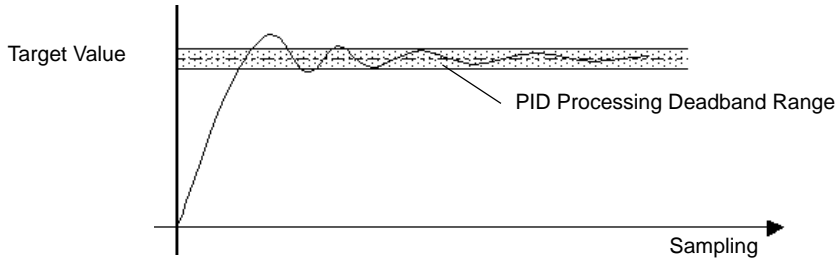
The setting range 0.00 to 3000.00 internal data becomes integer variables and decimals become unavailable.

For setting 0.1, use $0.1 \times 1000 = 100$.

Specify the value multiplied by 1000 for the variable name .TD.

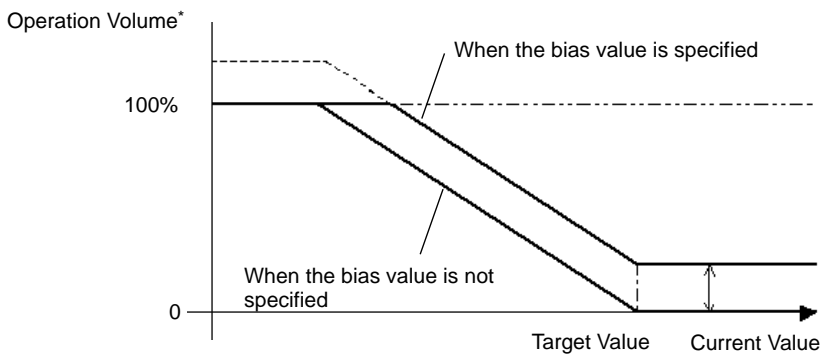
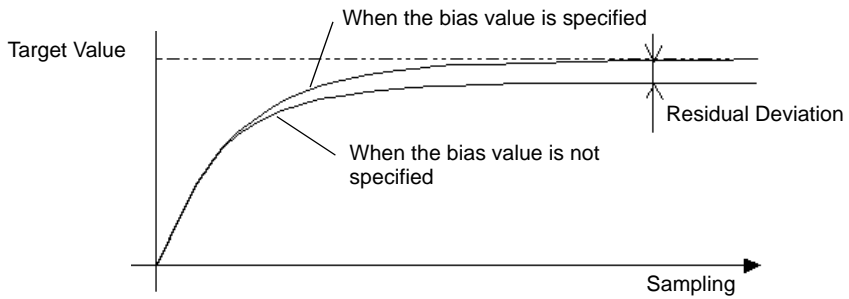
Processing Deadband Range (VariableName.PA)

In the "processing deadband range," PID control does not occur and the minimum value is output for smooth control without hunting.



Bias (VariableName.BA)

Sets the bias value (offset). This reduces any residual deviation incurred in the proportional control.



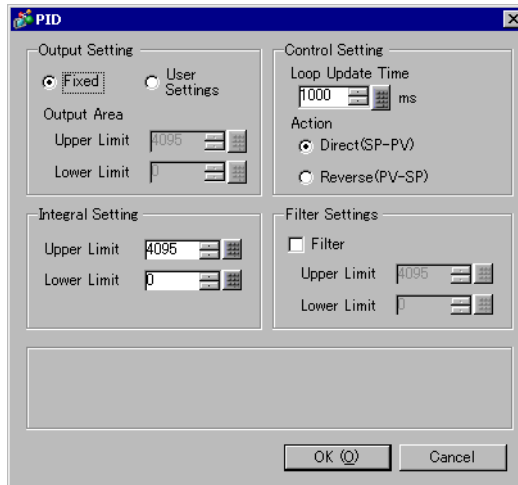
* Operation Volume: Output per unit time

Sampling Frequency (VariableName.ST)

This eliminates noise in the S2 value obtained in the control setting frequency. The moving average is calculated based on the previous filtering result and the newly obtained data. Specifying the sampling frequency minimizes the effect on the output value when the current data contain unexpected values. This is because the average of the previously measured data and the current data is used for the calculation. Specify a larger value than the control setting frequency for the sampling frequency. Specify 0 for the sampling frequency to disable the filter.

◆ Set Up by Double-clicking the PID Instruction

Double-click the PID instruction to specify the PID variables.



Output Setting (Range of Operand D1)

Specifies the upper and lower limits for the output value. The result of the calculation must be within this range.

Fixed Settings

The Output range is 0 to 4095.

User Settings

Specify the output range as required.

Range for the Upper Limit Lower Limit +1 to 32767

Range for the Lower Limit 0 to Upper Limit-1

Integral Setting

Specifies the upper and lower limits for the integral settings.

Control Setting

Loop Update Time: Sets the temporal frequency of obtaining S2 data. The frequency of obtaining data is also the frequency of updating the D1 output.

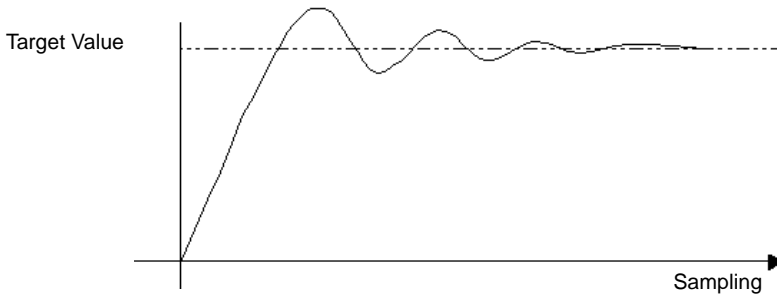
You can use the filtering feature to specify the frequency. The sampling frequency must be larger than the frequency of obtaining data.

Settings range from 10 to 65535 ms

Action:

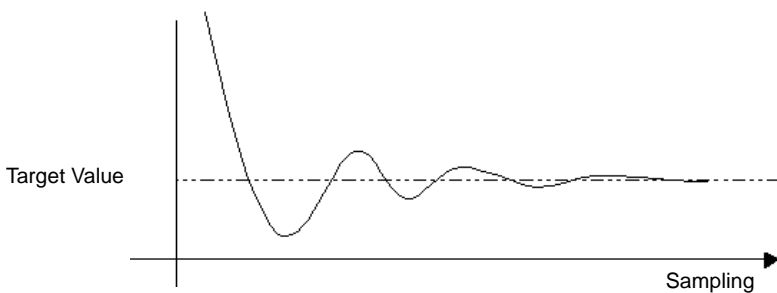
Direct (D1–D2)

Used to control the increase in operation volume when the process variable is smaller than the setpoint. (heating, and so on)



Reverse (D1–D2)

Used to control the increase in operation volume when the process variable is larger than the setpoint, such as cooling.



Filter Settings

Specifies the upper and lower limits for the output value. If the value exceeds the range, the value will be output as either the upper or lower limit. When the value exceeds the range, the bits above the upper and lower limits (VariableName.UO, VariableName.TO) turn ON.

Settings Range

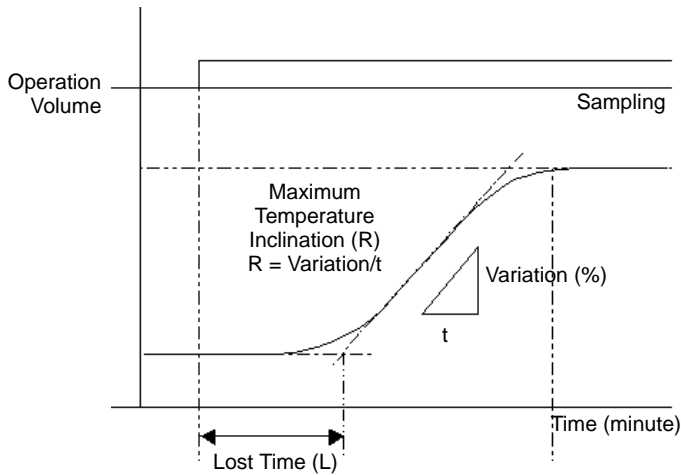
Dependent on the Output Settings Range

Upper Limit Output Settings Range (upper limit) to 32767

Lower Limit Output Settings Range (lower limit) to -32768

◆ **PID Constant Adjust**

The following explanation uses temperature control as an example. To optimize the result of the PID control, you need to optimize the constant values of P (proportional element), I (Integral element), and D (differential element). You can use the step-response method to derive a PID temperature constant for various setpoints. Note that the value might not be optimized depending on the use and the setpoint. In that case, perform online monitoring and adjust the value in the PID monitor window. Specify the setpoint value for the step-response method and output 100% of the operation volume onto the control target step. At this time, measure the maximum temperature inclination (R) and lost time (L) in the temperature graph shown below.



Insert the measured values for maximum temperature slope (R) and lost time (L) in the following formula to calculate the proportional constant, integral calculus time, and differential calculus time constants. Assign the calculated values to the values in the PID monitor window.

"Proportional Constant" = $100/(0.83 \cdot R \cdot L)$ [%]



"Integral Calculus Time" = $1/(2 \cdot L)$ [events/min] (formula = unidentified)

"Differential Calculus Time" = $0.5 \cdot L$ [min]

29.5.14 Function Instruction

■ SIN and SINP (Sine)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SIN (Sine - Level Sensitive)		Trigonometric Function	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SINP (Sine - positive transition)		Trigonometric Function	3 to 7

◆ Operand Settings

The following describes the specifiable content of operands S1 and D1 for the SIN and SINP instructions.

The actual number of steps in the SIN and SINP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in SIN and SINP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{\text{Data}_1 [0] = 2 \text{ steps}\} + \{\text{Result} [N] = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1) and (D1).

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
Specify real variable [variable]		3	O		

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Variable Format	Timer	.PT/.ET only	—	X	
	Counter	.PV/ .CV only	—	X	
	Date	.YR/ .MO/ .DAY only	—	X	
	Time	.HR/ .MIN/ .SEC only	—	X	
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W [constant]		—	X
		D_****.B/W [address]		—	X
	F_	—	1	O	
	R_	—	1	O	
	T_	.PT/.ET only	—	X	
	C_	.PV/ .CV only	—	X	
	N_	.YR/ .MO/ .DAY only	—	X	
	J_	.HR/ .MIN/ .SEC only	—	X	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X		
Constant (Cannot use for D1)	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O	

◆ **Explanation of the SIN and SINP Instructions**

The SIN and SINP instructions are sine instructions for trigonometric functions. The SIN instruction calculates the sine of S1 and stores the result in D1.

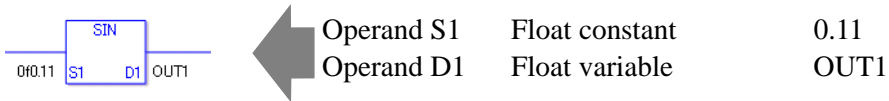
Enter the number of radians in S1 to get the result in D1 as a real value between -1.0 and 1.0.

The SIN and SINP instructions always pass power. When using the SIN and SINP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

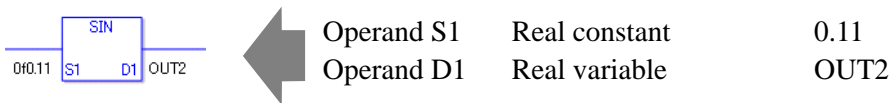
When operand D1 is a float variable

When Of (zero and lower case "f") is input, the following values become float values.



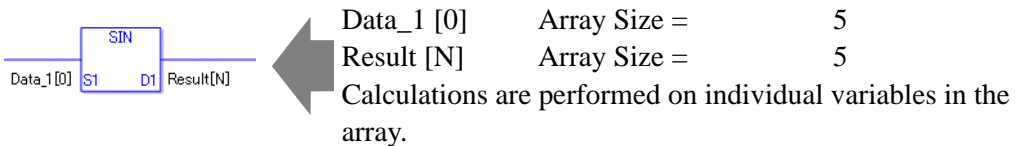
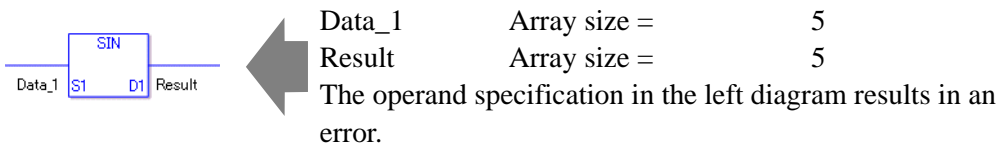
When operand D1 is a real variable

When Or (zero and lower case "r") is input, the following values become real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the variables are the same type.



System Variables Indicating Execution Results

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

SIN



- (1) The SIN instruction is executed when the positive transition instruction turns ON. The SIN instruction calculates the sine of Data_A and stores the result in D1. When using a normally open instruction, the SIN instruction is always executed as long as the normally open instruction is ON.

Program Example



SINP



- (1) SINP and SIN instructions differ in when they run. In SINP instructions, Even when using a normally open instruction, the SINP instruction executes only when it detects the upward transition. Therefore, the SINP instruction is executed only for one scan, even when the normally open instruction bit remains turn ON.

■ **COS and COSP (Cosine)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
COS (Cosine - Level Sensitive)		Trigonometric Function	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
COSP (Cosine - positive transition)		Trigonometric Function	3 to 7

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the COS and COSP instructions.

The actual number of steps in the COS and COSP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in COS and COSP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{ \text{Data}_1 [0] = 2 \text{ steps} \} + \{ \text{Result} [N] = 3 \text{ steps} \} + \{ 1 \text{ step} \} = 6 \text{ steps}$$

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the COS and COSP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (not including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant]	—	X
		Specify integer variable [Variable]	—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	Float Variable	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	Real Variable	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant (Cannot use for D1)	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O

◆ **Explanation of the COS and COSP Instructions**

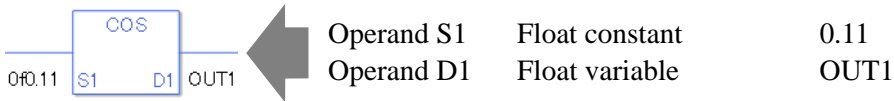
The COS and COSP instructions are cosine instructions for trigonometric functions. The COS instruction calculates the cosine of S1 and stores the result in D1. Enter the number of radians in S1 to get the result in D1 as a real value between -1.0 and 1.0.

The COS and COSP instructions are always conducted. When using the COS and COSP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

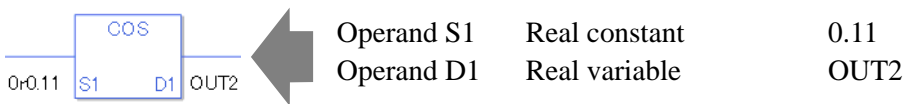
When operand D1 is a float variable

When 0f (zero and lower case "f") is input, the following values become float values.



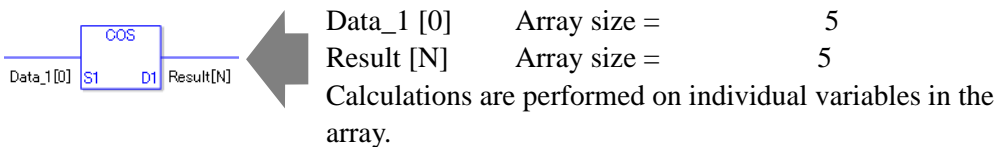
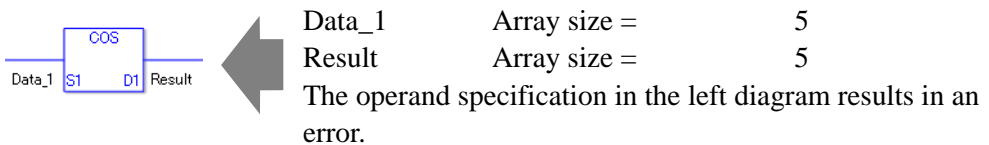
When operand D1 is a real variable

When 0r (zero and lower case "r") is input, the following values become real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

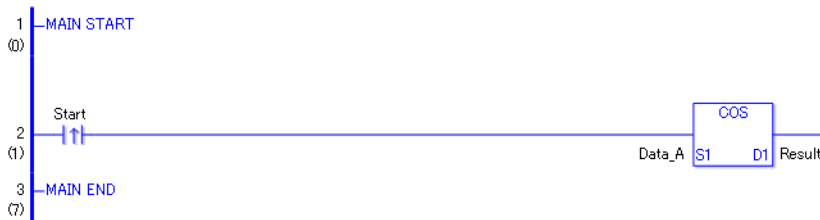
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

COS

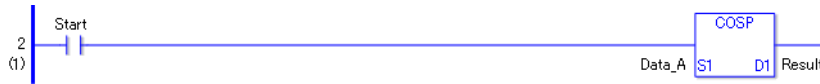


(1) The COS instruction is executed when the positive transition instruction turns ON. The COS instruction calculates the cosine of Data_A and stores the result in D1.

When using a normally open instruction, the COS instruction is always executed as long as the normally open instruction bit is ON.

Program Example



COSP



(1) The COSP and COS instructions differ in when they run. In COSP instructions, Even when using a normally open instruction, the COSP instruction executes only when it detects the upward transition. Therefore, the COSP instruction is executed only for one scan, even when the normally open instruction bit remains ON.

■ **TAN and TANP (Tangent)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
TAN (Tangent - Level Sensitive)		Trigonometric Function	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
TANP (Tangent - positive transition)		Trigonometric Function	3 to 7

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the TAN and TANP instructions.

The actual number of steps in the TAN and TANP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in TAN and TANP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 [0] = 2 \text{ steps}\} + \{Result [N] = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the TAN and TANP instructions

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
Specify real variable [variable]		3	O		

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Variable Format	Timer	.PT/.ET only	—	X	
	Counter	.PV/ .CV only	—	X	
	Date	.YR/ .MO/ .DAY only	—	X	
	Time	.HR/ .MIN/ .SEC only	—	X	
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	L_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	1	O	
	R_	—	1	O	
	T_	.PT/.ET only	—	X	
	C_	.PV/ .CV only	—	X	
	N_	.YR/ .MO/ .DAY only	—	X	
	J_	.HR/ .MIN/ .SEC only	—	X	
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	Integer	- 2147483648 to 2147483647	—	X	
	Float	±1.175494351e-38 to ±3.402823466e+38	1	O	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	O	

◆ **Explanation of the TAN and TANP Instructions**

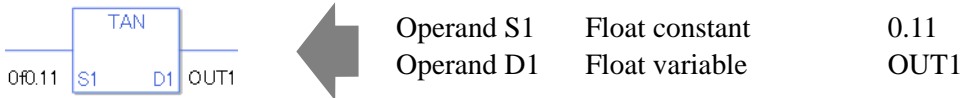
TAN and TANP instructions are tangent instructions for trigonometric functions. When the TAN instruction is executed and passes power, the value in S1 is TANed and the result is stored in D1. The S1 value is defined in radians and the D1 value results in a floating point number and should be set up with a real or float variable.

The TAN and TANP instructions always pass power. When using the TAN and TANP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

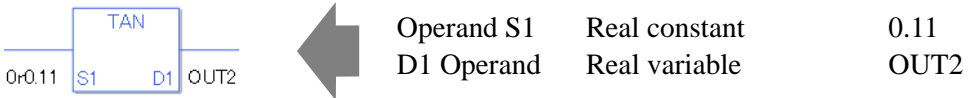
When operand D1 is a float variable

When Of (zero and lower case "f") is input, the following values become float values.



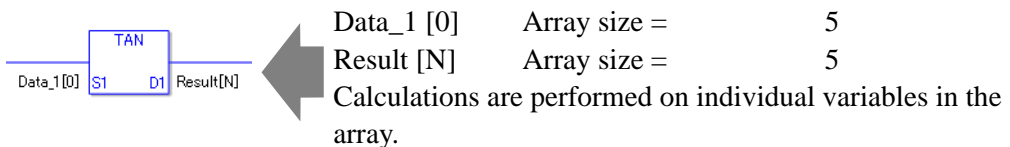
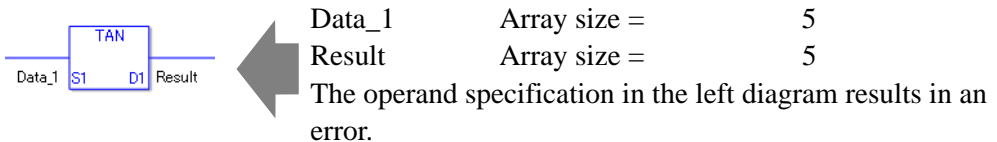
When operand D1 is a real variable

When Or (zero and lower case "r") is input, the following values become real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

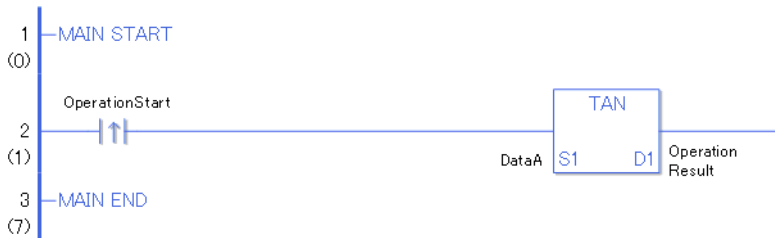
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

TAN

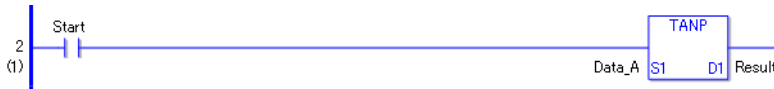


(1) The TAN instruction is executed when the positive transition instruction turns ON. The TAN instruction calculates the tangent of Data_A and stores the result in D1.

When using a normally open instruction, the TAN instruction is always executed as long as the normally open instruction bit is ON.

Program Example



TANP



(1) TANP and TAN instructions differ in when they run. In TANP instructions, even when using a normally open instruction, the TANP instruction executes only when it detects the upward transition. Therefore, the TANP instruction is executed only for one scan, even when the normally open instruction bit remains ON.

■ **ASIN and ASINP (Arc Sine)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ASIN (Arc Sine - Level Sensitive)		Trigonometric Function	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ASINP (Arc Sine - positive transition)		Trigonometric Function	3 to 7

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the ASIN and ASINP instructions.

The actual number of steps in the ASIN and ASINP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in ASIN and ASINP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{\text{Data}_1 [0] = 2 \text{ steps}\} + \{\text{Result} [N] = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the ASIN and ASINP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
	Date	.YR/ .MO/ .DAY only		—	X
Time	.HR/ .MIN/ .SEC only		—	X	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/.CV only	—	X
	N_	.YR/.MO/.DAY only	—	X
	J_	.HR/.MIN/.SEC only	—	X
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	X	
Constant (Cannot use for D1)	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O

◆ **Explanation of the ASIN and ASINP Instructions**

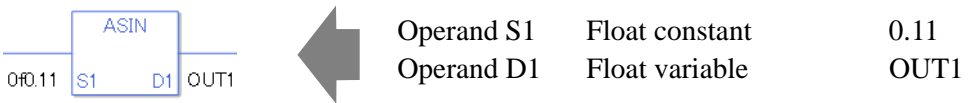
The ASIN and ASINP instructions are arc sine instructions for trigonometric functions. The ASIN instruction calculates the arc sine of S1 and stores the result in D1. $\text{Sin}^{-1}(S1)$ is stored in D1. Enter a value between -1.0 and 1.0 for S1. The result in D1 is shown in radians as a real value between $-\pi/2$ and $\pi/2$. π is approximately 3.1415926535897 (real number).

The ASIN and ASINP instructions are always conducted. When using the ASIN and ASINP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

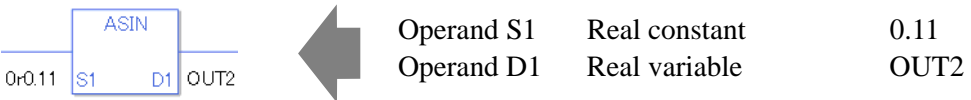
When operand D1 is a float variable

When Of (zero and lower case "f") is input, the following values become float values.



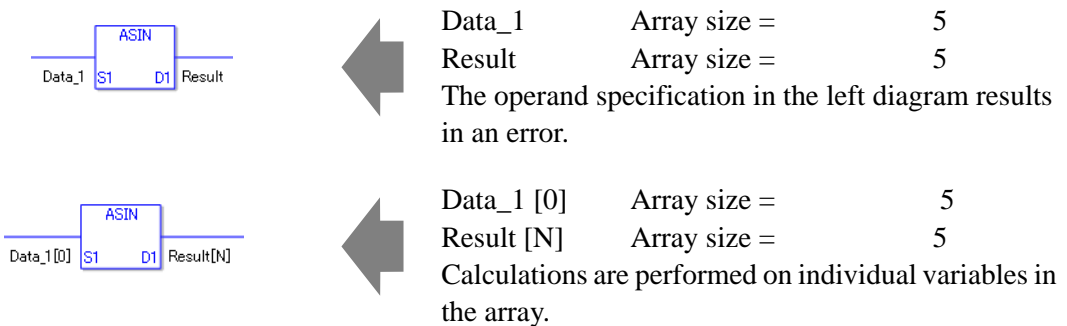
When operand D1 is a real variable

When Or (zero and lower case "r") is input, the following values become real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

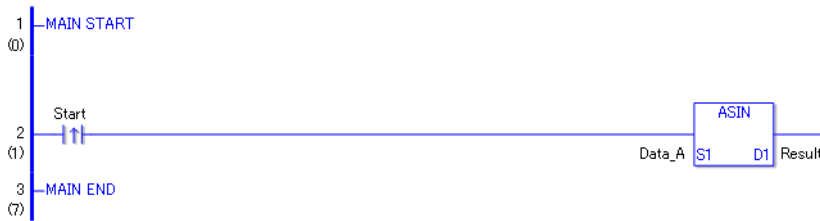
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

ASIN



- (1) The ASIN instruction is executed when the positive transition instruction turns ON. The ASIN instruction calculates the arc sine of Data_A and stores the result in D1. When using a normally open instruction, the ASIN instruction is always executed as long as the normally open instruction bit is ON.

Program Example



ASINP



- (1) ASINP and ASIN instructions differ in when they run. In ASINP instructions, even when using a normally open instruction, only the positive transition is detected and the ASINP instruction is executed. Therefore, the ASINP instruction is executed only for one scan, even when the normally open instruction bit remains ON.

■ **ACOS and ACOSP (Arc Cosine)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ACOS (Arc Cosine - Level Sensitive)		Trigonometric Function	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ACOSP (Arc Cosine - positive transition)		Trigonometric Function	3 to 7

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the ACOS and ACOSP instructions.

The actual number of steps in the ACOS and ACOSP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in ACOS and ACOSP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{\text{Data}_1 [0] = 2 \text{ steps}\} + \{\text{Result} [N] = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the ACOS and ACOSP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
Specify real variable [variable]		3	O		

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant (Cannot use for D1)	Integer	- 2147483648 to 2147483647	—	X
	Float	±1.175494351e-38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	O

◆ **Explanation of the ACOS and ACOSP Instructions**

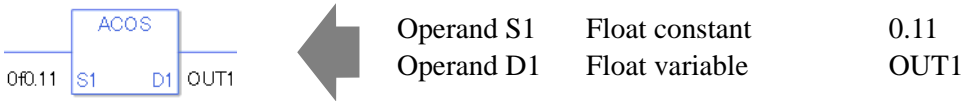
ACOS and ACOSP instructions are arc cosine instructions for trigonometric functions. When the ACOS instruction is executed and passes power, the S1 value is ACOSed and the result [COS⁻¹(S1)] is stored in D1. Input values between -1.0 to 1.0 for S1, and the result in D1 is a real number, measured in radians, between 0 and Pi. Pi is approximately 3.1415926535897 (real number).

The ACOS and ACOSP instructions are always conducted. When using the ACOS and ACOSP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

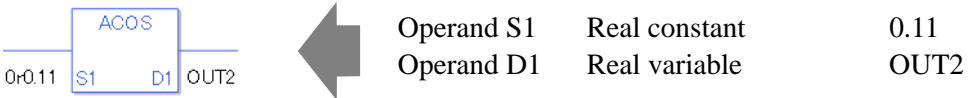
When operand D1 is a float variable

When Of (zero and lower case "f") is input, the following values become float values.



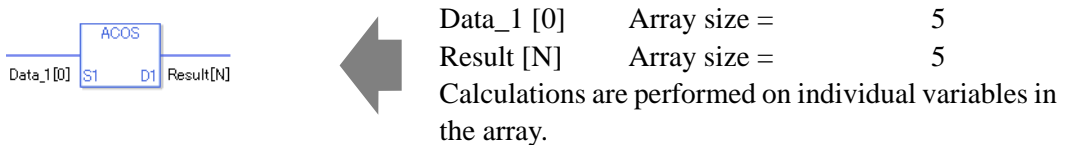
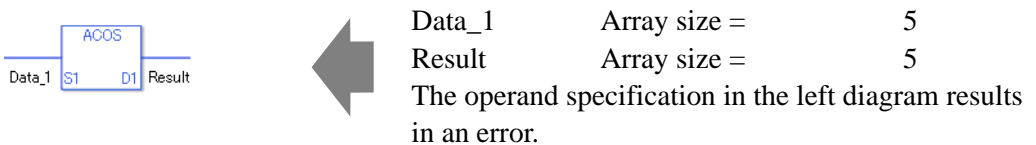
When operand D1 is a real variable

When Or (zero and lower case "r") is input, the following values become real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

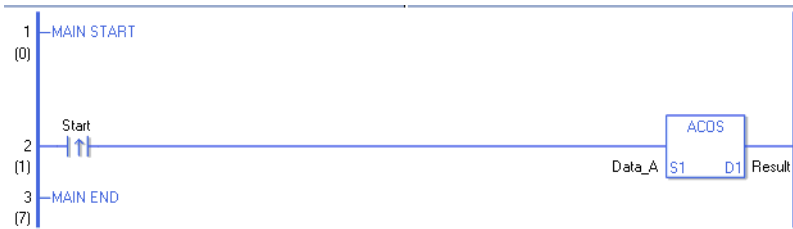
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

ACOS



(1) The ACOS instruction is executed when the positive transition instruction turns ON. The ACOS instruction calculates the arc cosine of Data_A and stores the result in D1.

When using a normally open instruction, the ACOS instruction is always executed as long as the normally open instruction bit ON.

Program Example



ACOSP



(1) ACOSP and ACOS instructions differ in when they run. In ACOSP instructions, even when using a normally open instruction, the ACOSP instruction executes only when it detects the upward transition. Therefore, the ACOSP instruction is executed only for one scan, even when the normally open instruction bit remains ON.

■ **ATAN and ATANP (Arc Tangent)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ATAN (Arc Tangent - Level Sensitive)		Trigonometric Function	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ATANP (Arc Tangent - positive transition)		Trigonometric Function	3 to 7

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the ATAN and ATANP instructions.

The actual number of steps in the ATAN and ATANP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in ATAN and ATANP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{ \text{Data}_1 [0] = 2 \text{ steps} \} + \{ \text{Result} [N] = 3 \text{ steps} \} + \{ 1 \text{ step} \} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the ATAN and ATANP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
Specify real variable [variable]		3	O		

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant (Cannot use for D1)	Integer	- 2147483648 to 2147483647	—	X
	Float	±1.175494351e-38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	O

◆ **Explanation of the ATAN and ATANP Instructions**

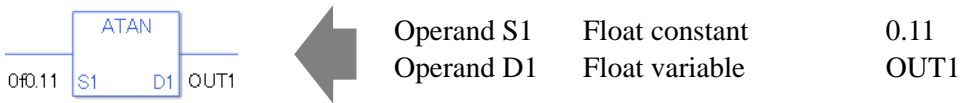
ATAN and ATANP instructions are arc tangent instructions for trigonometric functions. When the ATAN instruction is executed and passes power, the S1 value is ATANed and the result [TAN⁻¹(S1)] is stored in D1. Input values between -1.0 and 1.0 for S1, and the result in D1 is a real number, measured in radians, between -Pi/2 to Pi/2. Pi is approximately 3.1415926535897 (real number).

The ATAN and ATANP instructions are always conducted. When using the ATAN and ATANP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

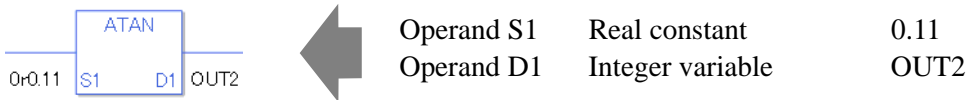
When operand D1 is a float variable

When Of (zero and lower case "f") is input, the following values become float values.



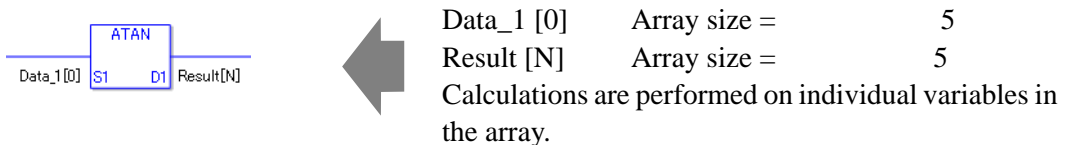
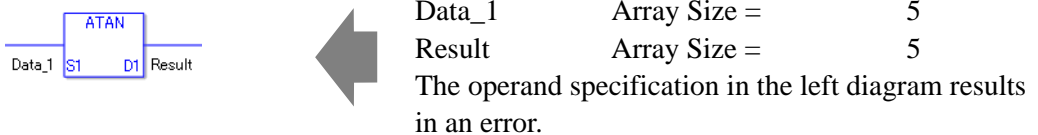
When operand D1 is a real variable

When Or (zero and lower case "r") is input, the following values become real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

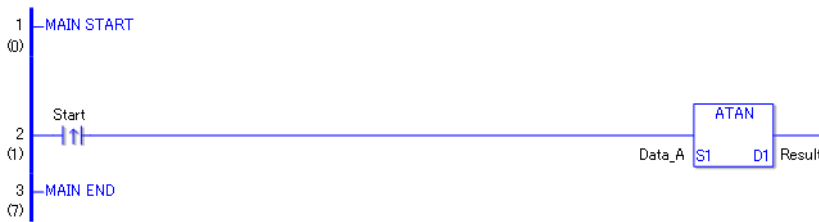
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

ATAN



- (1) The ATAN instruction will be executed when the positive transition instruction turns ON. The ATAN instruction calculates the arc tangent of Data_A and stores the result in D1. When using a normally open instruction, the ATAN instruction is always executed as long as the normally open instruction bit remains ON.

Program Example



ATANP



- (1) ATANP and ATAN instructions differ in when they run. In ATANP instructions, even when using a normally open instruction, the ATANP instruction executes only when it detects the upward transition. Therefore, the ATANP instruction is executed only for one scan, even when the normally open instruction bit remains ON.

■ **COT and COTP (Cotangent)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
COT (Cotangent - Level Sensitive)		Trigonometric Function	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
COTP (Cotangent - positive transition)		Trigonometric Function	3 to 7

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the COT and COTP instructions.

The actual number of steps in the COT and COTP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in COT and COTP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{ \text{Data}_1 [0] = 2 \text{ steps} \} + \{ \text{Result} [N] = 3 \text{ steps} \} + \{ 1 \text{ step} \} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the COT and COTP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Variable Format	Timer	.PT/.ET only	—	X	
	Counter	.PV/ .CV only	—	X	
	Date	.YR/ .MO/ .DAY only	—	X	
	Time	.HR/ .MIN/ .SEC only	—	X	
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	1	O	
	R_	—	1	O	
	T_	.PT/.ET only	—	X	
	C_	.PV/ .CV only	—	X	
	N_	.YR/ .MO/ .DAY only	—	X	
	J_	.HR/ .MIN/ .SEC only	—	X	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X		
Constant (Cannot use for D1)	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O	

◆ **Explanation of the COT and COTP Instructions**

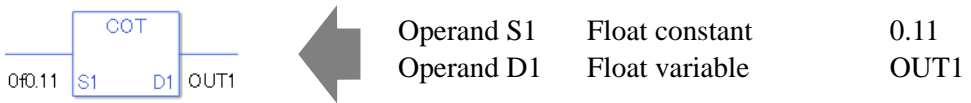
COT and COTP instructions are cotangent instructions for trigonometric functions. When the COT instruction is executed and passes power, the S1 value is COTed and the result [1/tan (S1)] is stored in D1. Input the number of radians in S1. The closer S1 is to a multiple of Pi results in a larger absolute value in D1, which can be expressed as a real number with a range of +/-2.225e-308 to +/-1.79e+308.

Pi is approximately 3.1415926535897 (real number).COT and COTP instructions always pass power. When using COT and COTP instructions, specify the same data type in operands S1 and D1 to avoid errors.

Refer to the following for specifying a constant.

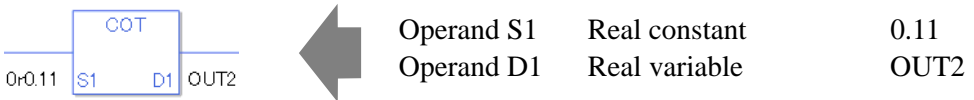
When operand D1 is a float variable

When Of (zero and lower case "f") is input, the following values become float values.



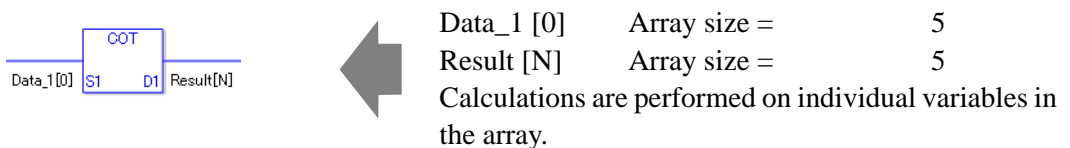
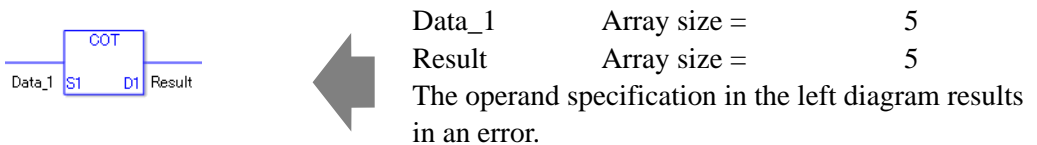
When operand D1 is a real variable

When Or (zero and lower case "r") is input, the following values become real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

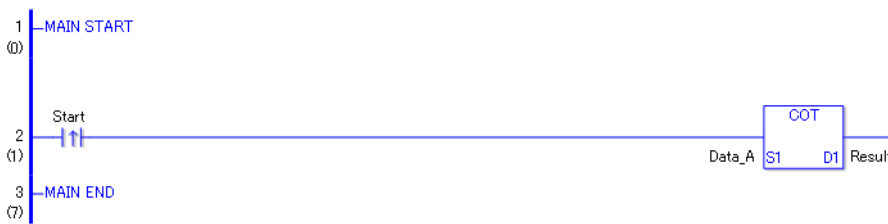
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

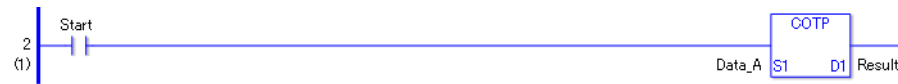
COT



- (1) The COT instruction is executed when the positive transition instruction turns ON. The COT instruction calculates the cotangent of Data_A and stores the result in D1. When using a normally open instruction, the COT instruction is always executed as long as the normally open instruction bit is ON.

Program Example



COTP



- (1) COTP and COT instructions differ in when they run. In COTP instructions, even when using a normally open instruction, the COTP instruction executes only when it detects the upward transition. Therefore, the COTP instruction is executed only for one scan, even when the normally open instruction bit remains ON.

■ **EXP and EXPP (Exponential)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
EXP (Exponent - Level Sensitive)		Other Function	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
EXPP (Exponent - positive transition)		Other Function	3 to 7

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the EXP and EXPP instructions.

The actual number of steps in the EXP and EXPP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in EXP and EXPP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 [0] = 2 \text{ steps}\} + \{Result [N] = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the EXP and EXPP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
Specify real variable [variable]		3	O		

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Variable Format	Timer	.PT/.ET only	—	X	
	Counter	.PV/.CV only	—	X	
	Date	.YR/.MO/.DAY only	—	X	
	Time	.HR/.MIN/.SEC only	—	X	
	PID	.KP/.TR/.TD/.PA/.BA/.ST only	—	X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	1	O	
	R_	—	1	O	
	T_	.PT/.ET only	—	X	
	C_	.PV/.CV only	—	X	
	N_	.YR/.MO/.DAY only	—	X	
	J_	.HR/.MIN/.SEC only	—	X	
	U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	X	
Constant (Cannot use for D1)	Integer	- 2147483648 to 2147483647	—	X	
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	O	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	O	

◆ **Explanation of EXP and EXPP Instructions**

The EXP and EXPP instructions are exponential instructions. The EXP instruction calculates the exponent of S1 and stores the result in D1.

The exponent of S1 is stored in D1. e to the power of S1 is output as a real value to D1.

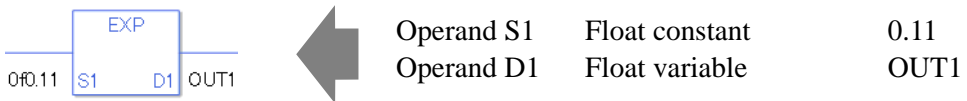
Operation expression: $D1 = e^{S1}$ e is approximately 2.7182818284590 (real number).

The EXP and EXPP instructions are always conducted. When using the EXP and EXPP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

When operand D1 is a float variable

When 0f (zero and lower case "f") is input, the following values become float values.



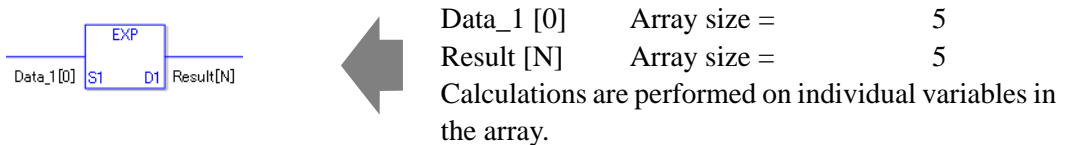
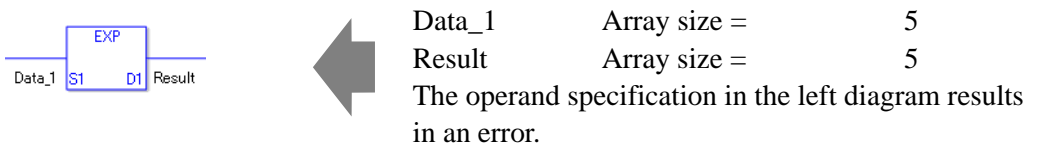
When operand D1 is a real variable

When 0r (zero and lower case "r") is input, the following values become real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

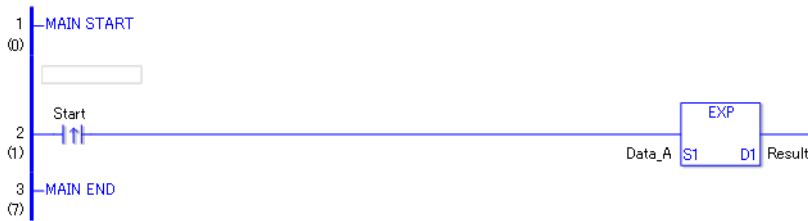
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

EXP



- (1) The EXP instruction is executed when the positive transition instruction turns ON. The EXP instruction calculates the exponent of Data_A and stores the result in D1. When using a normally open instruction, the EXP instruction is always executed as long as the normally open instruction bit ON.

Program Example



EXPP



- (1) The EXPP and EXP instructions differ in when they run. In EXPP instructions, even when using a normally open instruction, the EXPP instruction executes only when it detects the upward transition. Therefore, the EXPP instruction is executed only for one scan, even when the normally open instruction bit remains ON.

■ **LN and LNP (Logarithm)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
LN (Logarithm - Level Sensitive)		Other Function	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
LNP (Logarithm - positive transition)		Other Function	3 to 7

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the LN and LNP instructions.

The actual number of steps in the LN and LNP instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in LN and LNP instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{ \text{Data}_1 [0] = 2 \text{ steps} \} + \{ \text{Result} [N] = 3 \text{ steps} \} + \{ 1 \text{ step} \} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the LN and LNP instructions

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
Specify real variable [variable]		3	O		

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant (Cannot use for D1)	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O

◆ **Explanation of the LN and LNP Instructions**

The LN and LNP instructions are logarithmic instructions. The LN instruction calculates the natural logarithmic function of S1 and stores the result in D1. The result in D1 is output as a real value where e raised to the power of D1 equals S1.

Operation expression: $D1 = \log_e S1$ e is approximately 2.7182818284590 (real number).

The LN and LNP instructions are always conducted. When using the LN and LNP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

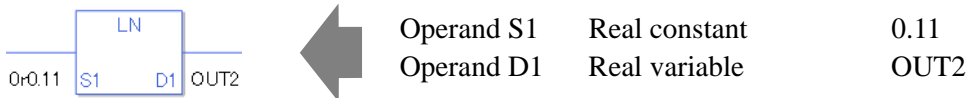
When operand D1 is a float variable

When Of (zero and lower case "f") is input, the following values become float values.



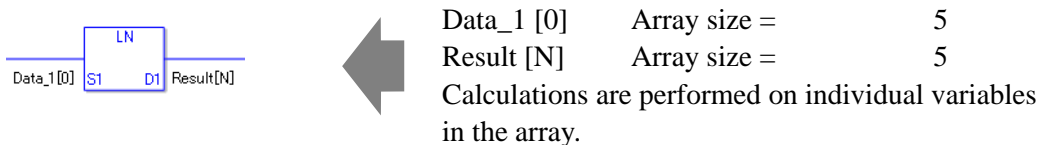
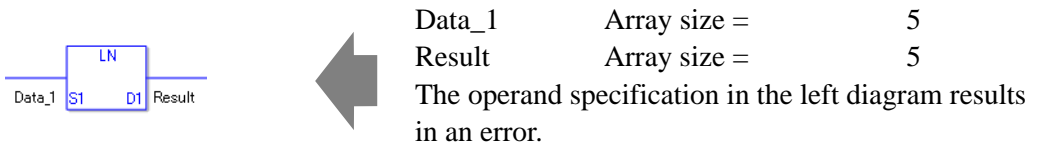
When operand D1 is a real variable

When Or (zero and lower case "r") is input, the following values become real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

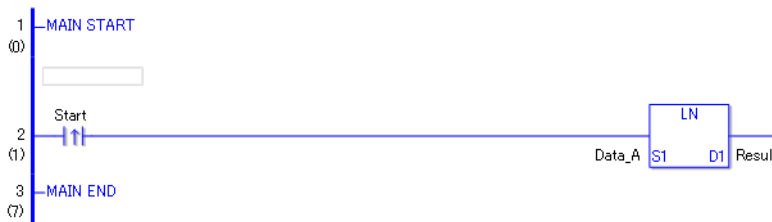
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

LN



(1) The LN instruction is executed when the positive transition instruction turns ON. The LN instruction calculates the natural logarithmic function of Data_A and stores the result in D1.

When using a normally open instruction, the LN instruction is always executed as long as the normally open instruction bit is ON.

Program Example



LNP



(1) The LNP and LN instructions differ in when they run. In the LNP instructions, even when using a normally open instruction, the LNP instruction executes only when it detects the upward transition. Therefore, the LNP instruction is executed only for one scan, even when the normally open instruction bit remains ON.

■ **LG10 and LG10P (Log Base 10)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
LG10 (Log Base 10 - Level Sensitive)		Other Function	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
LG10P (Log Base 10 - positive transition)		Other Function	3 to 7

◆ **Operand Settings**

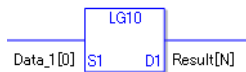
The following describes the specifiable content of operands S1 and D1 for the LG10 and LG10P instructions.

The actual number of steps in the LG10 and LG10P instructions depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in LG10 and LG10P instructions

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{ \text{Data}_1 [0] = 2 \text{ steps} \} + \{ \text{Result} [N] = 3 \text{ steps} \} + \{ 1 \text{ step} \} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and D1 for the LG10 and LG10P instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
Specify real variable [variable]		3	O		

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant (Cannot use for D1)	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O

◆ **Explanation of LG10 and LG10P Instructions**

The LG10 and LG10P instructions are logarithmic instructions. The LG10 instruction calculates the common logarithm function of S1 and stores the result in D1.

For the result in D1, the result of $\log_{10} S1$ is output as a real value.

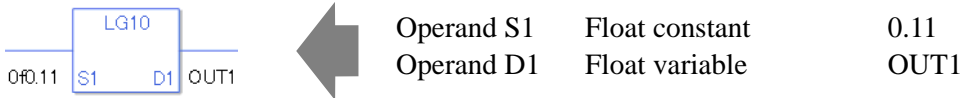
Equation: $D1 = \log_{10} S1$

The LG10 and LG10P instructions are always conducted. When using the LG10 and LG10P instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

When operand D1 is a float variable

When Of (zero and lower case "f") is input, the following values become float values.



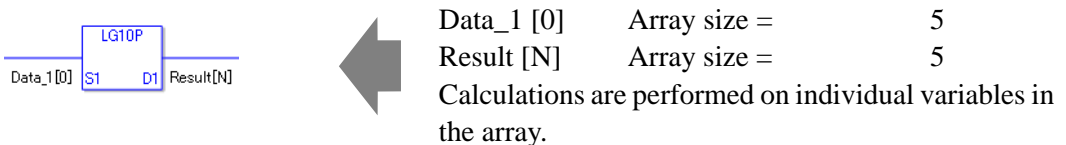
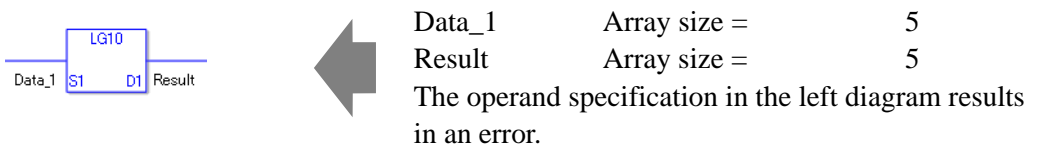
When operand D1 is a real variable

When Or (zero and lower case "r") is input, the following values become real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

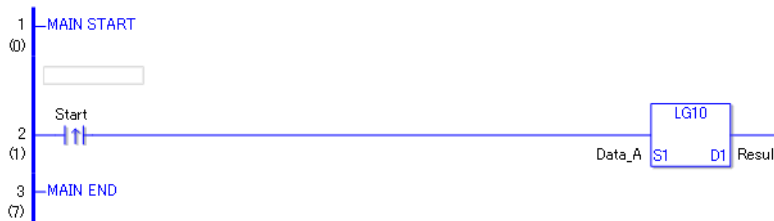
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

LG10



- (1) The LG10 instruction is executed when the positive transition instruction turns ON. The LG10 instruction calculates the common logarithm function of Data_A and stores the result in D1. When using a normally open instruction, the LG10 instruction is always executed as long as normally open instruction bit is ON.

Program Example

LG10P

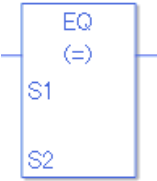


- (1) The LG10P and LG10 instructions differ in when they run. In the LG10P instructions, even when using a normally open instruction, the LG10P instruction executes only when it detects the upward transition. Therefore, the LG10P instruction is executed only for one scan, even when the normally open instruction bit remains ON.

29.5.15 Compare Instruction (Arithmetic)

■ EQ (=)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
EQ (Equal To - Level Sensitive)		Comparison	3 to 9

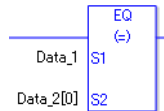
◆ Operand Settings

The following describes the specifiable content of operands S1 and S2 for the EQ instruction. The actual number of steps in the EQ instruction depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in EQ instruction

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{Data_2 [0] = 2 \text{ steps}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and S2 for the EQ instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O
	Float	—	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O
	Real	—	1	O
		Specify real variable [constant]	2	O
Specify real variable [variable]		3	O	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	- 2147483648 to 2147483647	1	O
	Float	±1.175494351e-38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	O

◆ **Explanation of the EQ Instruction**

The EQ instruction is a compare instruction. The EQ instruction compares S1 with S2 and if the result of the comparison is $S1 = S2$, the instruction passes power. Be careful when comparing real values. For example, if the operand value is 1.9999999999, it is not equal to 2.0000000000.

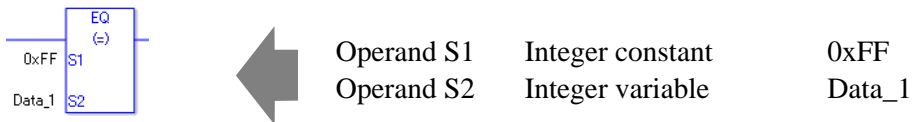
When using the EQ instruction, an error will occur if the variables specified in operands S1 and S2 are not the same type. Specify the same variable type in operands S1 and S2. Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



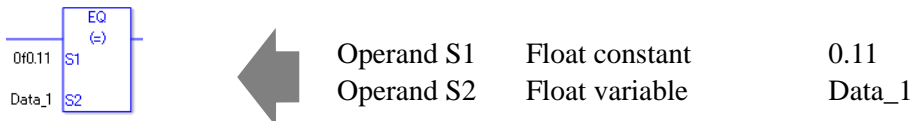
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case "x") is input, the following values become hexadecimal values.



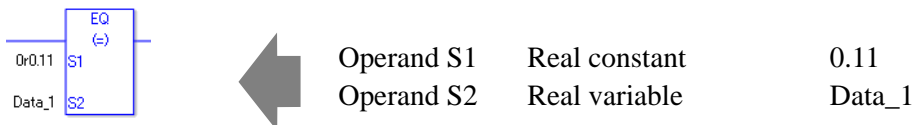
When entering float constants in operands S1 and S2

When 0f (zero and lower case "f") is input, the following values become float values.



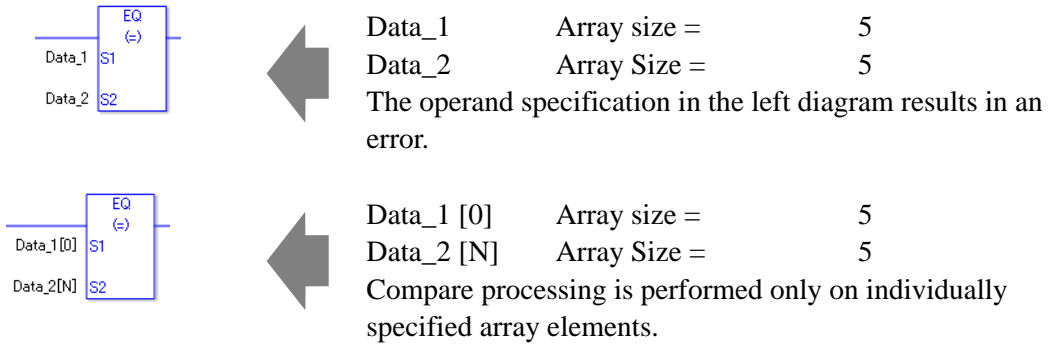
When entering real constants in operands S1 and S2

When 0r (zero and lower case "r") is input, the following values become real values.



When comparing data in a specified array (integer variable array) Specify the array using Data[0] or Data[N] (N indicates an integer variable).

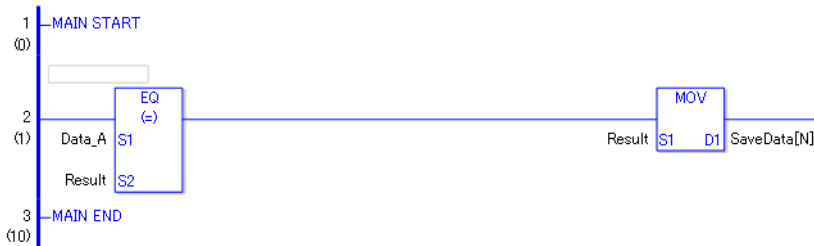
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

EQ

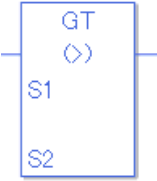
Compares integer variables and outputs the result in D1.



(1)Data_A and the operation result are compared to determine whether they are equal. If the result of the EQ instruction is S1 = S2, the EQ instruction passes power, then the instruction to the right of the EQ instruction is executed. In the above diagram, it's the MOV instruction.

■ **GT (>)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
GT (Greater Than - Level Sensitive)		Comparison	3 to 9

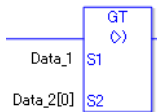
◆ **Operand Settings**

The following describes the specifiable content of operands S1 and S2 for the GT instruction. The actual number of steps in the GT instruction depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in GT instruction

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{Data_2 [0] = 2 \text{ steps}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and S2 for the GT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (including I/O)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]		2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]		3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]		4	O
		Float Variable		1	O
	Float	Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
		Real Variable		1	O
	Real	Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O

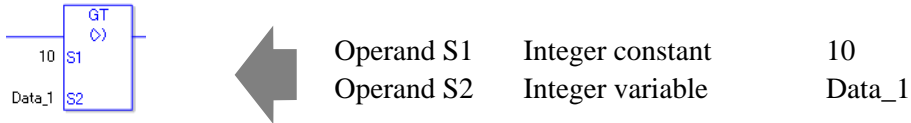
Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Variable Format	Timer	.PT/.ET only	2	O	
	Counter	.PV/.CV only	2	O	
	Date	.YR/.MO/.DAY only	2	O	
	Time	.HR/.MIN/.SEC only	2	O	
	PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	O	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	1	O	
	Q_	—	1	O	
	D_	Modifiers are not specified		1	O
		D_****.B/W[constant]		2	O
		D_****.B/W[address]		3	O
	F_	—	1	O	
	R_	—	1	O	
	T_	.PT/.ET only	2	O	
	C_	.PV/.CV only	2	O	
	N_	.YR/.MO/.DAY only	2	O	
	J_	.HR/.MIN/.SEC only	2	O	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	O		
Constant	Integer	– 2147483648 to 2147483647	1	O	
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O	

◆ **Explanation of the GT Instruction**

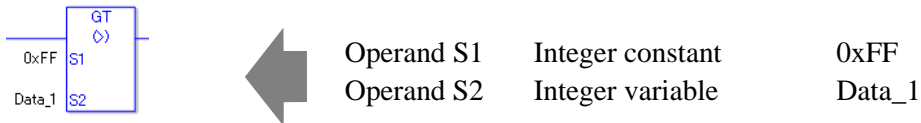
The GT instruction is a compare instruction. The GT instruction compares S1 with S2. If the result of the comparison is $S1 > S2$, the instruction passes power. Be careful when comparing real values. For example, if the operand value is 2.000000000001, it is still greater than 2. When using the GT instruction, an error will occur if the variables specified in operands S1 and S2 are not the same type. Specify the same variable type in operands S1 and S2. Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



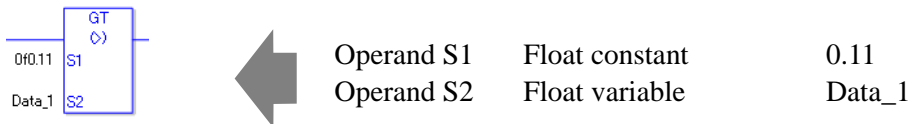
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case "x") is input, the following values become hexadecimal values.



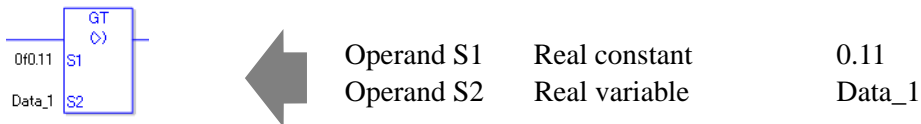
When entering float constants in operands S1 and S2

When 0f (zero and lower case "f") is input, the following values become float values.



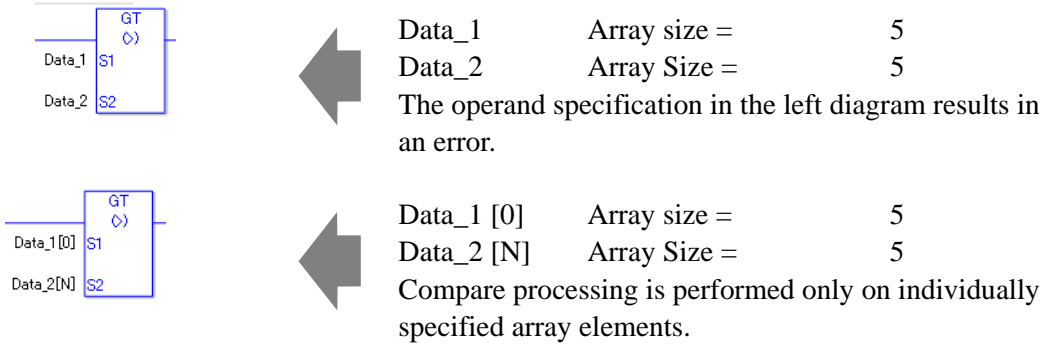
When entering real constants in operands S1 and S2

When 0r (zero and lower case "r") is input, the following values become real values.



When comparing data in a specified array (integer variable array) Specify the array using Data[0] or Data[N] (N indicates an integer variable).

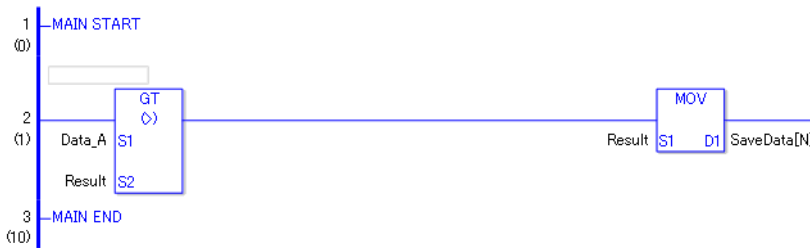
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

GT

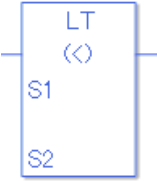
Compares integer variables and outputs the result in D1.



(1)Data_A and the operation result are compared to determine whether Data_A is greater than the operation result. If the result of the GT instruction is S1 > S2, the GT instruction passes power. Then the instruction to the right of the GT instruction is executed. In the above diagram, it's the MOV instruction.

■ **LT (<)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
LT (Less Than - Level Sensitive)		Comparison	3 to 9

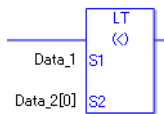
◆ **Operand Settings**

The following describes the specifiable content of operands S1 and S2 for the LT instruction. The actual number of steps in the LT instruction depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in LT instruction

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{Data_2 [0] = 2 \text{ steps}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and S2 for the LT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (including I/O)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant] or Specify integer variable B/ W[constant]		2	O
		Specify integer variable[variable] or Specify integer variable B/ W[variable]		3	O
		Specify integer variable[constant/ variable] .B/W[constant/variable]		4	O
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
Specify real variable [variable]		3	O		

Continued

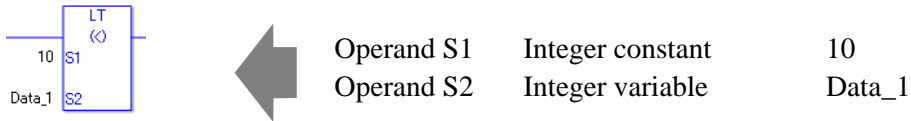
Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	- 2147483648 to 2147483647	1	O
	Float	±1.175494351e-38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	O

◆ **Explanation of the LT Instruction**

The LT instruction is a compare instruction. The LT instruction compares S1 with S2. If the result of the comparison is $S1 < S2$, the instruction passes power. Be careful when comparing real values. For example, if the operand value is 1.9999999999, it is less than 2. When using the LT instruction, an error will occur if the variables specified in operands S1 and S2 are not the same type. Specify the same variable type in operands S1 and S2.

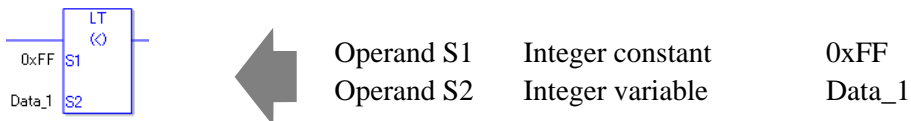
Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



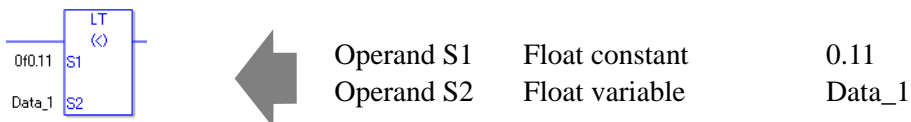
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case "x") is input, the following values become hexadecimal values.



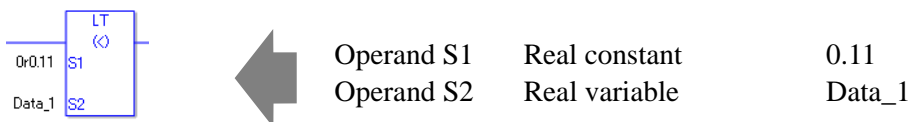
When entering float constants in operands S1 and S2

When 0f (zero and lower case "f") is input, the following values become float values.



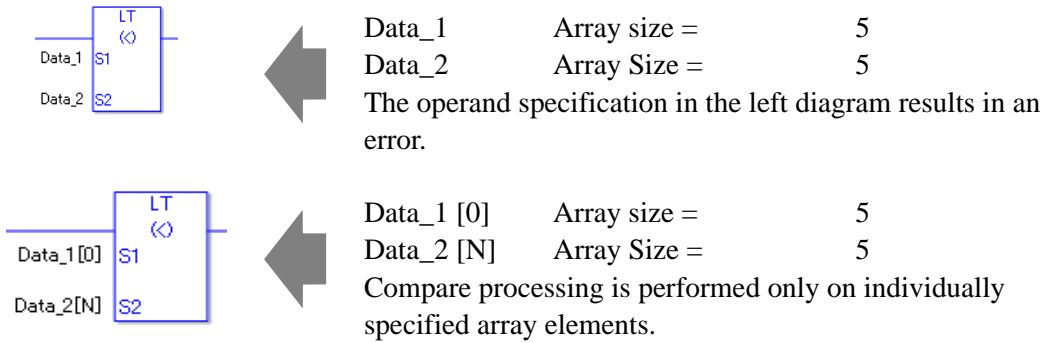
When entering real constants in operands S1 and S2

When 0r (zero and lower case "r") is input, the following values become real values.



When comparing data in a specified array (integer variable array) Specify the array using Data[0] or Data[N] (N indicates an integer variable).

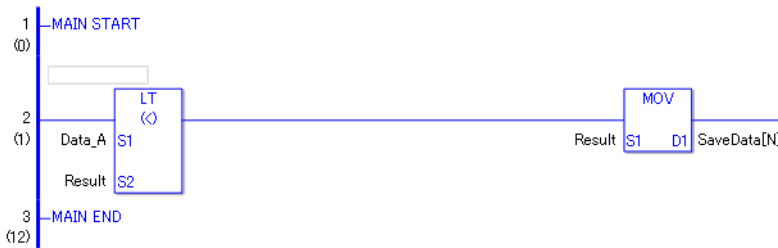
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

LT

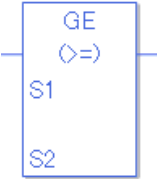
Compares integer variables and outputs the result in D1.



(1)Data_A and the operation result are compared to determine whether Data_A is less than the operation result. If the result of the LT instruction is S1 < S2, the LT instruction passes power. Then the instruction to the right of the LT instruction is executed. In the above diagram, it's the MOV instruction.

■ **GE (>=)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
GE (Greater Than or Equal To - Level Sensitive)		Comparison	3 to 9

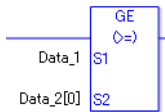
◆ **Operand Settings**

The following describes the specifiable content of operands S1 and S2 for the GE instruction. The actual number of steps in the GE instruction depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in GE instruction

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{Data_2 [0] = 2 \text{ steps}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and S2 for the GE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (including I/O)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]		2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]		3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]		4	O
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	- 2147483648 to 2147483647	1	O
	Float	±1.175494351e-38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	O

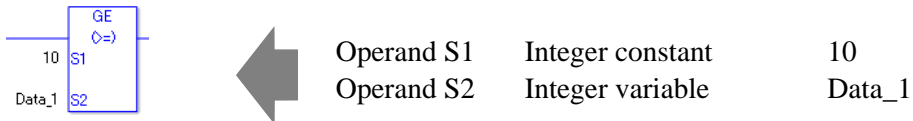
◆ **Explanation of the GE Instruction**

The GE instruction is a compare instruction. The GE instruction compares S1 with S2. If the result of the comparison is $S1 \geq S2$, the instruction passes power.

Be careful when comparing real values. For example, if the operand value is 1.99999999999, it is not greater than 2. When using the GE instruction, an error will occur if the variables specified in operands S1 and S2 are not the same type. Specify the same variable type in operands S1 and S2.

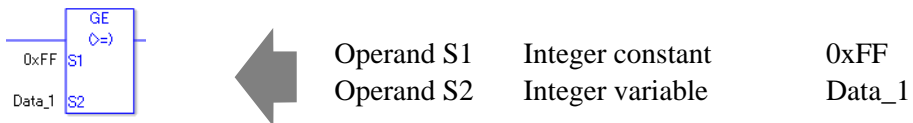
Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



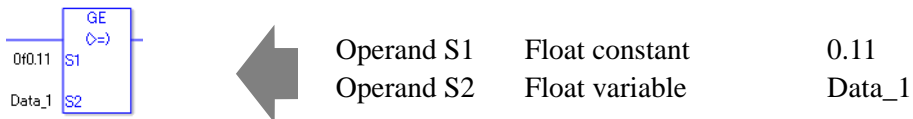
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case "x") is input, the following values become hexadecimal values.



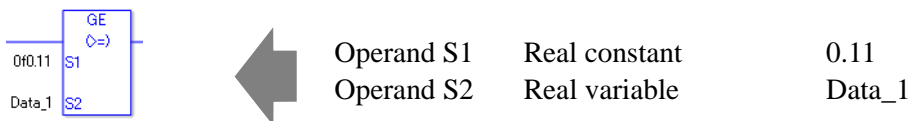
When entering float constants in operands S1 and S2

When 0f (zero and lower case "f") is input, the following values become float values.



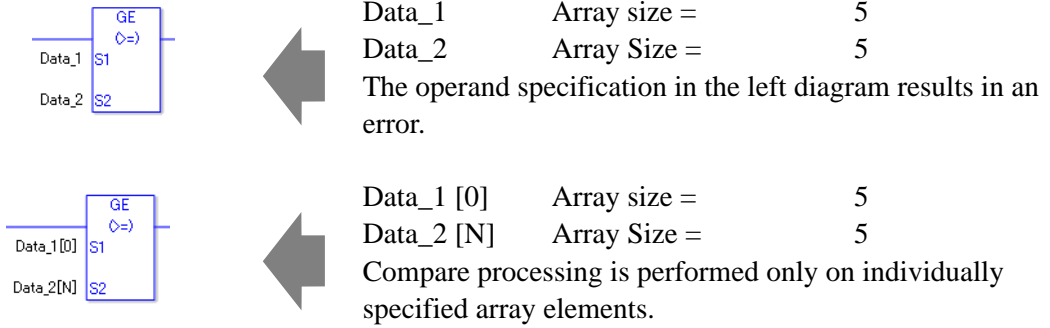
When entering real constants in operands S1 and S2

When 0r (zero and lower case "r") is input, the following values become real values.



When comparing data in a specified array (integer variable array) Specify the array using Data[0] or Data[N] (N indicates an integer variable).

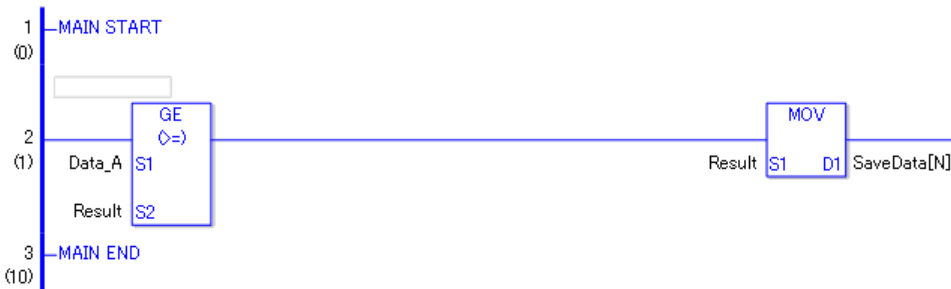
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

GE

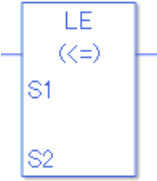
Compares integer variables and outputs the result in D1.



(1) Data_A and the operation result are compared to determine whether Data_A is greater than or equal the operation result. If the result of the GE instruction is S1 >= S2, the GE instruction passes power. Then the instruction to the right of the GE instruction is executed. In the above diagram, it's the MOV instruction.

■ **LE (<=)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
LE (Less Than or Equal To - Level Sensitive)		Comparison	3 to 9

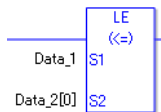
◆ **Operand Settings**

The following describes the specifiable content of operands S1 and S2 for the LE instruction. The actual number of steps in the LE instruction depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in LE instruction

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{Data_2 [0] = 2 \text{ steps}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of the S1 and S2 operands for the LE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O
	Float	Float Variable	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O
	Real	Real Variable	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O

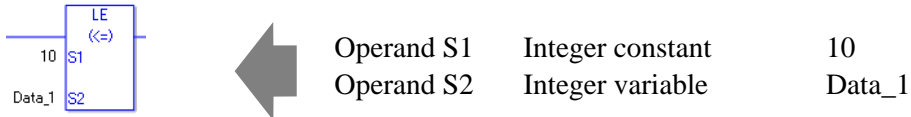
◆ **Explanation of LE Instructions**

The LE instruction is a compare instruction. The LE instruction compares S1 with S2. If the result of the comparison is $S1 \leq S2$, the instruction passes power.

Be careful when comparing real values. For example, if the operand is 2.000000000001, it is not less than or equal to 2. When using the LE instruction, an error will occur if the variables specified in operands S1 and S2 are not the same type. Specify the same variable type in operands S1 and S2.

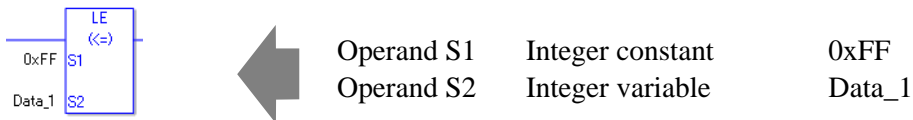
Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



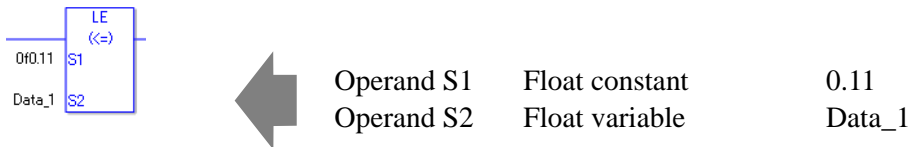
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case "x") is input, the following values become hexadecimal values.



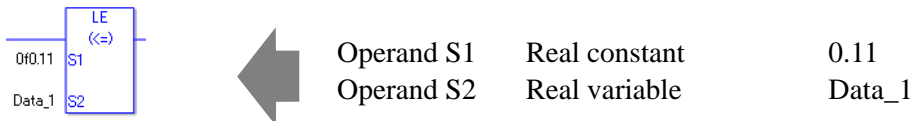
When entering float constants in operands S1 and S2

When 0f (zero and lower case "f") is input, the following values become float values.



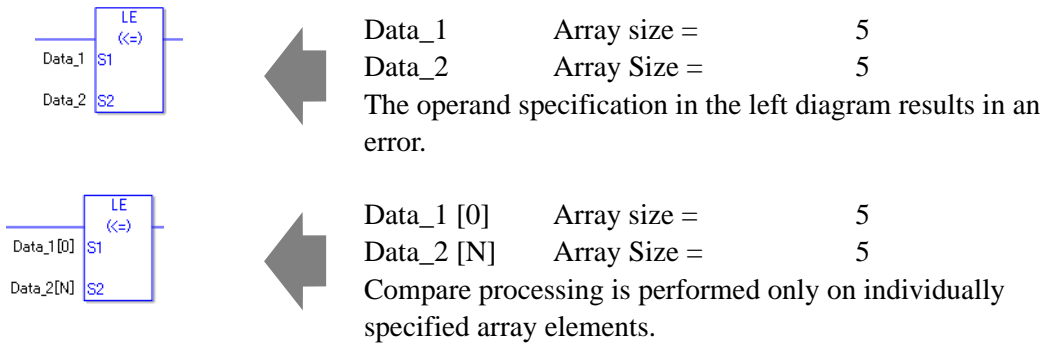
When entering real constants in operands S1 and S2

When 0r (zero and lower case "r") is input, the following values become real values.



When comparing data in a specified array (integer variable array) Specify the array using Data[0] or Data[N] (N indicates an integer variable).

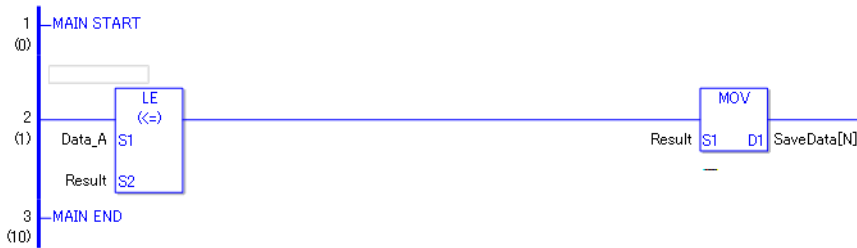
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

LE

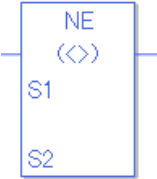
Compares integer variables and outputs the result in D1.



(1)Data_A and the operation result are compared to determine whether Data_A is less than or equal to the operation result. If the result of the LE instruction is S1 <= S2, the LE instruction passes power. Then the instruction to the right of the LE instruction is executed. In the above diagram, it's the MOV instruction.

■ **NE (<>)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NE (Not Equal - Level Sensitive)		Comparison	3 to 9

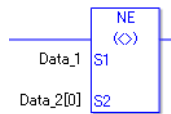
◆ **Operand Settings**

The following describes the specifiable content of operands S1 and S2 for the NE instruction. The actual number of steps in the NE instruction depends on the specified operand. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Converting the number of steps in NE instruction

(For the number of steps in an operand, refer to the operand settings on the next page.)



$$\{Data_1 = 1 \text{ step}\} + \{Data_2 [0] = 2 \text{ steps}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of operands S1 and S2 for the NE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] or Specify integer variable B/W[constant]	2	O
		Specify integer variable[variable] or Specify integer variable B/W[variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	4	O
	Float	Float Variable	1	O
		Specify float variable[constant]	2	O
		Specify float variable[variable]	3	O

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	Real Variable	1	O
		Specify real variable [constant]	2	O
		Specify real variable [variable]	3	O
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	2	O
		D_****.B/W[address]	3	O
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	- 2147483648 to 2147483647	1	O
	Float	±1.175494351e-38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	O

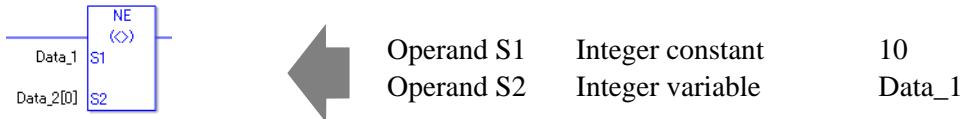
◆ **Explanation of NE Instruction**

The NE instruction is a compare instruction. The NE instruction compares S1 with S2. If the result of the comparison is $S1 \lt S2$, the instruction passes power.

Be careful when comparing real values. For example, if the operand value is 2.000000000001, it is not equal to 2. When using the NE instruction, an error will occur if the variables specified in operands S1 and S2 are not the same type. Specify the same variable type in operands S1 and S2.

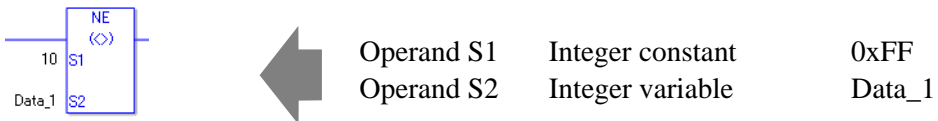
Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



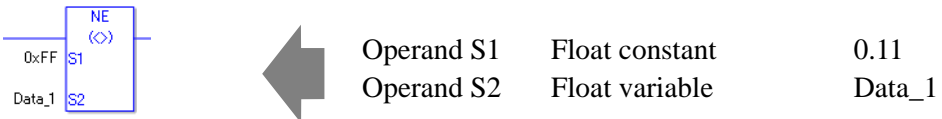
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case "x") is input, the following values become hexadecimal values.



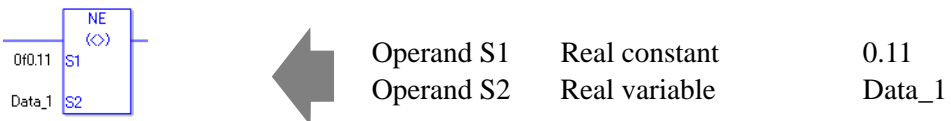
When entering float constants in operands S1 and S2

When 0f (zero and lower case "f") is input, the following values become float values.



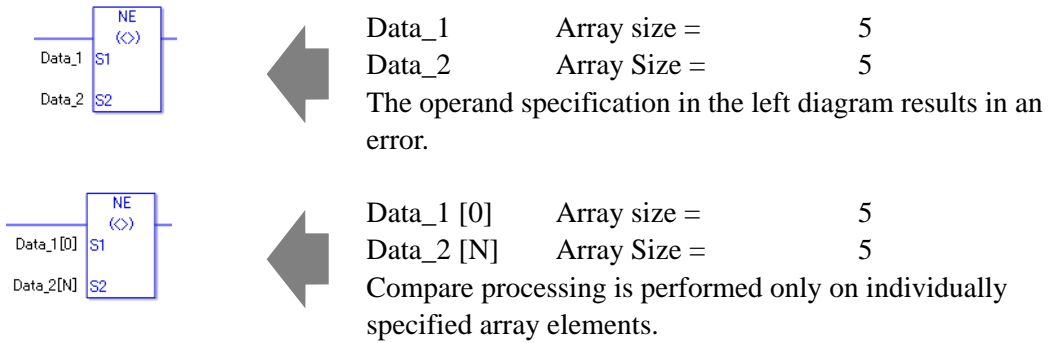
When entering real constants in operands S1 and S2

When 0r (zero and lower case "r") is input, the following values become real values.



When comparing data in a specified array (integer variable array) Specify the array using Data[0] or Data[N] (N indicates an integer variable).

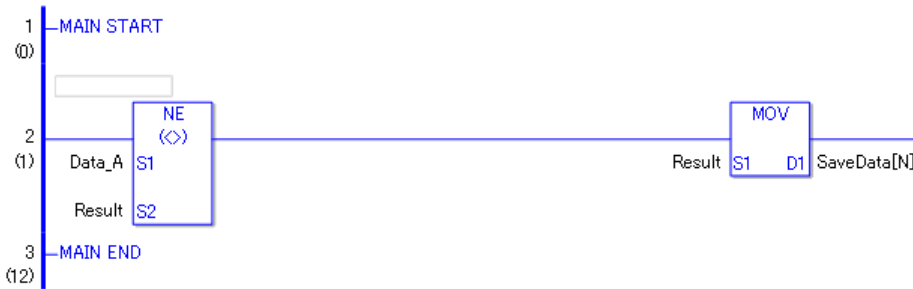
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

NE

Compares integer variables and outputs the result in D1.

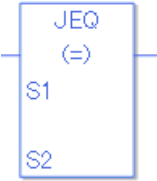


(1) Data_A and the operation result are compared to determine whether Data_A is not equal to the operation result. If the result of the NE instruction is S1 <> S2, the NE instruction passes power. Then the instruction to the right of the NE instruction is executed. In the above diagram, it's the MOV instruction.

29.5.16 Compare (Time)

■ JEQ (Equal)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JEQ (= Level Sensitive)		Time Compare	3

◆ Operand Settings

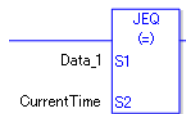
The following shows the configurable conditions for Operands (S1, S2) in the JEQ instruction.

The actual number of steps in the JEQ instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the JEQ instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{\text{Current time} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ Explanation of the JEQ Instruction

Time variables in JEQ instructions are structure variables. The following table lists the internal structures.

Time Variable

Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD.
VariableName.MIN	Integer Variable	Minutes are input in BCD.
VariableName.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the JEQ instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]		—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]		—	X
		Specify integer variable[constant/variable].B/W[constant/variable]		—	X
	Float	Float Variable		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	Real Variable	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC Structure elements are not specified.	1	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC Structure elements are not specified.	1	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the JEQ Instruction**

The JEQ instruction compares time. When the JEQ instruction is executed, S1 is compared to S2. The instruction passes power if the result is $S1 = S2$.

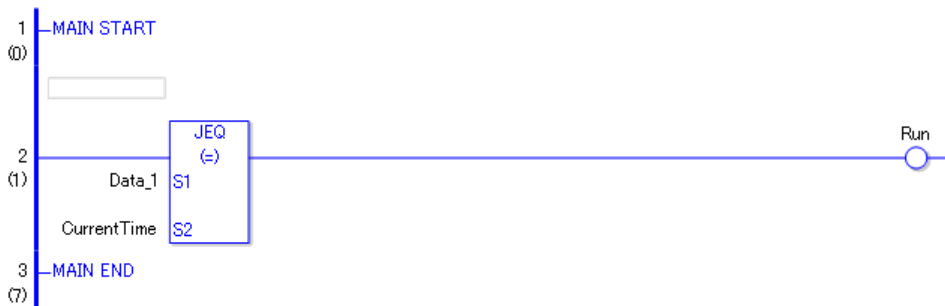
The hour, minute, and second variables are compared simultaneously. To compare a time of 10:20, input 0 for the seconds.

When using JEQ instructions, the only variables you can specify in operands S1 and S2 are time variables.

Program Example

JEQ

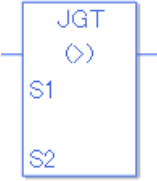
Compares the time variables and determines the result with the coil.



- (1) Compares Data_1 to the current time to determine whether they are equal. If the result is $S1 = S2$, the instruction passes power and an instruction to the right of the JEQ instruction is executed. In the above chart, the OUT instruction to the right of the JEQ instruction is executed.

■ **JGT (>)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JGT (> Level Sensitive)		Time Compare	3

◆ **Operand Settings**

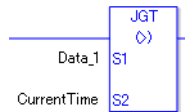
The following shows the configurable conditions for Operands (S1, S2) in the JGT instruction.

The actual number of steps in the JGT instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the JGT instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{Current \text{ time} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Explanation of the JGT Instruction**

Time variables in JGT instructions are structure variables. The following table lists the internal structures.

Time Variable

Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD.
VariableName.MIN	Integer Variable	Minutes are input in BCD.
VariableName.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, S2) in the JGT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant] or Specify integer variable B/ W[constant]	—	X
		Specify integer variable[variable] or Specify integer variable B/ W[variable]	—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC Structure elements are not specified.	1	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC Structure elements are not specified.	1	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the JGT Instruction**

The JGT instruction compares time. When the JGT instruction is executed, S1 is compared to S2. The instruction passes power if the result is $S1 > S2$.

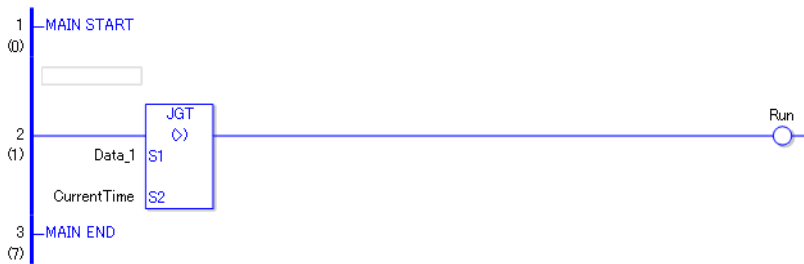
The hour, minute, and second variables are compared simultaneously. To compare a time of 10:20, input 0 for the seconds.

When using the JGT instruction, the only variables you can specify in operands S1 and S2 are time variables.

Program Example

JGT

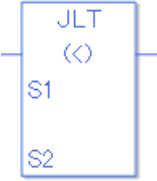
Compares the time variables and determines the result with the coil.



- (1) Compares Data_1 to the current time to determine whether Data_1 is greater. If the result is $S1 > S2$, the instruction passes power and the instruction to the right of the JGT instruction is executed. In the above chart, the OUT instruction to the right of the JGT instruction is executed.

■ **JLT (<)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JLT (< Level Sensitive)		Time Compare	3

◆ **Operand Settings**

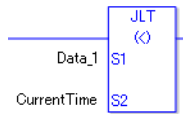
The following shows the configurable conditions for Operands (S1, S2) in the JLT instruction.

The actual number of steps in the JLT instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the JLT instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{Current \text{ time} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Explanation of the JLT Instruction**

Time variables in JLT instructions are structure variables. The following table lists the internal structures.

Time Variable

Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD.
VariableName.MIN	Integer Variable	Minutes are input in BCD.
VariableName.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the JLT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]	—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]	—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X
Real	—	—	X	
	Specify real variable [constant]	—	X	
	Specify real variable [variable]	—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC Structure elements are not specified.	1	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC Structure elements are not specified.	1	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the JLT Instruction**

The JLT instruction compares time. When the JLT instruction is executed, S1 is compared to S2. The instruction passes power if the result is $S1 < S2$.

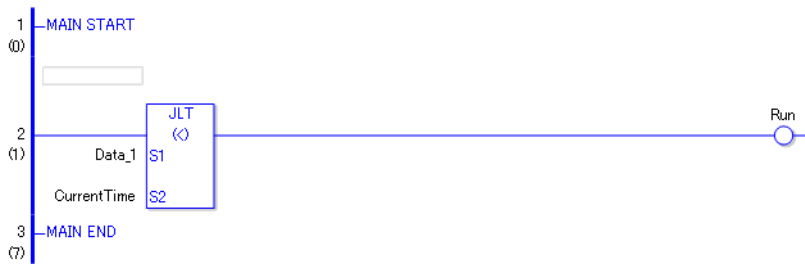
The hour, minute, and second variables are compared simultaneously. To compare a time of 10:20, input 0 for the seconds.

When using the JLT instruction, the only variables you can specify in operands S1 and S2 are time variables.

Program Example

JLT

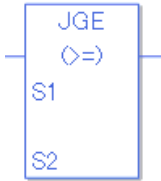
Compares the time variables and determines the result with the coil.



- (1) Compares Data_1 to the current time to determine whether Data_1 is less. If the result is $S1 < S2$, the instruction passes power and the instruction to the right of the JLT instruction is executed. In the above chart, the OUT instruction to the right of the JLT instruction is executed.

■ **JGE (>=)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JGE (>= Level Sensitive)		Time Compare	3

◆ **Operand Settings**

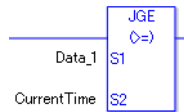
The following shows the configurable conditions for Operands (S1, S2) in the JGE instruction.

The actual number of steps in the JGE instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the JGE instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{Current \text{ time} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Explanation of the JGE Instruction**

Time variables in JGE instructions are structure variables. The following table lists the internal structures.

Time Variable

Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD.
VariableName.MIN	Integer Variable	Minutes are input in BCD.
VariableName.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the JGE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]		—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]		—	X
		Specify integer variable[constant/variable].B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	—		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC Structure elements are not specified.	1	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	L_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC Structure elements are not specified.	1	O
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	X

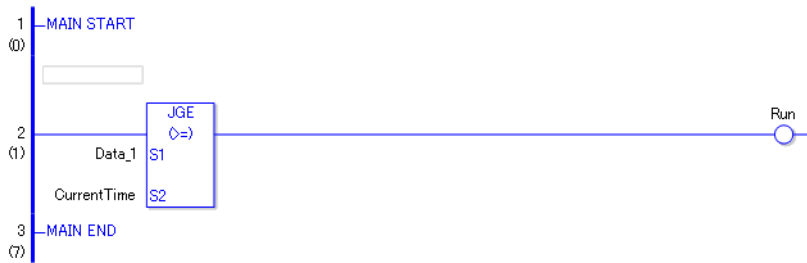
◆ **Explanation of the JGE Instruction**

The JGE instruction compares time. When the JGE instruction is executed, S1 is compared to S2. If the result is $S1 \geq S2$, the instruction passes power. The hour, minute, and time variables are compared simultaneously. To compare a time of 10:20, input 0 for the seconds. When using the JGE instruction, the only variables you can specify in operands S1 and S2 are time variables.

Program Example

JGE

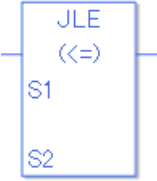
Compares the time variables and determines the result with the coil.



- (1) Compares Data_1 to the current time to determine whether Data_1 is greater or equal. If the result is $S1 \geq S2$, the instruction passes power and the instruction to the right of the JGE instruction is executed. In the above chart, the OUT instruction to the right of the JGE instruction is executed.

■ **JLE (<=)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JLE (<= Level Sensitive)		Time Compare	3

◆ **Operand Settings**

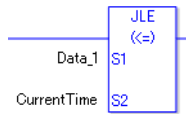
The following shows the configurable conditions for Operands (S1, S2) in the JLE instruction.

The actual number of steps in the JLE instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the JLE instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{Current \text{ time} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Explanation of the JLE Instruction**

Time variables in JLE instructions are structure variables. The following table lists the internal structures.

Time Variable

Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD.
VariableName.MIN	Integer Variable	Minutes are input in BCD.
VariableName.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the JLE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]	—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]	—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/.CV only	—	X
	Date	.YR/.MO/.DAY only	—	X
	Time	.HR/.MIN/.SEC Structure elements are not specified.	1	O
	PID	.KP/.TR/.TD/.PA/.BA/.ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/.CV only	—	X
	N_	.YR/.MO/.DAY only	—	X
	J_	.HR/.MIN/.SEC Structure elements are not specified.	1	O
	U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	X
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

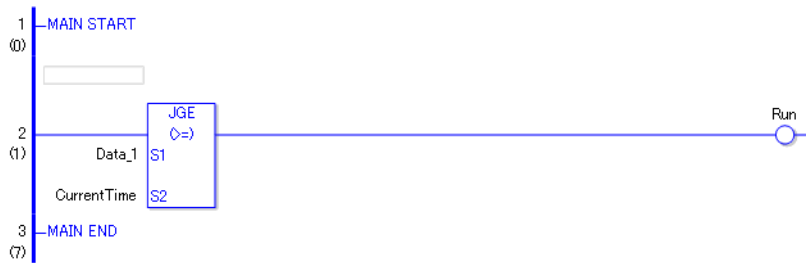
◆ **Explanation of the JLE Instruction**

The JLE instruction compares time. When the JLE instruction is executed, S1 is compared to S2. If the result is $S1 \leq S2$, the instruction passes power. The hour, minute and time variables are compared simultaneously. To compare a time of 10:20, input 0 for the seconds. When using the JLE instruction, the only variables you can specify in operands S1 and S2 are time variables.

Program Example

JLE

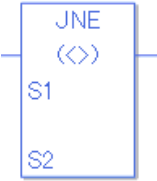
Compares the time variables and determines the result with the coil.



- (1) Compares Data_1 to the current time to determine whether Data_1 is less or equal. If the result is $S1 \leq S2$, the instruction passes power and the instruction to the right of the JLE instruction is executed. In the above chart, the OUT instruction to the right of the JLE instruction is executed.

■ **JNE (<>)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JNE (<> Level Sensitive)		Time Compare	3

◆ **Operand Settings**

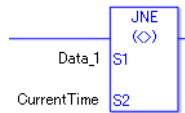
The following shows the configurable conditions for Operands (S1, S2) in the JNE instruction.

The actual number of steps in the JNE instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the JNE instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{Current \text{ time} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Explanation of the JNE Instruction**

Time variables in JNE instructions are structure variables. The following table lists the internal structures.

Time Variable

Time Variable	Variables Settings	Description
VariableName.HR	Integer Variable	Hours are input in BCD.
VariableName.MIN	Integer Variable	Minutes are input in BCD.
VariableName.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the JNE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]		—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC Structure elements are not specified.	1	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC Structure elements are not specified.	1	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

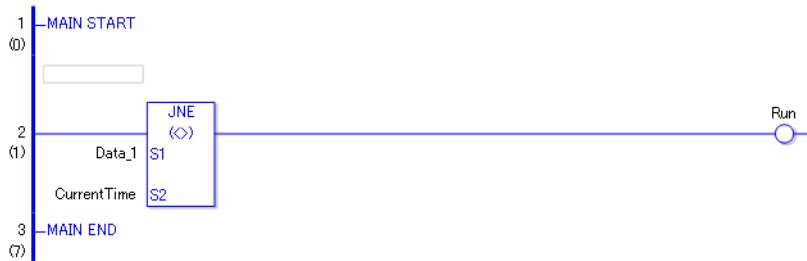
◆ **Explanation of the JNE Instruction**

The JNE instruction compares time. When the JNE instruction is executed, S1 is compared to S2. If the result is $S1 <> S2$, the instruction passes power. The hour, minute and time variables are compared simultaneously. To compare a time of 10:20, input 0 for the seconds. When using the JNE instruction, the only variables you can specify in operands S1 and S2 are time variables.

Program Example

JNE

Compares the time variables and determines the result with the coil.

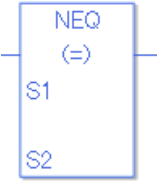


- (1) Compares Data_1 to the current time to determine whether they are unequal. If the result is $S1 <> S2$, the instruction passes power and the instruction to the right of the JNE instruction is executed. In the above chart, the OUT instruction to the right of the JNE instruction is executed.

29.5.17 Compare (Date)

■ NEQ (=)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NEQ (= Level Sensitive)		Date Compare	3

◆ Operand Settings

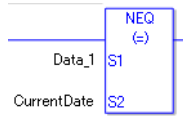
The following shows the configurable conditions for Operands (S1, S2) in the NEQ instruction.

The actual number of steps in the NEQ instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the NEQ instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{\text{Current time} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ Explanation of the NEQ Instruction

The date variables in NEQ instructions are structure variables. The following table lists the internal structures.

Date Variable

Date Variable	Variables Settings	Description
VariableName.YR	Integer Variable	The year is input in BCD.
VariableName.MO	Integer Variable	The month is input in BCD.
VariableName.DAY	Integer Variable	The day is input in BCD.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the NEQ instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]		—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]		—	X
		Specify integer variable[constant/variable].B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	J_	.HR/ .MIN/ .SEC only	—	X
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

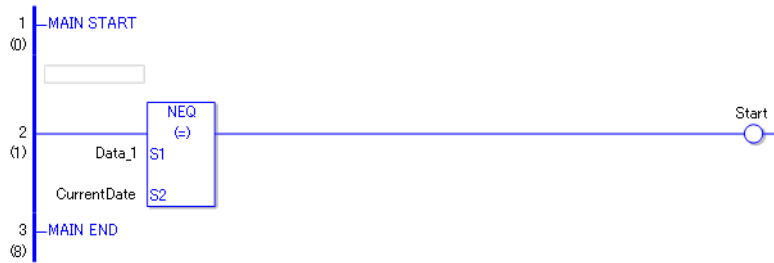
◆ **Explanation of the NEQ Instruction**

The NEQ instruction compares dates. When the NEQ instruction is executed, S1 is compared to S2. If the result is $S1 = S2$, the instruction passes power. The year, month and day variables are compared simultaneously. When using the NEQ instruction, the only variables you can specify in operands S1 and S2 are date variables.

Program Example

NEQ

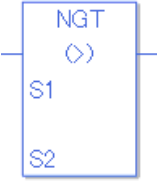
Compares the date variables and determines the result with the coil.



- (1) Compares Data_1 to the current date to determine whether they are equal. If the result is $S1 = S2$, the instruction passes power and the instruction to the right of the NEQ instruction is executed. In the above chart, the OUT instruction to the right of the NEQ instruction is executed.

■ **NGT (>)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NGT (> Level Sensitive)		Date Compare	3

◆ **Operand Settings**

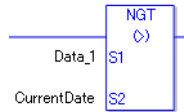
The following shows the configurable conditions for Operands (S1, S2) in the NGT instruction.

The actual number of steps in the NGT instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the NGT instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{Current \text{ date} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Explanation of the NGT Instruction**

Date variables in NGT instructions are structure variables. The following table lists the internal structures.

Date Variable

Date Variable	Variables Settings	Description
VariableName.YR	Integer Variable	The year is input in BCD.
VariableName.MO	Integer Variable	The month is input in BCD.
VariableName.DAY	Integer Variable	The day is input in BCD.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the NGT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]		—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]		—	X
		Specify integer variable[constant/variable].B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

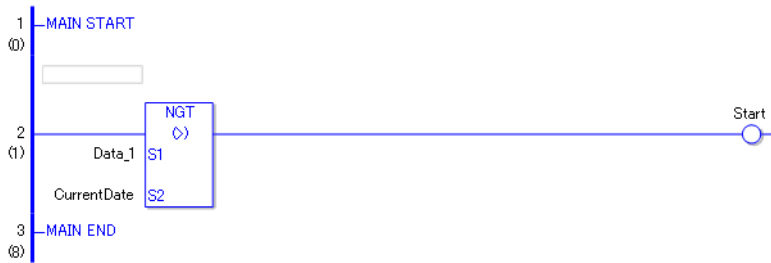
◆ **Explanation of the NGT Instruction**

The NGT instruction compares dates. When the NGT instruction is executed, S1 is compared to S2. If the result is $S1 > S2$, the instruction passes power. The year, month and day variables are compared simultaneously. When using the NGT instruction, the only variables you can specify in operands S1 and S2 are date variables.

Program Example

NGT

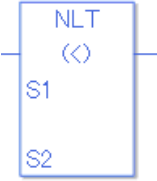
Compares the date variables and determines the result with the coil.



- (1) Compares Data_1 to the current date to determine whether Data_1 is greater. If the result is $S1 > S2$, the instruction passes power and the instruction to the right of the NGT instruction is executed. In the above chart, the OUT instruction to the right of the NGT instruction is executed.

■ **NLT (<)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NLT (< Level Sensitive)		Date Compare	3

◆ **Operand Settings**

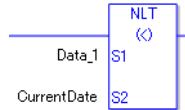
The following shows the configurable conditions for Operands (S1, S2) in the NLT instruction.

The actual number of steps in the NLT instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the NLT instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{Current \text{ date} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Explanation of the NLT Instruction**

Date variables in NLT instructions are structure variables. The following table lists the internal structures.

Date Variable

Date Variable	Variables Settings	Description
VariableName.YR	Integer Variable	The year is input in BCD.
VariableName.MO	Integer Variable	The month is input in BCD.
VariableName.DAY	Integer Variable	The day is input in BCD.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the NLT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]	—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]	—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

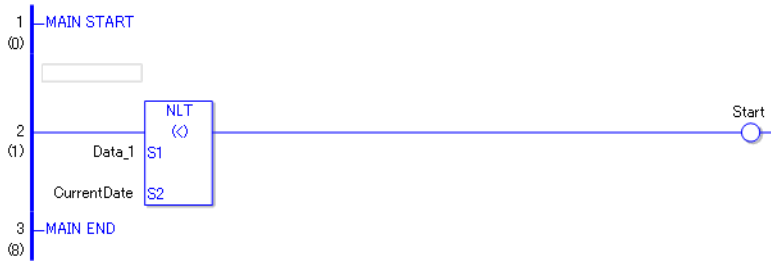
◆ **Explanation of the NLT Instruction**

The NLT instruction compares dates. When the NLT instruction is executed, S1 is compared to S2. If the result is $S1 < S2$, the instruction passes power. The year, month and day variables are compared simultaneously. When using the NLT instruction, the only variables you can specify in operands S1 and S2 are date variables.

Program Example

NLT

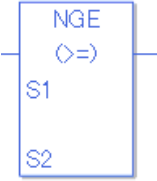
Compares the date variables and determines the result with the coil.



- (1) Compares Data_1 to the current date to determine whether Data_1 is less. If the result is $S1 < S2$, the instruction passes power and the instruction to the right of the NLT instruction is executed. In the above chart, the OUT instruction to the right of the NLT instruction is executed.

■ **NGE (>=)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NGE (>= Level Sensitive)		Date Compare	3

◆ **Operand Settings**

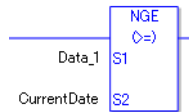
The following shows the configurable conditions for Operands (S1, S2) in the NGE instruction.

The actual number of steps in the NGE instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the NGE instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{Current \text{ date} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Explanation of the NGE Instruction**

Date variables in NGE instructions are structure variables. The following table lists the internal structures.

Date Variable

Date Variable	Variables Settings	Description
VariableName.YR	Integer Variable	The year is input in BCD.
VariableName.MO	Integer Variable	The month is input in BCD.
VariableName.DAY	Integer Variable	The day is input in BCD.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the NGE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Specify by words only (Example: [PLC1]D0000)	—	X
Internal Address	Bit	—	—	X
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X
Symbol	Bit	—	—	X
	Word	—	—	X
Variable Format	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer (including I/O)	Arrays and modifiers are not specified	—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]	—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]	—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable[variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

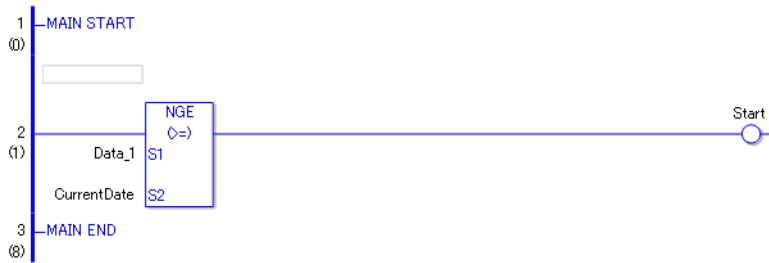
◆ **Explanation of the NGE Instruction**

The NGE instruction compares dates. When the NGE instruction is executed, S1 is compared to S2. If the result is $S1 \geq S2$, the instruction passes power. The year, month and day variables are compared simultaneously. When using the NGE instruction, the only variables you can specify in operands S1 and S2 are date variables.

Program Example

NGE

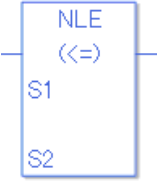
Compares the date variables and determines the result with the coil.



- (1) Compares Data_1 to the current date to determine whether Data_1 is greater or equal. If the result is $S1 \geq S2$, the instruction passes power and the instruction to the right of the NGE instruction is executed. In the above chart, the OUT instruction to the right of the NGE instruction is executed.

■ **NLE (<=)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NLE (<= Level Sensitive)		Date Compare	3

◆ **Operand Settings**

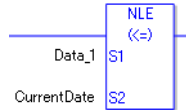
The following shows the configurable conditions for Operands (S1, S2) in the NLE instruction.

The actual number of steps in the NLE instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the NLE instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{Current \text{ date} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Explanation of the NLE Instruction**

Date variables in NLE instructions are structure variables. The following table lists the internal structures.

Date Variable

Date Variable	Variables Settings	Description
VariableName.YR	Integer Variable	The year is input in BCD.
VariableName.MO	Integer Variable	The month is input in BCD.
VariableName.DAY	Integer Variable	The day is input in BCD.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the NLE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]		—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

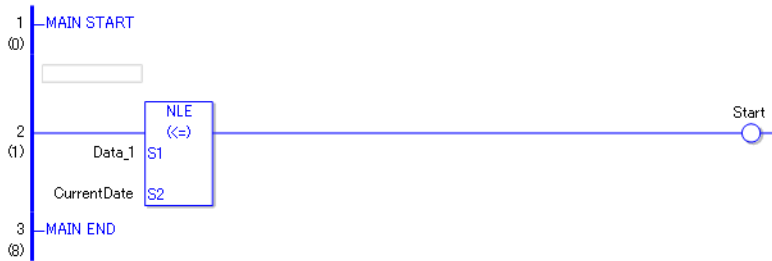
◆ **Explanation of the NLE Instruction**

The NLE instruction compares dates. When the NLE instruction is executed, S1 is compared to S2. If the result is $S1 \leq S2$, the instruction passes power. The year, month and day variables are compared simultaneously. When using the NLE instruction, the only variables you can specify in operands S1 and S2 are date variables.

Program Example

NLE

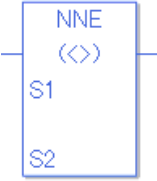
Compares the date variables and determines the result with the coil.



- (1) Compares Data_1 to the current date to determine whether Data_1 is less or equal. If the result is $S1 \leq S2$, the instruction passes power and the instruction to the right of the NLE instruction is executed. In the above chart, the OUT instruction to the right of the NLE instruction is executed.

■ **NNE (<>)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NNE (<> Level Sensitive)		Date Compare	3

◆ **Operand Settings**

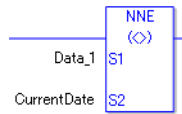
The following shows the configurable conditions for Operands (S1, S2) in the NNE instruction.

The actual number of steps in the NNE instruction depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand S2 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the NNE instruction

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 = 1 \text{ step}\} + \{Current \text{ date} = 1 \text{ step}\} + \{1 \text{ step}\} = 3 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Explanation of the NNE Instruction**

Date variables in NNE instructions are structure variables. The following table lists the internal structures.

Date Variable

Date Variable	Variables Settings	Description
VariableName.YR	Integer Variable	The year is input in BCD
VariableName.MO	Integer Variable	The month is input in BCD
VariableName.DAY	Integer Variable	The day is input in BCD

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, S2) in the NNE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant] or Specify integer variable B/W[constant]		—	X
		Specify integer variable[variable] or Specify integer variable B/W[variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY Structure elements are not specified.	1	O
	J_	.HR/ .MIN/ .SEC only	—	X
	U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

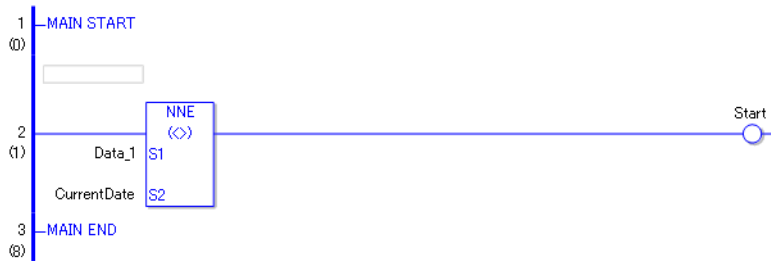
◆ **Explanation of the NNE Instruction**

The NNE instruction compares dates. When the NNE instruction is executed, S1 is compared to S2. If the result is $S1 \neq S2$, the instruction passes power. The year, month and day variables are compared simultaneously. When using the NNE instruction, the only variables you can specify in operands S1 and S2 are date variables.

Program Example

NNE

Compares the date variables and determines the result with the coil.





- (1) Compares Data_1 to the current date to determine whether they are unequal. If the result is $S1 \neq S2$, the instruction passes power and the instruction to the right of the NNE instruction is executed. In the above chart, the OUT instruction to the right of the NNE instruction is executed.

29.5.18 Convert (Data)

■ BCD/BCDP (BCD Convert)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BCD (BCD Convert - Level Sensitive)		Data Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BCDP (BCD Convert - positive transition)		Data Convert	3 to 7

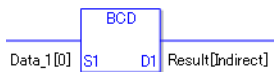
◆ Operand Settings

The following shows the configurable conditions for Operands (S1, D1) in the BCD/BCDP instructions.

The actual number of steps in the BCD/BCDP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in BCD/BCDP instructions (for the number of steps in an operand, refer to the operand settings in the next section.)



{Data_1 [0] = 2 steps} + {Conversion result [Specify indirectly] = 3 steps} + {1 step} = 6 steps

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1, D1) in the BCD/BCDP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format *(Notes 1) S1 = I/O Possible D1 = Input Not Possible	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer *(Notes 1)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant]		2	O
		Specify integer variable [Variable]		3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable [variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format *(Notes 1) S1 = I/O Possible D1 = Input Not Possible	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format *(Notes 2) D1 = Not Possible	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_*(Notes 2)	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant *(Notes 3) D1 = Not Possible	Integer *(Notes 3)	0 to 99999999	1	O
	Float	±1.175494351e-38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the BCD/BCDP Instructions**

The BCD/BCDP instructions convert values to binary coded decimal. The value in S1 is converted to a binary coded decimal and stored in D1.

BCD and BCDP instructions always pass power. The maximum value you can convert in operand S1 is 0x5F5E0FF.

If you try to convert a value that cannot be converted, the value in D1 becomes undefined.

When using BCD/BCDP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type.

Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

When operand D1 is an integer variable



Operand S1	Integer constant	10
Operand D1	Integer variable	OUT1

When operand D1 is an integer variable and you want to input hexadecimal values in operand S1.

When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal.



Operand S1	Integer constant	0xF
Operand D1	Integer variable	OUT2

When converting data in a specified array (integer variable array), specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the variables are the same type.



Data_1	Array size =	5
Result	Array size =	5

The operand specification in the left diagram results in an error.



Data_1 [0]	Array size =	5
Result [n]	Array size =	5

Calculations are performed on individual variables in the array.

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

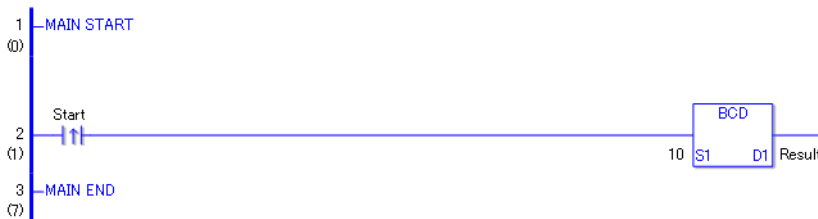
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

BCD

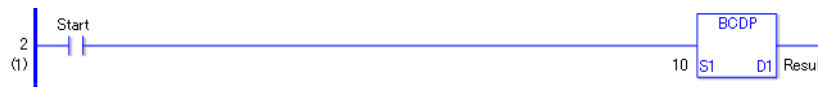
Converts a constant to binary coded decimal and stores it in the result data.



When the positive transition instruction turns ON, the BCD instruction will be executed. When the BCD instruction is executed, 10 (1010 in binary) is converted to a binary coded decimal and the binary code 0001 0000 <F3> is stored in D1. When using a normally open instruction, the BCD instruction is always executed as long as the normally open instruction variable remains ON.

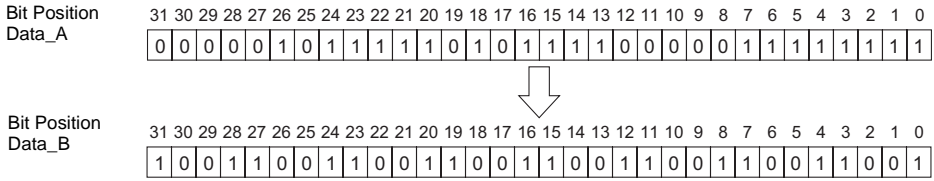
Program Example

BCDP





(1) The BCDP and BCD instructions have different ways of detecting when to execute. In the BCDP instruction, only the upward transition is detected and the BCDP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the BCDP instruction is executed only once (for 1 scan).

For example BCD conversion of S1 (Data_A) = "99999999" to D1 (Data_B).



■ **BIN/BINP (BIN Convert)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BIN (BIN Convert - Level Sensitive)		Data Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BINP (BIN Convert - positive transition)		Data Convert	3 to 7

◆ **Operand Settings**

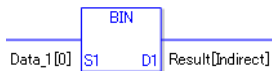
The following shows the configurable conditions for Operands (S1, D1) in the BIN/BINP instructions.

The actual number of steps in the BIN/BINP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in BIN/BINP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



{Data_1 [0] =2 steps} + {Conversion result [Specify indirectly] = 3 steps} + {1 step} = 6 steps

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the BIN/BINP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format *(Notes 1) S1 = I/O Possible D1 = Input Not Possible	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format *(Notes 1) S1 = I/O Possible D1 = Input Not Possible	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format *(Notes 2) D1 = Not Possible	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_ *(Notes 2)	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant *(Notes 3) D1 = Not Possible	Integer *(Notes 3)	0 to 99999999 (BCD value)	1	O
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X

◆ **Explanation of the BIN/BINP Instructions**

The BIN/BINP instructions converts BCD values to binary. The value in S1 is converted to binary and stored in D1.

BIN and BINP instructions always pass power. The maximum value you can convert in operand S1 is 0x5F5E0FF.

If you try to convert a value that cannot be converted, the value in D1 becomes undefined.

When using the BIN/BINP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type.

Specify the same variable type in operands S1 and D1.

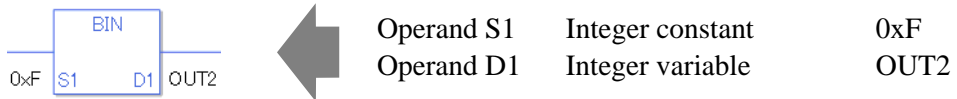
Refer to the following for specifying a constant.

When operand D1 is an integer variable



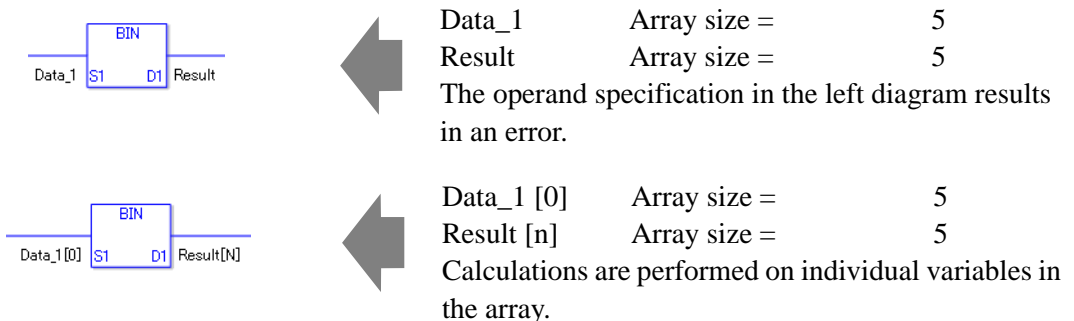
When operand D1 is an integer variable and you want to input hexadecimal values in operand S1.

When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal.



When converting data in a specified array (integer variable array), specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

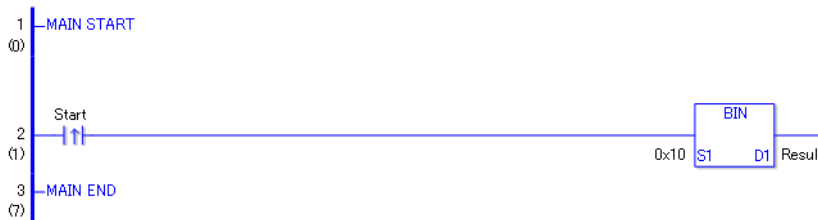
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

BIN

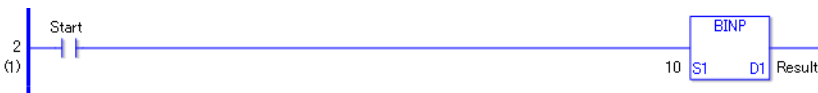
Converts a constant from BCD to binary and stores the converted value in the result data.



- (1) When the positive transition instruction turns ON, the BIN instruction will be executed. When the BIN instruction is executed, 0001 0000 (10 in hexadecimal) is converted to binary and the value 1010 is stored in D1. When using a normally open instruction, the BIN instruction is always executed as long as the normally open instruction variable remains ON.

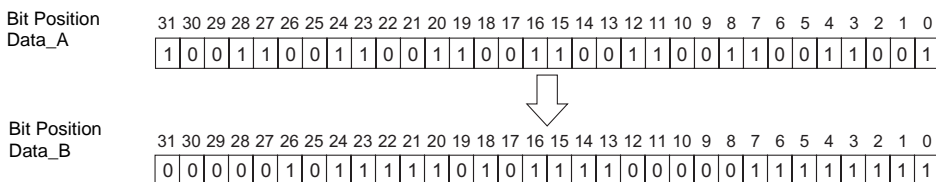
Program Example

BINP





- (1) The BINP and BIN instructions have different ways of detecting when to execute. In the BINP instruction, only the upward transition is detected and the BINP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the BINP instruction is executed only once (for 1 scan).

For example BIN conversion of S1 (Data_A) = "99999999" BCD to D1 (Data_B).



■ ENCO/ENCOP (Encode)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ENCO (Encode - Level Sensitive)		Data Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ENCOP (Encode - positive transition)		Data Convert	3 to 7

◆ Operand Settings

The following shows the configurable conditions for Operands (S1, D1) in the ENCO/ENCOP instructions.

The actual number of steps in the ENCO/ENCOP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in ENCO/ENCOP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



{Data_1 [0] =2 steps} + {Conversion result [Specify indirectly] = 3 steps} + {1 step} = 6 steps

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the ENCO/ENCOP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format *(Notes 1) S1 = I/O Possible D1 = Input Not Possible	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] array	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Variable Format *(Notes 1) S1 = I/O Possible D1 = Input Not Possible	Real	—	—	X	
		Specify real variable [constant]	—	X	
		Specify real variable [variable]	—	X	
	Timer	.PT/.ET only	2	O	
	Counter	.PV/ .CV only	2	O	
	Date	.YR/ .MO/ .DAY only	2	O	
	Time	.HR/ .MIN/ .SEC only	2	O	
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format *(Notes 2) D1 = Not Possible	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_*(Notes 2)	—	1	O	
	Q_	—	1	O	
		D_	Modifiers are not specified	1	O
			D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X	
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT/.ET only	2	O	
	C_	.PV/ .CV only	2	O	
	N_	.YR/ .MO/ .DAY only	2	O	
	J_	.HR/ .MIN/ .SEC only	2	O	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O		
Constant *(Notes 3) D1 = Not Possible	Integer *(Notes 3)	-2147483648 to 2147483647	1	O	
	Float	±1.175494351e-38 to ±3.402823466e+38	—	X	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	X	

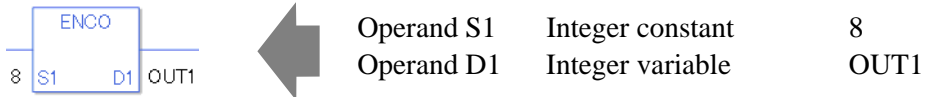
◆ **Explanation of the ENCO/ENCOP Instructions**

The ENCO/ENCOP instructions encode values. The value in S1 is encoded and saved in D1. Among the 32 bits of S1, the position of the ON bit is output to D1 as a binary value. When multiple bits are ON in S1, the uppermost bit position is output. The ENCO/ENCOP instructions always pass power.

When using ENCO/ENCOP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type.

Refer to the following for specifying a constant.

When operand D1 is an integer variable

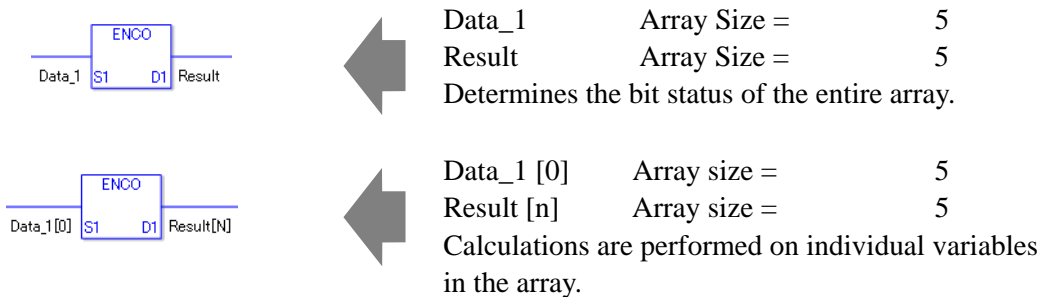


When operand D1 is an integer variable and you want to input hexadecimal values in operand S1.

When 0x (zero and lower case "x") is input, the following values will be interpreted as hexadecimal values.



To convert data in a specified array (integer variable array), you can either specify the entire array with operands S1 and D1, or specify the array elements individually.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

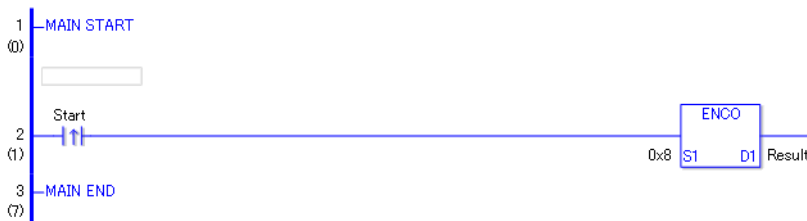
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

ENCO

Converts a constant and stores the converted value in the result data.



When the positive transition instruction turns ON, the ENCO instruction will be executed. When the ENCO instruction is executed, 0000 1000 (8 in hexadecimal) is converted and the binary value 0011 (3) is stored in D1. When using a normally open, the ENCO instruction is always executed as long as the normally open instruction variable remains ON.

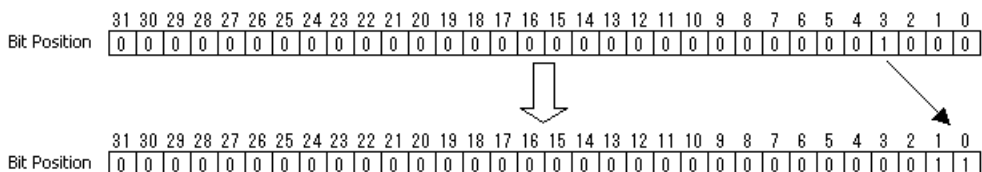
Program Example

ENCOP





(1) The ENCOP and ENCO instructions have different ways of detecting when to execute. In the ENCOP instruction, only the upward transition is detected and the ENCOP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the ENCOP instruction is executed only once (for 1 scan).

Example: When 0x00000008 is input in S1, the output in D1 will be 0x00000003.



■ **DECO/DECOP (Decode)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
DECO (Decode - Level Sensitive)		Data Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
DECOP (Decode - positive transition)		Data Convert	3 to 7

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the DECO/DECOP instructions.

The actual number of steps in the DECO/DECOP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in DECO/DECOP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



{Data_1 [0] = 2 steps} + {Conversion result [Specify indirectly] = 3 steps} + {1 step} = 6 steps

One final step is required in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the DECO/DECOP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	O
Internal Address	Bit	—	—	X
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	O
Symbol	Bit	—	—	X
	Word	—	1	O
Variable Format *(Notes 1) S1 = I/O Possible D1 = Input Not Possible	Bit	Specify a bit	—	X
		Specify bit array ([constant])	—	X
		Specify bit array ([variable])	—	X
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant] array	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]	—	X
	Float	—	—	X
		Specify float variable[constant]	—	X
		Specify float variable [variable]	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format *(Notes 1) S1 = I/O Possible D1 = Input Not Possible	Real	—	—	X
		Specify real variable [constant]	—	X
		Specify real variable [variable]	—	X
	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format *(Notes 2) D1 = Not Possible	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_*(Notes 2)	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W [constant]	—	X
		D_****.B/W [address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant *(Notes 3) D1 = Not Possible	Integer *(Notes 3)	0 to 131071 (Specified array)	1	O
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	X

◆ **Explanation of the DECO/DECOP Instructions**

The DECO/DECOP instructions decode values. The value in S1 is decoded and saved in D1. The single bit position in D1 corresponding to the value in S1 is turned ON. When you use an output array, you can decode a bit position up to the maximum (4096 x 32 - 1 = 131071). The DECO/DECOP instructions always pass power. When using DECO/DECOP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1. Refer to the following for specifying a constant.

When operand D1 is an integer variable

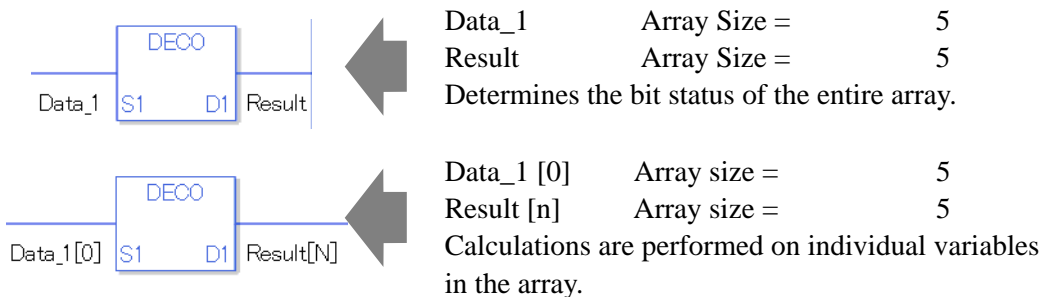


When operand D1 is an integer variable and you want to input hexadecimal values in operand S1.

When 0x (zero and lower case "x") is input, the following values are interpreted as hexadecimal values.



To convert data in a specified array (integer variable array), you can either specify the entire array with operands S1 and D1, or specify the array elements individually.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

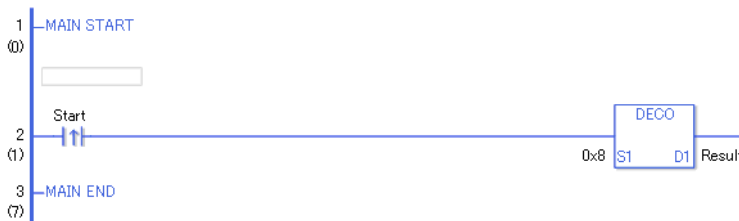
When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

DECO

Converts a constant and stores the converted value in the result data.



When the positive transition instruction turns ON, the DECO instruction will be executed.

When the DECO instruction is executed, 0000 1000 (8 in hexadecimal) is converted and the binary value 1 0000 0000 is stored in D1.

When using a normally open instruction, the DECO instruction is always executed as long as the normally open instruction variable remains ON.

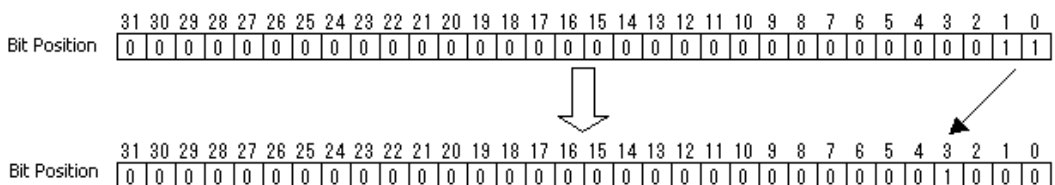
Program Example

DECOP





(1) The DECOP and DECO instructions have different ways of detecting when to execute. In the DECOP instruction, only the upward transition is detected and the DECO instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the DECOP instruction is executed only once (for 1 scan).

For example When 3 is input in S1, the output D1 becomes 8.



■ **RAD/RADP (Convert to Radians)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
RAD (Convert to Radians - Level Sensitive)		Data Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
RADP (Convert to Radian - positive transition)		Data Convert	3 to 7

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1 and D1) in the RAD/RADP instructions.

The actual number of steps in the RAD/RADP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in RAD/RADP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{ \text{Data_1 [0]} = 2 \text{ steps} \} + \{ \text{Conversion result [N]} = 3 \text{ steps} \} + \{ 1 \text{ step} \} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1 and D1) in the RAD/RADP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
	Date	.YR/ .MO/ .DAY only		—	X
Time	.HR/ .MIN/ .SEC only		—	X	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	1	O	
	R_	—	1	O	
	T_	.PT/.ET only	—	X	
	C_	.PV/ .CV only	—	X	
	N_	.YR/ .MO/ .DAY only	—	X	
	J_	.HR/ .MIN/ .SEC only	—	X	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X		
Constant *(Notes 1) D1 = Not Possible	Integer	- 2147483648 to 2147483647	—	X	
	Float *(Notes 1)	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	O	
	Real *(Notes 1)	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	O	

◆ **Explanation of the RAD/RADP Instructions**

RAD and RADP instructions are radian conversion instructions that convert degrees to radians. When the RAD instruction is executed and passes power, the number of degrees is input in S1, and the converted number of radians is stored in D1. Pi is approximately 3.1415926535897 (real number). RAD and RADP instructions always pass power.

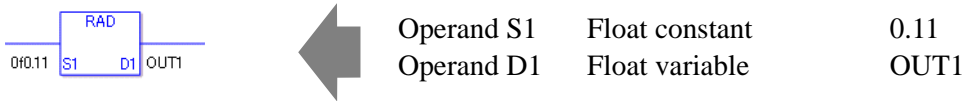
When using RAD/RADP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type.

Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

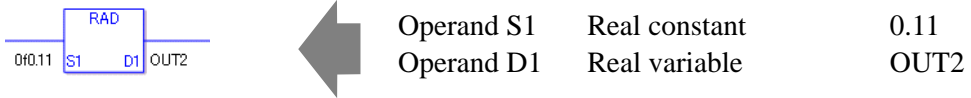
When operand D1 is a float variable

When Of (zero and lower case "f") is input, the following values are interpreted as float values.



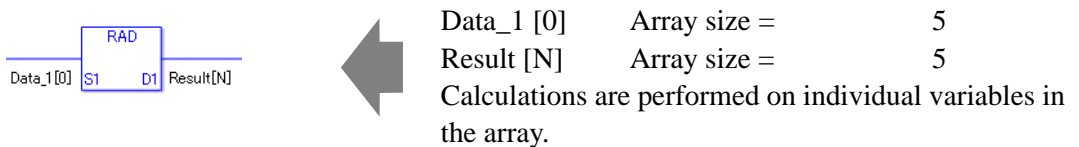
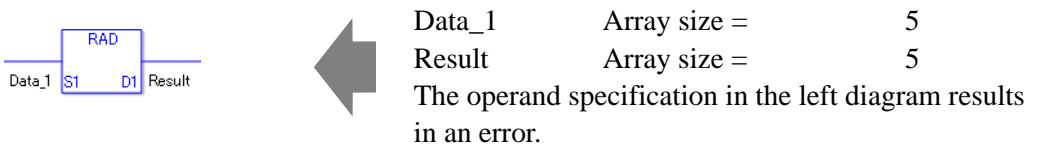
When operand D1 is a real variable

When Or (zero and lower case "r") is input, the following values are interpreted as real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

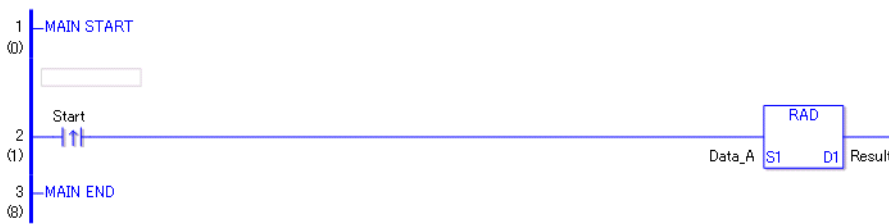
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

RAD



- (1) When the positive transition instruction turns ON, the RAD instruction will be executed. When the RAD instruction is executed, the result of Data_A is stored in D1. When using a normally open, the RAD instruction is always executed as long as the normally open instruction variable remains ON.

Program Example



RADP



- (1) The RADP and RAD instructions have different ways of detecting when to execute. In the RADP instruction, only the upward transition is detected and the RADP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the RADP instruction is executed only once (for 1 scan).

■ **DEG/DEGP (Convert to Degrees)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
DEG (Convert to Degrees - Level Sensitive)		Data Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
DEGP (Convert to Degrees - positive transition)		Data Convert	3 to 7

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the DEG/DEGP instructions.

The actual number of steps in the DEG/DEGP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in DEG/DEGP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{ \text{Data}_1 [0] = 2 \text{ steps} \} + \{ \text{Conversion result} [N] = 3 \text{ steps} \} + \{ 1 \text{ step} \} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the DEG/DEGP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
Date	.YR/ .MO/ .DAY only		—	X	
Time	.HR/ .MIN/ .SEC only		—	X	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	1	O	
	R_	—	1	O	
	T_	.PT/.ET only	—	X	
	C_	.PV/.CV only	—	X	
	N_	.YR/.MO/.DAY only	—	X	
	J_	.HR/.MIN/.SEC only	—	X	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	X		
Constant *(Notes 1) D1 = Not Possible	Integer	- 2147483648 to 2147483647	—	X	
	Float *(Notes 1)	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	O	
	Real *(Notes 1)	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	O	

◆ **Explanation of the DEG and DEGP Instructions**

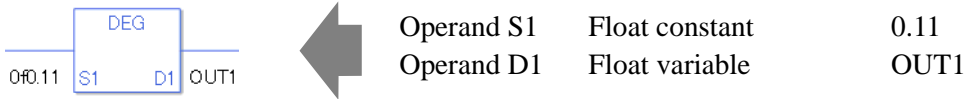
The DEG/DEGP instructions convert values to degrees. The unit of angular measure, radian, is converted to degrees and stored in D1.

Pi is approximately 3.1415926535897 (real number). DEG and DEGP instructions always pass power. When using DEG and DEGP instructions, specify the same data type in operands S1 and D1 to avoid errors.

Refer to the following for specifying a constant.

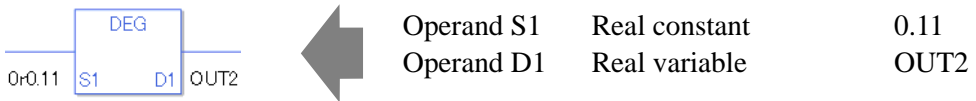
When operand D1 is a float variable

When Of (zero and lower case "f") is input, the following values are interpreted as float values.



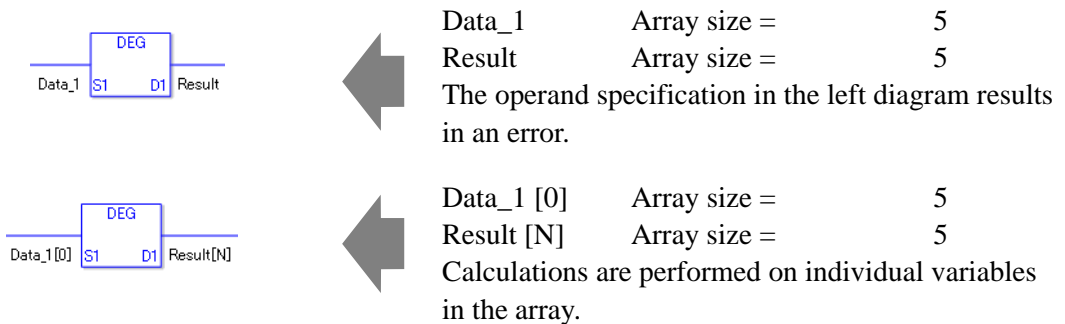
When operand D1 is a real variable

When Or (zero and lower case "r") is input, the following values are interpreted as real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

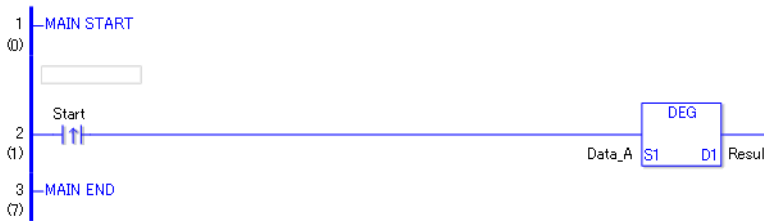
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

DEG



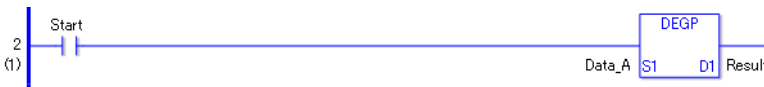
(1) When the positive transition instruction turns ON, the DEG instruction will be executed.

When the DEG instruction is executed, the result of Data_A is stored in D1.

When using a normally open, the DEG instruction is always executed as long as the normally open instruction variable remains ON.

Program Example



DEGP



(1) The DEGP and DEG instructions have different ways of detecting when to execute. In the DEGP instruction, only the upward transition is detected and the DEGP is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the DEGP instruction is executed only once (for 1 scan).

■ **SCL/SCLP (Scale Convert)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SCL (Scale Convert - Level Sensitive)		Data Convert	7 to 11
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SCLP (Scale Convert - positive transition)		Data Convert	7 to 11

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the SCL/SCLP instructions.

The actual number of steps in the SCL/SCLP instruction depends on the specified operands.

The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Convert the number of steps in the SCL/SCLP instructions

(For the number of steps in an operand, refer to the operand settings in the next page.)



$$\{ \text{Data}_1 [0] = 2 \text{ steps} \} + \{ \text{Conversion result} [N] = 3 \text{ steps} \} + \{ 5 \text{ steps} \} = 10 \text{ steps}$$

The last five steps are included in the instruction. Be sure to add those five steps.

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the SCL/SCLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format *(Notes 1) D1 = Input Not Possible	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer *(Notes 1)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant]		2	O
		Specify integer variable [Variable]		3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O
	Timer	.PT/.ET only		2	O
	Counter	.PV/ .CV only		2	O
Date	.YR/ .MO/ .DAY only		2	O	
Time	.HR/ .MIN/ .SEC only		2	O	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		2	O	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Address Format *(Notes 2) D1 = Not Possible	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	L_ *(Notes 2)	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	1	O
	R_	—	1	O
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant *(Notes 3) D1 = Constant Not Possible	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	2	O

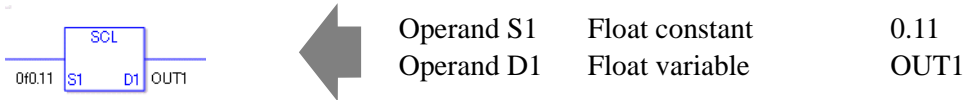
◆ **Explanation of the SCL/ SCLP Instructions**

The SCL/SCLP instructions convert values to scales. The value in S1 is converted according to the upper and lower limits and the converted value is stored in D1. An error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in operands S1 and D1.

Refer to the following for specifying a constant.

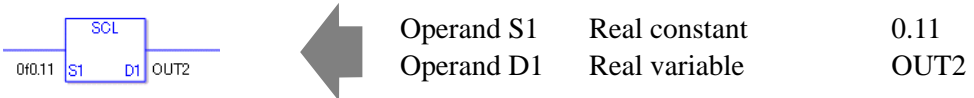
When operand D1 is a float variable

When 0f (zero and lower case "f") is input, the following values are interpreted as float values.



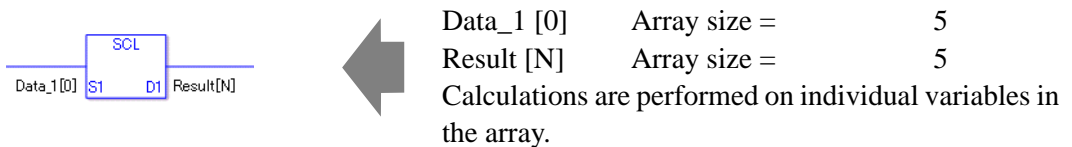
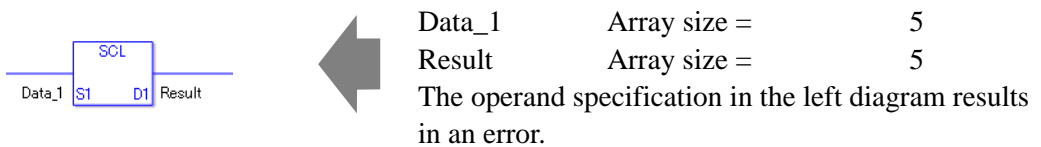
When operand D1 is a real variable

When 0r (zero and lower case "r") is input, the following values are interpreted as real values.



When calculating data in a specified array, specify the array with Data [0] or Data [N] (N is an integer variable).

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

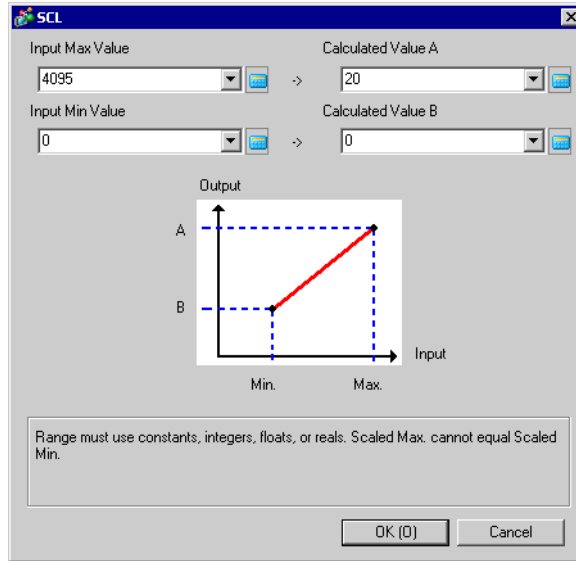
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

◆ **Upper and Lower Limits for Input and Output**

Double-click the SCL instruction to display the following dialog box. In the dialog box, specify the settings for the maximum and minimum input values and for output A and output B.

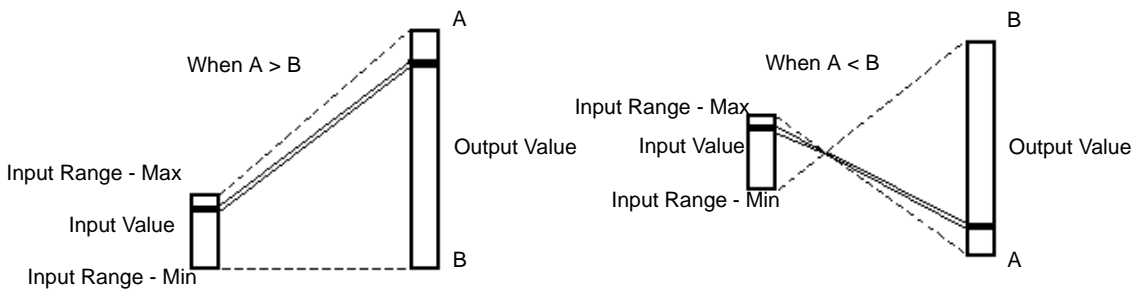


(Notes 1) When setting the maximum/minimum input values and output values A and B, you cannot indirectly designate array elements.

Array Variable Name: Data
 Array Size: 5
 O Data [0] X Data [N]

(Note 2) When using real or float variables in operands S1 or D1, and using constants to define the minimum/maximum input and output values in A and B, use "0r" and "0f" to denote real and float values.

When output value A is greater than output value B

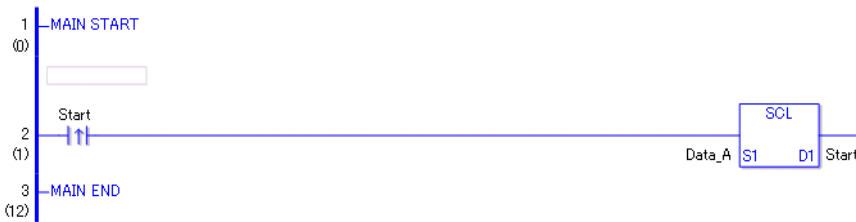


Program Example

SCL

Converting an analog input value (0 to 4095) to a current value in the range of 4 to 20 [ma] and expressing the value as a decimal.

In the SCL instruction settings in the dialog box, set maximum input value = 0r4095, minimum input value = 0r0, A = 0r20, and B =0r4.



- (1) When the positive transition instruction turns ON, the SCL instruction will be executed. When the SCL instruction is executed, the result of Data_A is stored in D1. When using a normally open, the SCL instruction is always executed as long as the normally open instruction variable remains ON.

Program Example

SCLP





- (1) The SCLP and SCL instructions differ by when they run. In the SCLP instructions, even when using a normally open instruction, only the positive transition is detected and the SCLP instruction is executed. Therefore, the SCLP instruction is executed only for one scan, even when the normally open instruction bit remains ON. 31

29.5.19 Convert Type

■ I2F/I2FP (Integer to Float Conversion)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
I2F (Integer to Float Conversion - Level Sensitive)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
I2FP (Integer to Float Conversion - positive transition)		Type Convert	3 to 7

◆ Operand Settings

The following shows the configurable conditions for Operands (S1, D1) in the I2F/I2FP instructions.

The actual number of steps in the I2F/I2FP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the I2F/I2FP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{\text{Data_1 [0]} = 2 \text{ steps}\} + \{\text{Conversion result [N]} = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the I2F/I2FP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format *(Notes 1) including I/O	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer *(Notes 1)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant]		2	O
		Specify integer variable [Variable]		3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Arrays and modifiers are not specified		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	Arrays and modifiers are not specified		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X
	Timer	.PT/.ET only		2	O
	Counter	.PV/ .CV only		2	O
	Date	.YR/ .MO/ .DAY only		2	O
Time	.HR/ .MIN/ .SEC only		2	O	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		2	O	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the I2F/I2FP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	—		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
	Date	.YR/ .MO/ .DAY only		—	X
Time	.HR/ .MIN/ .SEC only		—	X	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	1	O
	R_	—	—	X
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
	J_	.HR/ .MIN/ .SEC only	—	X
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Explanation of the I2F/I2FP Instructions**

The I2F/I2FP instructions convert integer variables to float variables. Specify the integer variable or constant in S1 that you want to convert, and specify float variable for the conversion output in D1. You can specify only an integer variable for input in S1 and a float variable for output in D1. Use the convert instruction when you want to use different variable types in a calculation or Comparison.

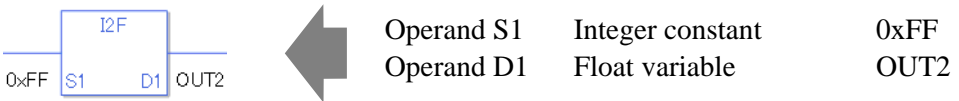
Refer to the following for specifying a constant.

When operand S1 is an integer constant



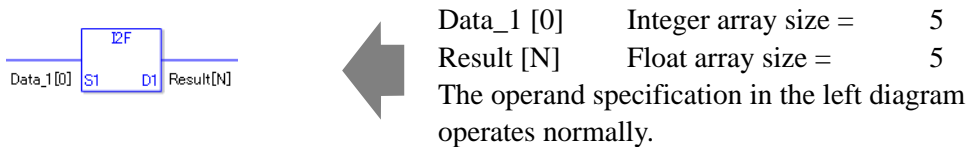
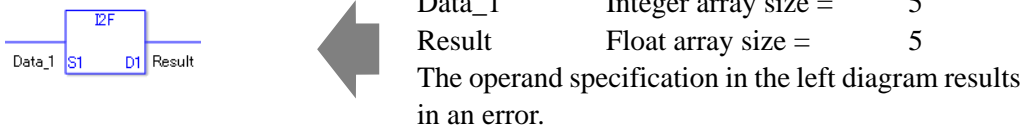
When operand S1 is an integer constant and you want to input a hexadecimal value in operand S1.

When 0x (zero and lower case "x") is input, the following values become hexadecimal values.



Note that specified arrays (entire arrays) cannot be converted.

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

I2F



(1) When the positive transition instruction turns ON, the I2F instruction will be executed.

When the I2F instruction is executed, the result of the I2F conversion of Data_A is stored in D1.

When using a normally open instruction, the I2F instruction is always executed as long as the normally open instruction variable remains ON.

Program Example



I2FP



(1) The I2FP and I2F instructions have different ways of detecting when to execute. In the I2FP instruction, only the upward transition is detected and the I2FP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the I2FP instruction is executed only once (for 1 scan).

■ **I2R/I2RP (Integer to Real Conversion)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
I2R (Integer to Real Conversion - Level Sensitive)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
I2RP (Integer to Real Conversion - positive transition)		Type Convert	3 to 7

◆ **Operand Settings**

The following shows the configurable conditions of Operands (S1, D1) in the I2R/I2RP instructions.

The actual number of steps in the I2R/I2RP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the I2R/I2RP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 [0] = 2 \text{ steps}\} + \{\text{Conversion result } [N] = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the I2R/I2RP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format *(Notes 1) including I/O	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer *(Notes 1)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant]		2	O
		Specify integer variable [Variable]		3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Arrays and modifiers are not specified		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	Arrays and modifiers are not specified		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X
	Timer	.PT/.ET only		2	O
	Counter	.PV/ .CV only		2	O
	Date	.YR/ .MO/ .DAY only		2	O
	Time	.HR/ .MIN/ .SEC only		2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		2	O	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	1	O
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/.CV only	2	O
	N_	.YR/.MO/.DAY only	2	O
	J_	.HR/.MIN/.SEC only	2	O
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	O	
Constant	Integer	– 2147483648 to 2147483647	1	O
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the I2R/I2RP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
	Date	.YR/ .MO/ .DAY only		—	X
Time	.HR/ .MIN/ .SEC only		—	X	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

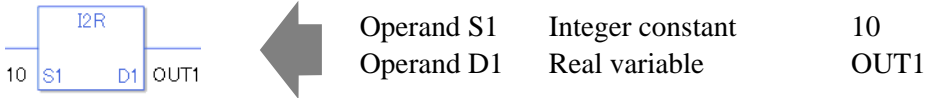
Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	1	O	
	T_	.PT/.ET only	—	X	
	C_	.PV/ .CV only	—	X	
	N_	.YR/ .MO/ .DAY only	—	X	
J_	.HR/ .MIN/ .SEC only	—	X		
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X		
Constant	Integer	- 2147483648 to 2147483647	—	X	
	Float	±1.175494351e-38 to ±3.402823466e+38	—	X	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	X	

◆ **Explanation of the I2R/I2RP Instructions**

The I2R/I2RP instructions convert integer variables to real variables. Specify the integer variable or constant in S1 that you want to convert, and specify real variable for the conversion output in D1. You can specify only an integer variable for input in S1 and a real variable for output in D1. Use the convert instruction when you want to use different variable types in a calculation or Comparison.

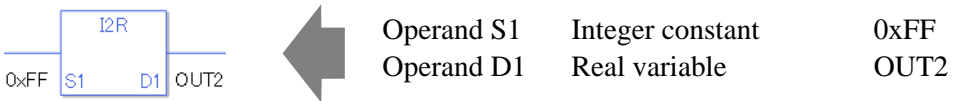
Refer to the following for specifying a constant.

When operand S1 is an integer constant



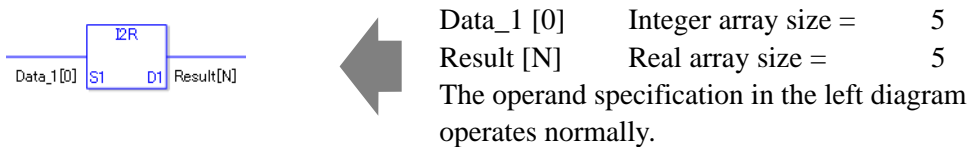
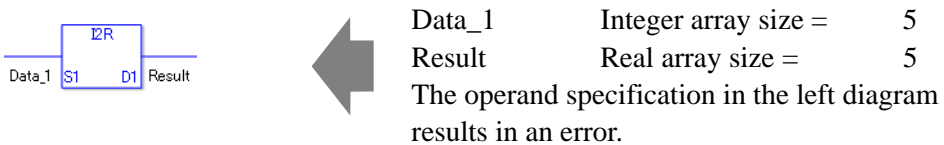
When operand S1 is an integer constant and you want to input a hexadecimal value in operand S1.

When 0x (zero and lower case "x") is input, the following values become hexadecimal values.



Note that specified arrays (entire arrays) cannot be converted.

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

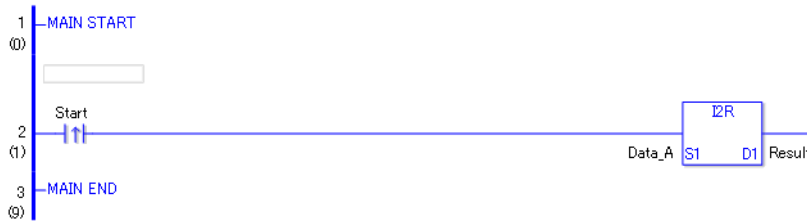
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

I2R



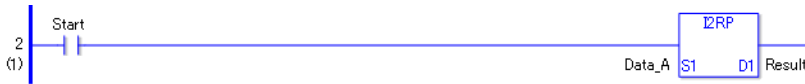
(1) When the positive transition instruction turns ON, the I2R instruction will be executed.

When the I2R instruction is executed, the result of the I2R conversion of Data_A is stored in D1.

When using a normally open instruction, the I2R instruction is always executed as long as the normally open instruction variable remains ON.

Program Example



I2RP



(1) The I2RP and I2R instructions have different ways of detecting when to execute. In the I2RP instruction, only the upward transition is detected and the I2RP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the I2RP instruction is executed only once (for 1 scan).

■ F2I/F2IP (Float to Integer Conversion)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
F2I (Float to Integer Conversion - Level Sensitive)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
F2IP (Float to Integer Conversion - positive transition)		Type Convert	3 to 7

◆ Operand Settings

The following shows the configurable conditions for Operands (S1, D1) in the F2I/F2IP instructions.

The actual number of steps in the F2I/F2IP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the F2I/F2IP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 [0] = 2 \text{ steps}\} + \{Conversion \text{ result } [N] = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the F2I/F2IP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	—		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
	Date	.YR/ .MO/ .DAY only		—	X
Time	.HR/ .MIN/ .SEC only		—	X	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	1	O	
	R_	—	—	X	
	T_	.PT/.ET only	—	X	
	C_	.PV/ .CV only	—	X	
	N_	.YR/ .MO/ .DAY only	—	X	
	J_	.HR/ .MIN/ .SEC only	—	X	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X		
Constant	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X	

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the F2I/F2IP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer *(Notes 1)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant]		2	O
		Specify integer variable [Variable]		3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Arrays and modifiers are not specified		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	Arrays and modifiers are not specified		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X
	Timer	.PT/.ET only		2	O
	Counter	.PV/ .CV only		2	O
	Date	.YR/ .MO/ .DAY only		2	O
	Time	.HR/ .MIN/ .SEC only		2	O
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		2	O	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	1	O	
	D_	Modifiers are not specified		1	O
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT/.ET only	2	O	
	C_	.PV/.CV only	2	O	
	N_	.YR/.MO/.DAY only	2	O	
	J_	.HR/.MIN/.SEC only	2	O	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	O		
Constant	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X	

◆ **Explanation of the F2I/F2IP Instructions**

The F2I/F2IP instructions convert float variables to integer variables. Specify the float variable or constant in S1 that you want to convert, and specify integer variable for the conversion output in D1. You can specify only a float variable for input in S1 and an integer variable for output in D1. Use the convert instruction when you want to use different variable types in a calculation or Comparison.

Refer to the following for specifying a constant.

When operand S1 is a float constant



Operand S1	Float constant	0f0.11
Operand D1	Integer variable	OUT1

Note that specified arrays (entire arrays) cannot be converted.

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



Data_1	Float array size =	5
Result	Integer array size =	5

The operand specification in the left diagram results in an error.



Data_1 [0]	Float array size =	5
Result [N]	Integer array size =	5

The operand specification in the left diagram operates normally.

◆ **System Variables Indicating Execution Results**

- #L_CalcZero System variable that turns ON when the result is 0.
- #L_CalcCarry System variable that turns ON when the result overflows.
- #L_CalcErrCode System variable that stores the error code when an operation error occurs.

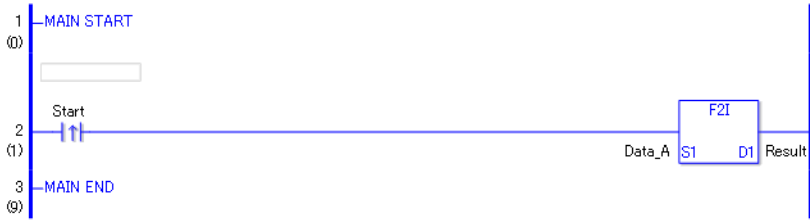
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

F2I



- (1) When the positive transition instruction turns ON, the F2I instruction will be executed. When the F2I instruction is executed, the result of the F2I conversion of Data_A is stored in D1. When using normally open instruction, the F2I instruction is always executed as long as the normally open instruction variable remains ON.

Program Example



F2IP



- (1) The F2IP and F2I instructions have different ways of detecting when to execute. In the F2IP instruction, only the upward transition is detected and the F2IP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the F2IP instruction is executed only once (for 1 scan).

■ **F2R/F2RP (Float to Real Conversion)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
F2R (Float to Real Conversion - Level Sensitive)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
F2RP (Float to Real Conversion/ positive transition)		Type Convert	3 to 7

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the F2R/F2RP instructions.

The actual number of steps in the F2R/F2RP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the F2R/F2RP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{ \text{Data}_1 [0] = 2 \text{ steps} \} + \{ \text{Conversion result} [N] = 3 \text{ steps} \} + \{ 1 \text{ step} \} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the F2R/F2RP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	—		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
	Date	.YR/ .MO/ .DAY only		—	X
Time	.HR/ .MIN/ .SEC only		—	X	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	1	O	
	R_	—	—	X	
	T_	.PT/.ET only	—	X	
	C_	.PV/ .CV only	—	X	
	N_	.YR/ .MO/ .DAY only	—	X	
	J_	.HR/ .MIN/ .SEC only	—	X	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X		
Constant	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	1	O	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X	

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the F2R/F2RP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer *(Notes 1)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
	Date	.YR/ .MO/ .DAY only		—	X
	Time	.HR/ .MIN/ .SEC only		—	X
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

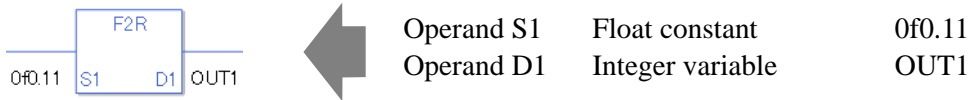
Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	1	O	
	T_	.PT/.ET only	—	X	
	C_	.PV/.CV only	—	X	
	N_	.YR/.MO/.DAY only	—	X	
	J_	.HR/.MIN/.SEC only	—	X	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	X		
Constant	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X	

◆ **Explanation of the F2R/F2RP Instructions**

The F2R/F2RP instructions convert float variables to real variables. Specify the float variable or constant in S1 that you want to convert, and specify real variable for the conversion output in D1. You can specify only a float variable for input in S1 and a real variable for output in S2. Use the convert instruction when you want to use different variable types in the calculation and Comparison.

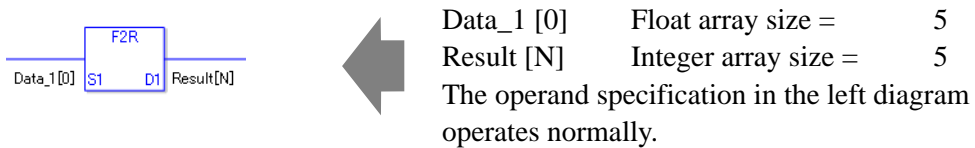
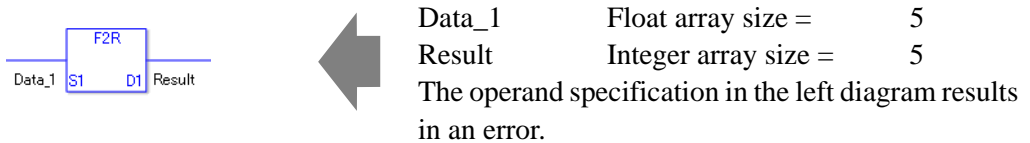
Refer to the following for specifying a constant.

When operand S1 is a float constant



Note that specified arrays (entire arrays) cannot be converted.

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

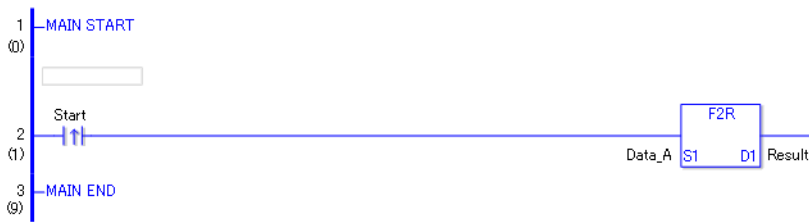
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

F2R



(1) When the positive transition instruction turns ON, the F2R instruction will be executed.

When the F2R instruction is executed, the result of the F2R conversion of Data_A is stored in D1.

When using a normally open instruction, the F2R instruction is always executed as long as the normally open instruction variable remains ON.

Program Example



F2RP



(1) The F2RP and F2R instructions have different ways of detecting when to execute. In the F2RP instruction, only the upward transition is detected and the F2RP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the F2RP instruction is executed only once (for 1 scan).

■ **R2I/R2IP (Real to Integer Conversion)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
R2I (Real to Integer Conversion - Level Sensitive)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
R2IP (Real to Integer Conversion - positive transition)		Type Convert	3 to 7

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the R2I/R2IP instructions.

The actual number of steps in the R2I/R2IP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the R2I/R2IP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 [0] = 2 \text{ steps}\} + \{\text{Conversion result } [N] = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the R2I/R2IP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
	Date	.YR/ .MO/ .DAY only		—	X
Time	.HR/ .MIN/ .SEC only		—	X	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
J_	.HR/ .MIN/ .SEC only	—	X	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	Integer	- 2147483648 to 2147483647	—	X
	Float	±1.175494351e-38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	1	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the R2I/R2IP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer *(Notes 1)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant]		2	O
		Specify integer variable [Variable]		3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Arrays and modifiers are not specified		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	Arrays and modifiers are not specified		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Variable Format *(Notes 1) Output only	Timer	.PT/.ET only	2	O	
	Counter	.PV/ .CV only	2	O	
	Date	.YR/ .MO/ .DAY only	2	O	
	Time	.HR/ .MIN/ .SEC only	2	O	
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	1	O	
	D_	Modifiers are not specified		1	O
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT/.ET only	2	O	
	C_	.PV/ .CV only	2	O	
	N_	.YR/ .MO/ .DAY only	2	O	
	J_	.HR/ .MIN/ .SEC only	2	O	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O		
Constant	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X	

◆ **Explanation of the R2I/R2IP Instructions**

The R2I/R2IP instructions convert real variables to integer variables. Specify the real variable or constant in S1 that you want to convert, and specify integer variable for the conversion output in D1. You can specify only a real variable for input in S1 and an integer variable for output in D1. Use the convert instruction when you want to use different variable types in a calculation or comparison.

Refer to the following for specifying a constant.

When operand S1 is a real constant



Operand S1	Real constant	0r0.11
Operand D1	Integer variable	OUT1

Note that specified arrays (entire arrays) cannot be converted.

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



Data_1	Real array size =	5
Result	Integer array size =	5
The operand specification in the left diagram results in an error.		



Data_1 [0]	Real array size =	5
Result [N]	Integer array size =	5
The operand specification in the left diagram operates normally.		

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

R2I



(1) When the positive transition instruction turns ON, the R2I instruction will be executed.

When the R2I instruction is executed, the result of the R2I conversion of Data_A is stored in D1.

When using a normally open instruction, the R2I instruction is always executed as long as the normally open instruction variable remains ON.

Program Example



R2IP



(1) The R2IP and R2I instructions have different ways of detecting when to execute. In the R2IP instruction, only the upward transition is detected and the R2IP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the R2IP instruction is executed only once (for 1 scan).

■ **R2F/R2FP (Real to Float Conversion)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
R2F (Real to Float Conversion - Level Sensitive)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
R2FP (Real to Float Conversion - positive transition)		Type Convert	3 to 7

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the R2F/R2FP instructions.

The actual number of steps in the R2F/R2FP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the R2F/R2FP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{Data_1 [0] = 2 \text{ steps}\} + \{Conversion \text{ result } [N] = 3 \text{ steps}\} + \{1 \text{ step}\} = 6 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the R2F/R2FP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	Real Variable		1	O
		Specify real variable [constant]		2	O
		Specify real variable [variable]		3	O
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
Date	.YR/ .MO/ .DAY only		—	X	
Time	.HR/ .MIN/ .SEC only		—	X	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	—	X
	D_	Modifiers are not specified	—	X
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	1	O
	T_	.PT/.ET only	—	X
	C_	.PV/ .CV only	—	X
	N_	.YR/ .MO/ .DAY only	—	X
J_	.HR/ .MIN/ .SEC only	—	X	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Constant	Integer	- 2147483648 to 2147483647	—	X
	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	X
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	1	O

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the R2F/R2FP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer *(Notes 1)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Float Variable		1	O
		Specify float variable[constant]		2	O
		Specify float variable[variable]		3	O
	Real	—		—	X
		Specify real variable [constant]		—	X
Specify real variable [variable]		—	X		

Continued

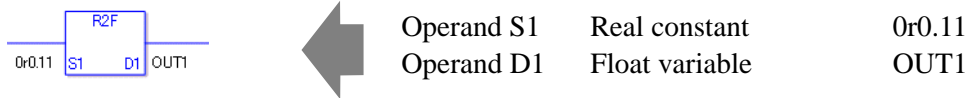
Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Variable Format *(Notes 1) Output only	Timer	.PT/.ET only	—	X	
	Counter	.PV/ .CV only	—	X	
	Date	.YR/ .MO/ .DAY only	—	X	
	Time	.HR/ .MIN/ .SEC only	—	X	
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	1	O	
	R_	—	—	X	
	T_	.PT/.ET only	—	X	
	C_	.PV/ .CV only	—	X	
	N_	.YR/ .MO/ .DAY only	—	X	
	J_	.HR/ .MIN/ .SEC only	—	X	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X		
Constant	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X	

◆ **Explanation of the R2F/R2FP Instructions**

The R2F/R2FP instructions convert real variables to float variables. Specify the real variable or constant in S1 that you want to convert, and specify float variable for the conversion output in D1. You can specify only a real variable for input in S1 and a float variable for output in D1. Use the convert instruction when you want to use different variable types in a calculation or comparison.

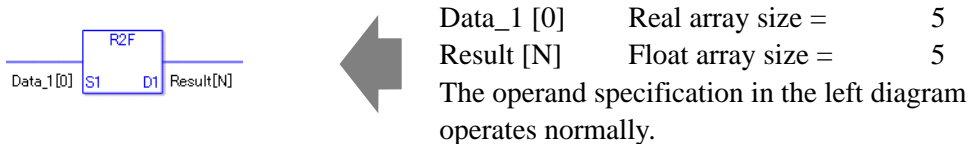
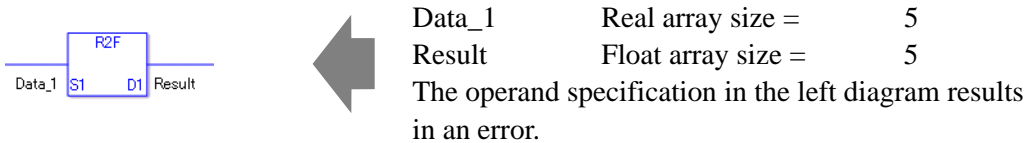
Refer to the following for specifying a constant.

When operand S1 is a real constant



Note that specified arrays (entire arrays) cannot be converted.

When operands S1 and D1 specify the entire array, an error will occur even if the specified variables are the same type.



◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

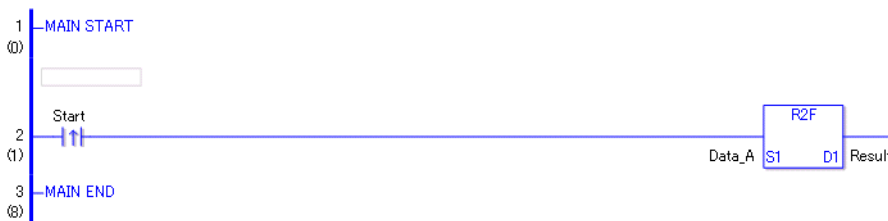
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

R2F



(1) When the positive transition instruction turns ON, the R2F instruction will be executed.

When the R2F instruction is executed, the result of the R2F conversion of Data_A is stored in D1.

When using a normally open instruction, the R2F instruction is always executed as long as the normally open instruction variable remains ON.

Program Example



R2FP



(1) The R2FP and R2F instructions have different ways of detecting when to execute. In the R2FP instruction, only the upward transition is detected and the R2FP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the R2FP instruction is executed only once (for 1 scan).

■ **H2S/H2SP (Time to Seconds)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
H2S (Time to Seconds Conversion - Level Sensitive)		Type Convert	3 to 5
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
H2SP (Time to Seconds Conversion - positive transition)		Type Convert	3 to 5

◆ **Operand Settings**

The following shows the configurable conditions for Operands (S1, D1) in the H2S/H2SP instructions.

The actual number of steps in the H2S/H2SP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the H2S/H2SP instructions

(For the number of steps in an operand, refer to the operand settings in the next section.)



$$\{ \text{Elapsed time} = 1 \text{ step} \} + \{ \text{Total seconds [0]} = 2 \text{ steps} \} + \{ 1 \text{ step} \} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the H2S/H2SP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	—		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
	Date	.YR/ .MO/ .DAY only		—	X
Time	Other than .HR / .MIN / .SEC		1	O	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

Instruction Descriptions

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT/.ET only	—	X	
	C_	.PV/ .CV only	—	X	
	N_	.YR/ .MO/ .DAY only	—	X	
	J_	Other than .HR / .MIN / .SEC	1	O	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X		
Constant	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X	

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the H2S/H2SP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer *(Notes 1)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant]		2	O
		Specify integer variable [Variable]		3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Arrays and modifiers are not specified		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	Arrays and modifiers are not specified		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Variable Format *(Notes 1) Output only	Timer	.PT/.ET only	2	O	
	Counter	.PV/ .CV only	2	O	
	Date	.YR/ .MO/ .DAY only	2	O	
	Time	.HR/ .MIN/ .SEC only	2	O	
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	1	O	
	D_	Modifiers are not specified		1	O
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT/.ET only	2	O	
	C_	.PV/ .CV only	2	O	
	N_	.YR/ .MO/ .DAY only	2	O	
	J_	.HR/ .MIN/ .SEC only	2	O	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O		
Constant	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X	

◆ **Explanation of the H2S/H2SP Instructions**

The H2S/H2SP instructions convert seconds in time variables to integer variables. Specify the time variable in S1 that you want to convert, and specify integer variable for the conversion output in D1. You can specify only a time variable for input in S1 and an integer variable for output in S2. Time variables cannot be configured in arrays. 0:30 will be converted to 1800 seconds and 14:00 will be converted to 50400 seconds.

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.

When the execution results in an error, the error code is stored in #L_CalcErrCode.

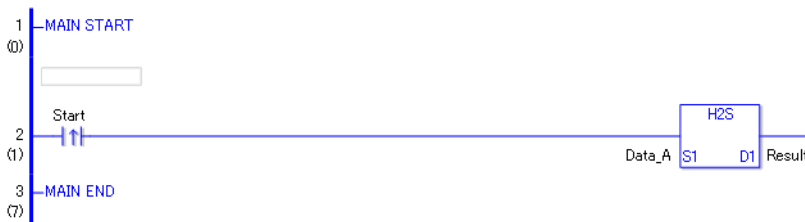
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

H2S



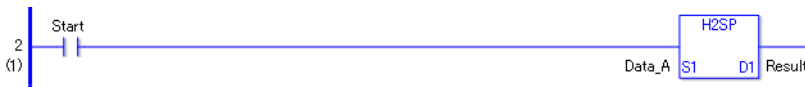
(1) When the positive transition instruction turns ON, the H2S instruction will be executed.

When the H2S instruction is executed, the result of the H2S conversion of Data_A is stored in D1.

When using a normally open instruction, the H2S instruction is always executed as long as the normally open instruction variable remains ON.

Program Example



H2SP



(1) The H2SP and H2S instructions have different ways of detecting when to execute. In the H2SP, only the upward transition is detected and the H2SP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the H2SP instruction is executed only once (for 1 scan).

■ **S2H/S2HP (Seconds to Time)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
S2H (Seconds to Time Conversion - Level Sensitive)		Type Convert	3 to 5
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
S2HP (Seconds to Time Conversion - positive transition)		Type Convert	3 to 5

◆ **Operand Settings**

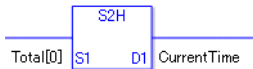
The following shows the configurable conditions for Operands (S1, D1) in the S2H/S2HP instructions.

The actual number of steps in the S2H/S2HP instructions depends on the specified operands. The following describes how to calculate the number of steps.

Number of steps in operand S1 + Number of steps in operand D1 + 1 = Total number of steps in one instruction

Example: Calculate the number of steps in the S2H/S2HP instructions

(For the number of steps in an operand, refer to the operand settings in the next page.)



$$\{ \text{Elapsed time} = 1 \text{ step} \} + \{ \text{Total seconds [0]} = 2 \text{ steps} \} + \{ 1 \text{ step} \} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following describes the specifiable content of Operand (S1) in the S2H/S2HP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	1	O	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	O	
Symbol	Bit	—	—	X	
	Word	—	1	O	
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer *(Notes 1)	Arrays and modifiers are not specified		1	O
		Specify integer variable[constant]		2	O
		Specify integer variable [Variable]		3	O
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	Arrays and modifiers are not specified		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	Arrays and modifiers are not specified		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
Variable Format *(Notes 1) Output only	Timer	.PT/.ET only	2	O
	Counter	.PV/ .CV only	2	O
	Date	.YR/ .MO/ .DAY only	2	O
	Time	.HR/ .MIN/ .SEC only	2	O
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O
Address Format	X_	—	—	X
	Y_	—	—	X
	M_	—	—	X
	I_	—	—	X
	Q_	—	1	O
	D_	Modifiers are not specified	1	O
		D_****.B/W[constant]	—	X
		D_****.B/W[address]	—	X
	F_	—	—	X
	R_	—	—	X
	T_	.PT/.ET only	2	O
	C_	.PV/ .CV only	2	O
	N_	.YR/ .MO/ .DAY only	2	O
	J_	.HR/ .MIN/ .SEC only	2	O
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	O	
Constant	Integer	– 2147483648 to 2147483647	—	X
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X

◆ **Operand Settings**

The following describes the specifiable content of Operand (D1) in the S2H/S2HP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
External Device Address	Bit	—	—	X	
	Word	Specify by words only (Example: [PLC1]D0000)	—	X	
Internal Address	Bit	—	—	X	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	X	
Symbol	Bit	—	—	X	
	Word	—	—	X	
Variable Format	Bit	Specify a bit	—	X	
		Specify bit array ([constant])	—	X	
		Specify bit array ([variable])	—	X	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	X
		Specify integer variable[constant]		—	X
		Specify integer variable [Variable]		—	X
		Specify integer variable[constant/variable] .B/W[constant/variable]		—	X
	Float	—		—	X
		Specify float variable[constant]		—	X
		Specify float variable[variable]		—	X
	Real	—		—	X
		Specify real variable [constant]		—	X
		Specify real variable [variable]		—	X
	Timer	.PT/.ET only		—	X
	Counter	.PV/ .CV only		—	X
	Date	.YR/ .MO/ .DAY only		—	X
Time	Other than .HR / .MIN / .SEC		1	O	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	X	

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		—	X
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT/.ET only	—	X	
	C_	.PV/ .CV only	—	X	
	N_	.YR/ .MO/ .DAY only	—	X	
	J_	Other than .HR / .MIN / .SEC	1	O	
U_	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X		
Constant	Integer	– 2147483648 to 2147483647	—	X	
	Float	±1.175494351e–38 to ±3.402823466e+38	—	X	
	Real	±2.2250738585072014e–308 to ±1.7976931348623158e+308	—	X	

◆ **Explanation of the S2H/S2HP Instructions**

The S2H/S2HP instructions convert integer variables to seconds in time variables. Specify the integer variable in S1 that you want to convert, and specify time variable for the conversion output in D1. You can specify only an integer variable for input in S1 and a time variable for output in D1. Time variables cannot be configured in arrays. 0:30 will be converted to 1800 seconds. 14:00 will be converted to 50400 seconds.

◆ **System Variables Indicating Execution Results**

When the execution result is 0, #L_CalcZero turns on.
 When the execution results in an error, the error code is stored in #L_CalcErrCode.

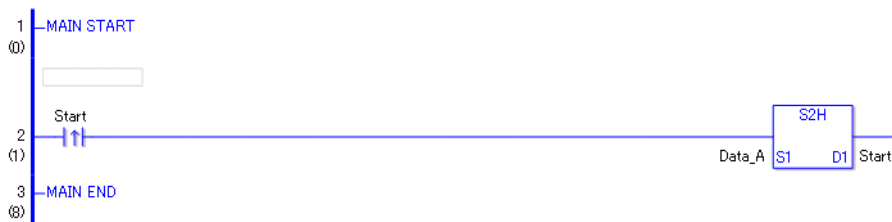
(Notes)

When checking the result using system variables, make sure the check takes place after the instruction has been executed.

When checking the state after multiple instructions have been executed, system variables will store the result only of the last processed instruction.

Program Example

S2H

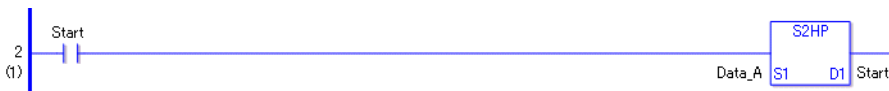


(1) When the positive transition instruction turns ON, the S2H instruction will be executed.
 When the S2H instruction is executed, the result of the S2H conversion of Data_A is stored in D1.

When using normally open instruction, the S2H instruction is always executed as long as the normally open instruction variable remains ON.

Program Example

S2HP



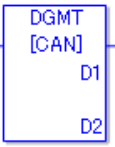



(1) The S2HP and S2H instructions have different ways of detecting when to execute. In the S2HP instruction, only the upward transition is detected and the S2HP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the S2HP instruction is executed only once (for 1 scan).

29.5.20 I/O Driver Instructions

■ SDOR, SDOW, DGMT, DGSL (CANopen Driver)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SDOR		Defined Node read object dictionary	9 to 21
SDOW		Defined Node write to object dictionary	9 to 21
DGMT		Read the state on the master.	5 to 9
DGSL		Read the state on the slave.	5 to 9

◆ **Operand Settings**

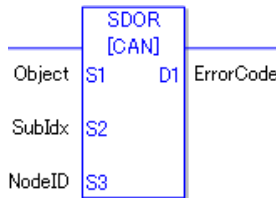
The following describes the specifiable content of operands S1 to S10, and D1 to D5 for I/O Driver Instructions.

The number of steps in I/O Driver instructions depends on the specification method and the number of operands used. The following describes how to calculate the number of steps.
 $3 + \text{Number of steps in operand S1} + \dots + \text{Number of steps in operand S10} + \text{Number of steps in operand D1} + \dots + \text{Number of steps in operand D5} = \text{Total number of steps in one instruction}$

NOTE • For more information regarding each operand, please refer to the following.
 ☞ "30.7.6 Using I/O Driver Instructions" (page 30-180)

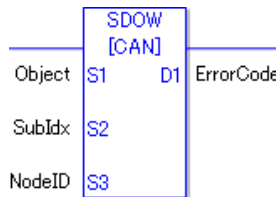
For example, Converting the number of steps in SDOR, SDOW, DGMT and DGSL
 (For the number of steps in an operand, refer to the operand settings on the next page.)

SDOR



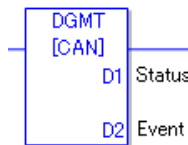
$3 \text{ Steps} + \{\text{Object} = 1 \text{ Step}\} + \{\text{SubIdx} = 1 \text{ Step}\} + \{\text{NodeID} = 1 \text{ Step}\} + \{\text{Length} = 1 \text{ Step}\} + \{\text{Offset} = 1 \text{ Step}\} + \{\text{ErrorCode} = 1 \text{ Step}\} = 9 \text{ Steps}$

SDOW



$3 \text{ Steps} + \{\text{Object} = 1 \text{ Step}\} + \{\text{SubIdx} = 1 \text{ Step}\} + \{\text{NodeID} = 1 \text{ Step}\} + \{\text{Length} = 1 \text{ Step}\} + \{\text{Offset} = 1 \text{ Step}\} + \{\text{ErrorCode} = 1 \text{ Step}\} = 9 \text{ Steps}$

DGMT



$3 \text{ Steps} + \{\text{Status} = 1 \text{ Step}\} + \{\text{Event} = 1 \text{ Step}\} = 5 \text{ Steps}$

DGSL



$3 \text{ Steps} + \{\text{NodeID} = 1 \text{ Step}\} + \{\text{Diagnostics} = 1 \text{ Step}\} = 5 \text{ Steps}$

The first three steps are the number of steps required by all I/O Driver Instructions. Be sure to add these three steps for I/O Driver Instructions.

◆ **Operand Settings**

The following describes the specifiable content of Operands (S1 to S10 and D1 to D5).

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X
External Device Address	Bit	—	—	X
	Integer	—	—	X
Internal Address	Bit	—	—	X
	Integer	—	—	X
Symbol	Bit	—	—	X
	Integer	—	—	X
Variable Format *(Note 1) Sx=Input and output not allowed Dx=Input and output not allowed	Bit	—	—	X
	Integer *(Note 1)	Arrays and modifiers are not specified	1	O
		Specify integer variable[constant]	2	O
		Specify integer variable [Variable]	3	O
		Specify integer variable B/W[variable] Specify integer variable B/W [constant]	—	X
	Float	—	—	X
	Real	—	—	X
	Timer	.PT/.ET only	—	X
	Counter	.PV/ .CV only	—	X
	Date	.YR/ .MO/ .DAY only	—	X
	Time	.HR/ .MIN/ .SEC only	—	X
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	X

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: O Not Possible: X	
Address Format	X_	—	—	X	
	Y_	—	—	X	
	M_	—	—	X	
	I_	—	—	X	
	Q_	—	—	X	
	D_	Modifiers are not specified		1	O
		D_****.B/W[constant]		—	X
		D_****.B/W[address]		—	X
	F_	—	—	X	
	R_	—	—	X	
	T_	.PT/.ET only	—	X	
C_	.PV/.CV only	—	X		
N_	.YR/.MO/.DAY only	—	X		
Address Format	J_	.HR/.MIN/.SEC only	—	X	
	U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	X	
Constant	Integer	-2147483648 to 2147483647	—	X	
	Float	+/-1.17549435138 e-38 to +/-3.402823466e+38	—	X	
	Real	+/-2.2250738585072014e-308 to +/-1.7976931348623158e+308	—	X	

◆ **Explanation of SDOR, SDOW, DGMT and DGMT Instructions**

- From the I/O Driver, the SDOR instruction runs the SDO command (read) defined in the source operand. After the I/O Driver completes the SDO command, execution results are set to the destination operand.
- From the I/O Driver, the SDOW instruction runs the SDO command (write) defined in the source operand. After the I/O Driver completes the SDO command, execution results are set to the destination operand.
- From the I/O Driver, the DGMT instruction reads the master diagnostic results. The I/O Driver then sets the read results to the destination operand.
- From the I/O Driver, the DGSL instruction reads the slave diagnostic results defined in the source operand. The I/O Driver then sets the read results to the destination operand.
- Instructions execute when they receive power, and pass power for one scan after the instruction completes execution.

NOTE

- For more information regarding SDOR, SDOW, DGMT, and DGSL instructions (CANopen driver), refer to the following.
☞ "30.7 Controlling External I/O by Using HTB" (page 30-146)
-

◆ Restrictions

- An error occurs if these instructions are not set up with the CANopen driver, or when operands are set up with invalid data types.
- You can use up to 15 I/O Driver instructions, which includes other I/O Drivers.
- These instructions can be used in the MAIN and SUB programs only. They cannot be used in INIT.
- Instructions executed only at startup, such as MOVP, cannot be supported. To execute instructions only at startup, use PT instructions.

