

30



Instruction List

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

30.1	Instruction List	30-2
30.2	Instruction Notation List.....	30-5
30.3	Operand-Configurable Addresses.....	30-28
30.4	The Number of Steps	30-32
30.5	Explanations of Each Instruction.....	30-33

30.1 Instruction List

The below table provides a list of instructions available for the logic program. All of the below instructions will apply to some available models. The instructions are divided into the following 8 categories: (1) Basic Instructions, (2) Timer Instructions, (3) Counter Instructions, (4) Read/Write Instructions, (5) Operation Instructions, (6) Function Instructions, (7) Comparison Instructions, and (8) Conversion Instructions.

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

Continued

Category		Instruction Name	Instruction
Operation Instruction	Arithmetic Operation	Add	ADD
		Subtract	SUB
		Multiplication	MUL
		Division	DIV
		Modulus	MOD
		Increment	INC
		Decrement	DEC
	Time	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	
Function Instruction	Operation	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	Exponential	EXP
		Logarithm	LN
		Log Base 10	LG10

Continued

Category		Instruction Name	Instruction
Compare Instruction	Arithmetic	=	EQ
		>	GT
		<	LT
		>=	GE
		<=	LE
		≠	NE
	Time	=	JEQ
		>	JGT
		<	JLT
		>=	JGE
		<=	JLE
		≠	JNE
	Date	=	NEQ
		>	NGT
		<	NLT
>=		NGE	
<=		NLE	
≠		NNE	
Convert Instruction	Data Convert	BCD Convert	BCD
		BIN Convert	BIN
		Encode	ENCO
		Decode	DECO
		Convert to Radian	RAD
		Convert Degree	DEG
		Scale	SCL
	Type Convert	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		

30.2 Instruction Notation List

This list shows the categorized instruction names and symbols.








IMPORTANT

- The number of steps in the instructions differs depending on the format (the use of modifiers) specified for the operands.
- For details on the number of steps, please refer to the relevant instructions.

30.2.1 Basic Instruction

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Ladder Symbol	
Basic	Bit	Normally Open	NO	1 to 5 Step	1	
		Normally Closed	NC	1 to 5 Step	1	
		Output Coil	OUT	1 to 5 Step	1	
		Negative Output Coil	OUTN	1 to 5 Step	1	
		Set Coil	SET	1 to 5 Step	1	
		Reset Coil	RST	1 to 5 Step	1	
	Pulse	Positive transition	PT	2 to 5 Step	1	
		Negative transition	NT	2 to 5 Step	1	
	Program Control	Jump	JMP	2 Step	—	
		Up Detection Jump	JMPP	2 to 5 Step	—	
		Subroutine Call	JSR	2 Step	—	
		Up Detection Subroutine Call	JSRP	2 Step	—	
		Return	RET	1 Step	—	

Continued



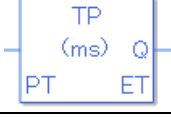


Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Ladder Symbol
Basic	Program Control	Repeat	FOR	2 to 4 Step	1	
			NEXT	1 Step	—	
		Inverse	INV	1 Step	—	
		Processing Completed	EXIT	1 Step	—	
		Power Bar Control	PBC	3 Step	2	
			PBR	2 Step	1	
		Logic Wait	LWA	2 Step	1	

(Note)

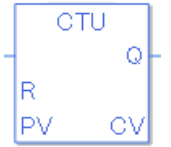
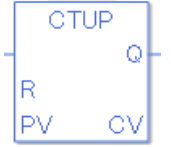
To use 1 Step, the number of steps must be fewer than the clear bit variables (M address) + 1536. If more than 1536 bit variables are created with clear bit variable settings, there will be 2 Steps.

Set up a dialog log in the save area settings.

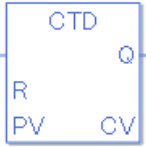
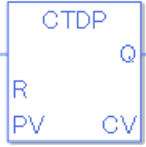
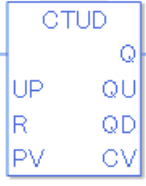
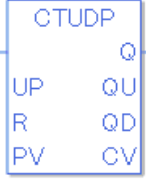
30.2.2 Timer Instruction

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



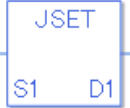

30.2.3 Counter Instruction

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol
Counter	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 the Instructions	Number of Operands	Determination of Input	Ladder Symbol
Counter	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	

30.2.4 R/W Instructions

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol
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	

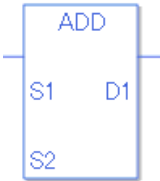
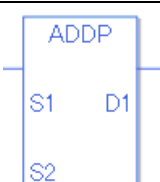
Continued

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol
Read/Write	Date	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	







30.2.5 Arithmetic Operation

IMPORTANT


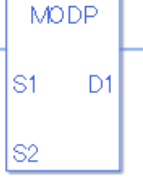




- The actual number of steps depends on the format of the operands in the basic instructions. The number of steps shown in the following table applies when variables are not modified, the number of array elements is 0, and Keep is disabled. For details on the number of steps, please see the relevant instructions.

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol
Operation	Arithmetic	Add	ADD	4 to 13 Step	3	Level	
			ADDP	4 to 13 Step	3	Up Edge	

Continued

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol
Operation	Arithmetic	Subtract	SUB	4 to 13 Step	3	Level	
			SUBP	4 to 13 Step	3	Up Edge	
		Multiplication	MUL	4 to 13 Step	3	Level	
			MULP	4 to 13 Step	3	Up Edge	
		Division	DIV	4 to 13 Step	3	Level	
			DIVP	4 to 13 Step	3	Up Edge	

Continued

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol
Operation	Arithmetic	Modulus	MOD	4 to 13 Step	3	Level	
			MODP	4 to 13 Step	3	Up Edge	
		Increment	INC	2 to 4 Step	1	Level	
			INCP	2 to 4 Step	1	Up Edge	
		Decrement	DEC	2 to 4 Step	1	Level	
			DECP	2 to 4 Step	1	Up Edge	

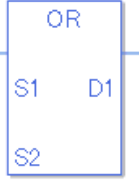





30.2.6 Time Operation

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol
Operation	Time	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	



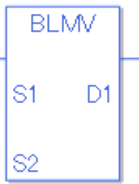
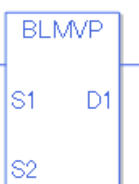
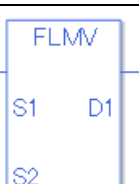
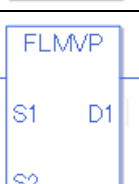
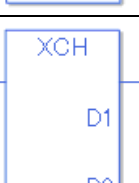
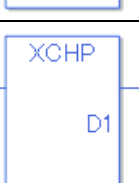
30.2.7 Logical Operation

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

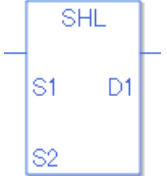
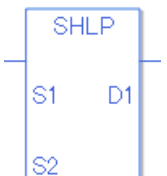
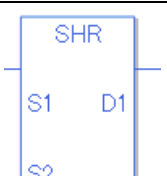
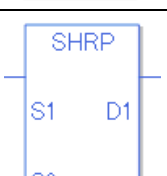
Continued

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


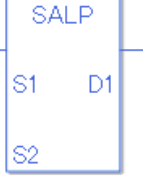
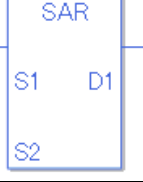

30.2.8 Transfer Instructions

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol	
Operation	Transfer	Transfer	MOV	3 to 9 Step	2	Level	
			MOVP	3 to 9 Step	2	Up Edge	
		Block Transfer (Block Transfer)	BLMV	6 to 10 Step	3	Level	
			BLMVP	6 to 10 Step	3	Up Edge	
		Multipoint Transfer (Fill Transfer)	FLMV	4 to 10 Step	3	Level	
			FLMVP	4 to 10 Step	3	Up Edge	
		Exchange	XCH	3 to 7 Step	2	Level	
			XCHP	3 to 7 Step	2	Up Edge	



30.2.9 Shift Instructions

Category		Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol
Operation	Shift	Shift Left	SHL	4 to 10 Step	3	Level	
			SHLP	4 to 10 Step	3	Up Edge	
		Shift Right	SHR	4 to 10 Step	3	Level	
			SHRP	4 to 10 Step	3	Up Edge	

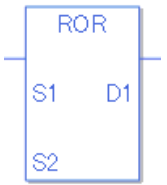
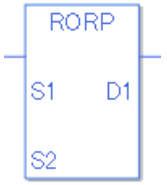
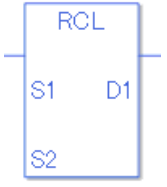
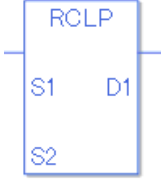
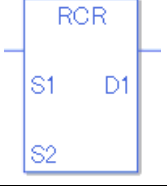
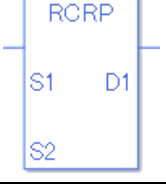
Continued

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










30.2.10 Rotation Instructions

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







Continued

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









30.2.11 Function Operations

Category	Instruction Name	Pro EX Instruction Notation	Number of Steps in the Instructions	Number of Operands	Determination of Input	Ladder Symbol
Function Operation	Sum	SUM	4 to 10 Step	3	Level	
		SUMP	4 to 10 Step	3	Up Edge	
	Average	AVE	4 to 10 Step	3	Level	
		AVEP	4 to 10 Step	3	Up Edge	
	Square Root	SQRT	3 to 7 Step	2	Level	
		SQ RTP	3 to 7 Step	2	Up Edge	
	Bit Count	BCNT	3 to 9 Step	2	Level	
		BCNTP	3 to 9 Step	2	Up Edge	
	PID	PID	10 to 18 Step	5	Level	







30.2.12 Trigonometric Function

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

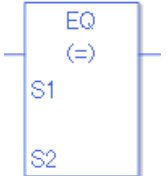
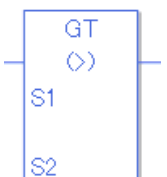
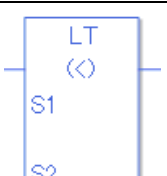
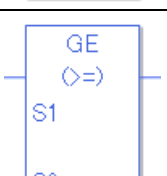
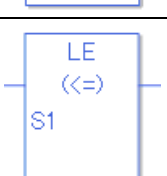
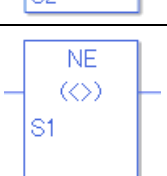
Continued

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

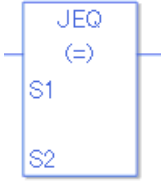
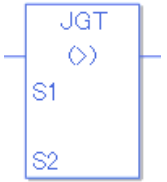
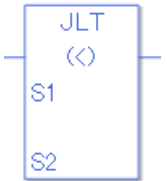
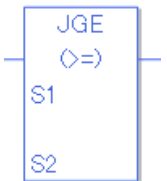
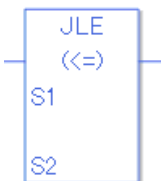
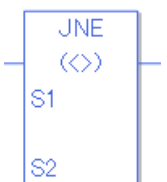
30.2.13 Other Functions

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

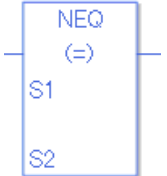
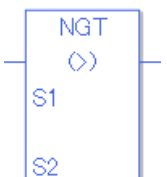
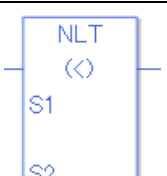
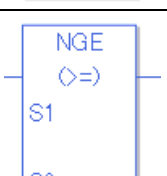
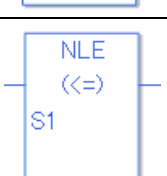
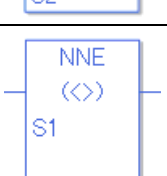
30.2.14 Arithmetic Compare

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











30.2.15 Time Compare

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





30.2.16 Date Compare

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







30.2.17 Data Conversion Instructions

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










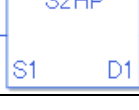
Continued

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

30.2.18 Type Conversion Instructions

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

Continued

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

30.3 Operand-Configurable Addresses

The instructions for the symbol variables, connection device addresses, and constants that are configurable for an operand are outlined.

The configurable contents differ depending on the instructions. Please see the description for each instruction.

30.3.1 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

30.3.2 Symbol

This function allows you to change the connection device address to a name that is easy for users to understand. Be sure to link the connection device to the arbitrary name.

e.g. To assign the arbitrary name "RUN" to PLC "X0000" of Mitsubishi Electronic Corporation, you must 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.

30.3.3 LS Address

It is the address of the internal memory area of the GP unit. Please be noted that the specification method differs 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 Setting	Bit	[#MEMLINK]010000	Bit Specifications for GP Internal Memory
	Word	[#MEMLINK]0100	Word Specifications for GP Internal Memory

30.3.4 User 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
User Area	Bit	[#INTERNAL]USR0010000	Bit Specifications for GP Internal Memory
	Word	[#INTERNAL]USR00100	Word Specifications for GP Internal Memory

30.3.5 System Variable

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

Name	Type	example	Description
System Variable	Bit	#L_Clock100ms	Bit Type for GP System Variables
	Integer	#L_ScanTime	Integer Types for GP System Variables

30.3.6 Variable

The variables can be used with all GP models. You can use the variables as you like, since there is no need to pay attention to the addresses. The variables have modifiers (*1) and sequences (*2). Using modifiers allows bit access or byte access for integer variables.

Name	Type	example	Description
Variable	Bit	Arbitrary Name	Bit-type variable. Sequence specifications are allowed.
	Integer	Same as above	Integer-type variable. Sequence and modifier specifications are allowed.
	Float	Same as above	32 bit-float variable. Sequence specifications are allowed.
	Real	Same as above	64-bit real variable. Sequence specifications are allowed.
	Timer	Same as above	Timer variable. Structure variable.
	Counter	Same as above	Counter variable. Structure variable.
	Date	Same as above	Date variable. Structure variable.
	Time	Same as above	Time variable. Structure variable.
PID	Same as above	PID variable. Structure variable.	

*1 As modifiers, you can use 3 specifications; bit specification, byte specification, and word specification. Modifiers can only use integer variables.

Specification method: bit specification to variable name.X[0], byte specification to variable name.B[0], word specification to variable name.W[0]

*2 Sequential memories can be specified. Only these configurable variables can be used: bit, integer, float, and real. Specification method: variable name[10]

*3 Variable assemblies become a structure. The variables defined as structure variables are: timer, counter, time, date, and PID.

■ Structure Variable

Timer Variable

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

Counter Variable

Counter Variable	Variable 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	Turns ON while counting up.
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	Counter setting value.
Variable Name.CV	Integer Variable	Current value on the counter.

Time Variable

Time Variable	Variable Settings	Description
Variable Name.HR	Integer Variable	BCD is used for hours indicating the time.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.SEC	Integer Variable	Seconds are input in BCD.

Date Variable

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

PID Variable

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

30.3.7 Logic Device when using the Address Format

When logic is set for the address format, the devices below become available.

Name	Type	Name	Description
Logic Address	Bit	X_/Y_/M_	Bit-type logic address.
	Integer	D_/I_/Q_	Word-type logic address. Same as for variables, modifiers can be used.
	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 variables.
	Counter	C_	Counter-type logic address. It is a structure, the same as variables.
	Date	N_	Date-type logic address. It is a structure, the same as variables.
	Time	J_	Time-type logic address. It is a structure, the same as variables.
PID	U_	PID-type logic address. It is a structure, the same as variables.	

30.4 The Number of Steps

The conversion of the number of steps is described. (For details on the number of steps in the instructions, please refer to the relevant instructions.)

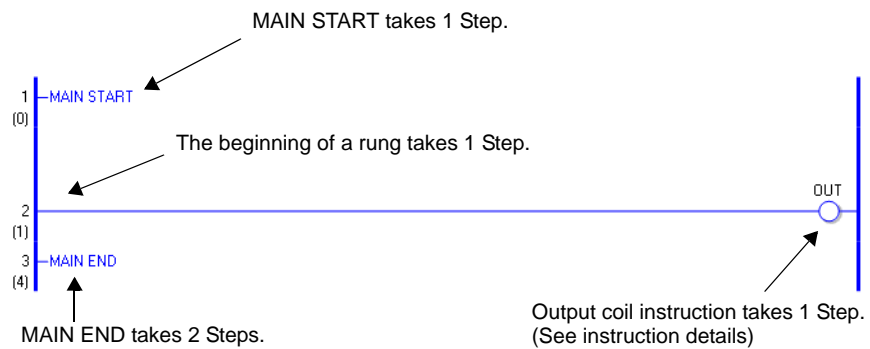
The following program uses only the output coil OUT which is always ON.

Definition of variable OUT

Variable Name : OUT

Keep/Clear Setting : Clear

Array Elements : None



There are a total number of 5 Steps .


Since 1-Step instructions are optimized by saving and error checking, the actual number of steps may differ from the tally indicated under each rung number.


30.5 Explanations of Each Instruction

30.5.1 Bit Instructions

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

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	No. of Instruction Steps
NO (Normally Open)	S1 	Input	1 to 5

Ladder Instruction Name	Ladder Symbol	Feature	No. of Instruction Steps
NC (Normally Closed)	S1 	Input	1 to 5

◆ Operand Settings

The following table lists the configurable conditions for Operand (S1).

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
External Device Address	Bit	—	2	○
	Word	Specify a bit in the word. (Example: [PLC1]D0000.00)	3	○
Internal Address	Bit	—	2	○
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	3	○
Symbol	Bit	—	2	○
	Word	—	—	×

Continued

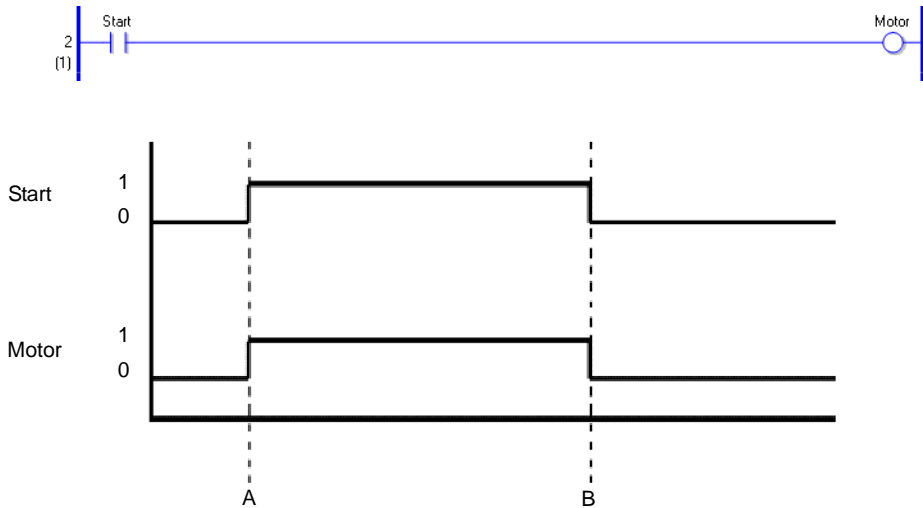
Explanations of Each Instruction

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Variable Format	Bit	Arrays are not specified. Specify inputs, outputs, or up to 1536 Clear items.	1	○
		Arrays are not specified. Keep items or more than 1536 Clear items.	2	○
		Specify bit array ([constant])	3	○
		Specify bit array ([variable])	4	○
	Integer	Arrays and modifiers are not specified	—	×
		Specify Integer Variable .X[constant]	3	○
		Specify Integer Variable .X[variable]	4	○
		Specify integer variable [constant/variable] .X [constant/variable]	5	○
	Float	—	—	×
	Real	—	—	×
	Timer	.Q / .TI / .R only	3	○
	Counter	.R / .UP / .QU / .QD / .Q only	3	○
	Date	—	—	×
	Time	—	—	×
PID	.Q / .UO / .TO / .PF / .IF only	3	○	
Address Format	X_	—	1	○
	Y_	—	1	○
	M_	Within the range of Clear items (M_0000 to M_1535)	1	○
	I_	—	—	×
	Q_	—	—	×
	D_	Modifiers are not specified	—	×
		D_****.X [constant]	3	○
		D_****.X [address]	4	○
	F_	—	—	×
	R_	—	—	×
	T_	.Q / .TI / .R only	3	○
	C_	.R / .UP / .QU / .QD / .Q only	3	○
	N_	—	—	×
	J_	—	—	×
U_	.Q / .UO / .TO / .PF / .IF only	3	○	

◆ **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



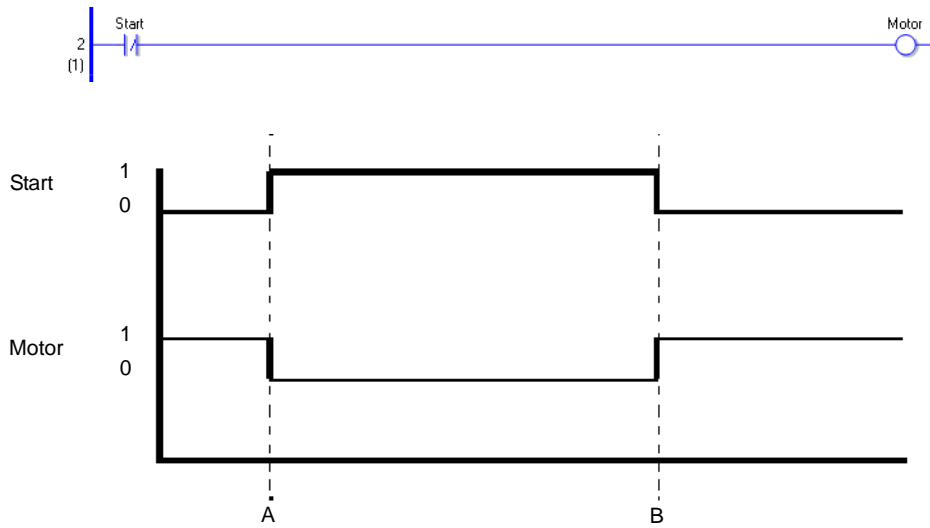
Point A When the bit variable Start turns ON, the NO instruction closes the contacts and the bit variable Motor turns ON.

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 an 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 Keep.
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	No. of Instruction Steps
OUT (Output Coil)	D1 	Output	1 to 5
Ladder Instruction Name	Ladder Symbol	Feature	No. of Instruction Steps
OUTN (Negative Output Coil)	D1 	Output	1 to 5

◆ Operand Settings

The following table lists the configurable conditions for Operand (D1).

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
External Device Address	Bit	—	2	○
	Word	Specify a bit in the word. (Example: [PLC1]D0000.00)	3	○
Internal Address	Bit	—	2	○
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	3	○
Symbol	Bit	—	2	○
	Word	—	—	×
Variable Format	Bit	Arrays are not specified. Up to 1536 Outputs set to Clear	1	○
		Arrays are not specified. Keep items or more than 1536 Clear items.	2	○
		Specify bit array ([constant])	3	○
		Specify bit array ([variable])	4	○
	Integer	Arrays and modifiers are not specified	—	×
		Specify Integer Variable .X[constant]	3	○
		Specify Integer Variable .X[variable]	4	○
		Specify integer variable [constant/variable] .X [constant/variable]	5	○
	Float	—	—	×
	Real	—	—	×
	Timer	.Q / .TI / .R only	3	○
	Counter	.R / .UP / .QU / .QD / .Q only	3	○
	Date	—	—	×
Time	—	—	×	
PID	.Q / .UO / .TO / .PF / .IF only	3	○	

Continued

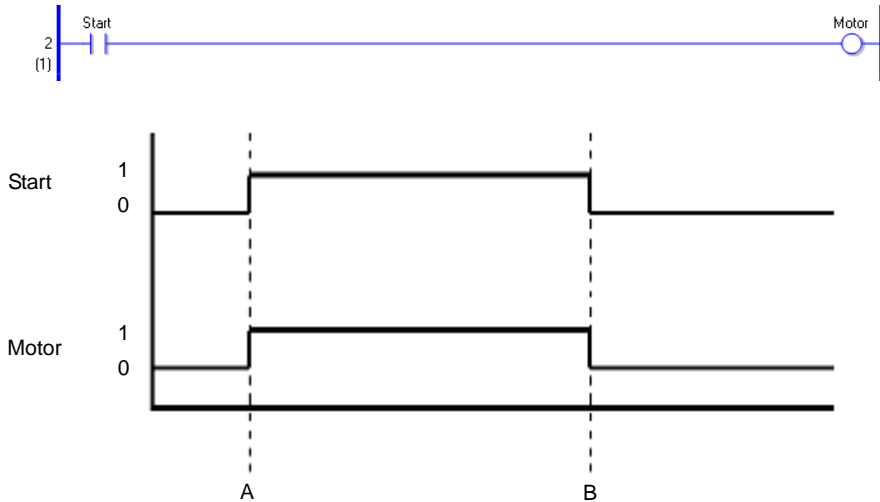
Explanations of Each Instruction

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	1	○
	M_	Specify within the range of Clear items (M_0000 to M_1535).	1	○
	I_	—	—	×
	Q_	—	—	×
	D_	Modifiers are not specified	—	×
		D_****.X [constant]	3	○
		D_****.X [address]	4	○
	F_	—	—	×
	R_	—	—	×
	T_	.Q / .TI / .R only	3	○
	C_	.R / .UP / .QU / .QD / .Q only	3	○
	N_	—	—	×
J_	—	—	×	
U_	.Q / .UO / .TO / .PF / .IF only	3	○	

◆ **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



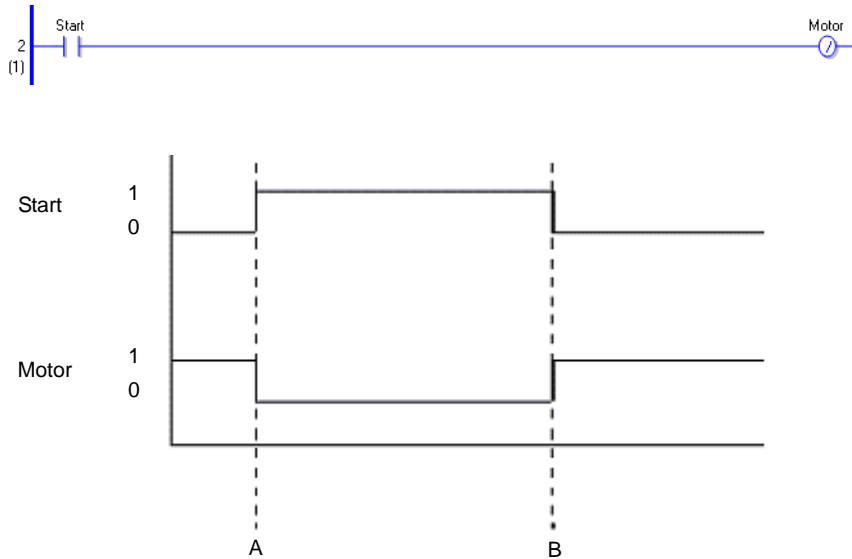
Point A When the bit variable Start turns ON, the bit variable Motor of the OUT instruction turns ON.

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



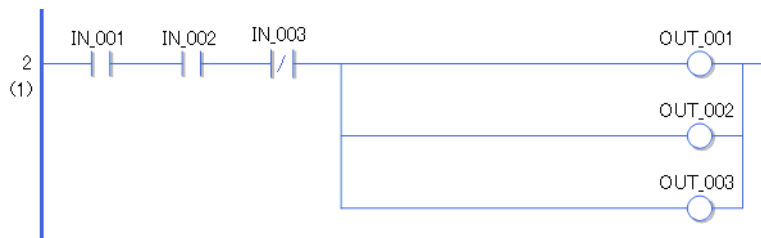
Point A When the bit variable Start turns ON, the bit variable Motor of the OUTN instruction turns OFF.

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 Keep.

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	No. of Instruction Steps
SET (Set Coil)	D1 	Output	1 to 5
Ladder Instruction Name	Ladder Symbol	Feature	No. of Instruction Steps
RST (Reset Coil)	D1 	Output	1 to 5

◆ Operand Settings

The following table lists the configurable conditions for Operand (D1).

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
External Device Address	Bit	—	2	○
	Word	Specify a bit in the word. (Example: [PLC1]D0000.00)	3	○
Internal Address	Bit	—	2	○
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	3	○
Symbol	Bit	—	2	○
	Word	—	—	×
Variable Format	Bit	Arrays are not specified. Up to 1536 Outputs set to Clear	1	○
		Arrays are not specified. Keep items or more than 1536 Clear items.	2	○
		Specify bit array ([constant])	3	○
		Specify bit array ([variable])	4	○
	Integer	Arrays and modifiers are not specified	—	×
		Specify Integer Variable .X [constant]	3	○
		Specify Integer Variable .X[variable]	4	○
		Specify Integer Variable [constant/variable] .X [constant/variable]	5	○
	Float	—	—	×
	Real	—	—	×
	Timer	.Q / .TI / .R only	3	○
	Counter	.R / .UP / .QU / .QD / .Q only	3	○
	Date	—	—	×
	Time	—	—	×
PID	.Q / .UO / .TO / .PF / .IF only	3	○	

Continued

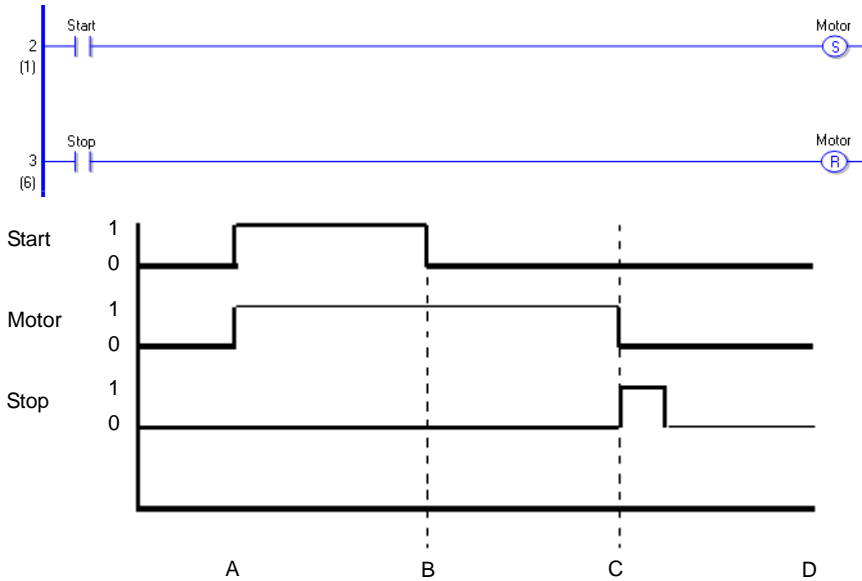
Explanations of Each Instruction

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	1	○
	M_	Specify within the range of Clear items (M_0000 to M_1535).	1	○
	I_	—	—	×
	Q_	—	—	×
	D_	Modifiers are not specified	—	×
		D_****.X [constant]	3	○
		D_****.X [address]	4	○
	F_	—	—	×
	R_	—	—	×
	T_	.Q / .TI / .R only	3	○
	C_	.R / .UP / .QU / .QD / .Q only	3	○
	N_	—	—	×
J_	—	—	×	
U_	.Q / .UO / .TO / .PF / .IF only	3	○	

◆ **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.

30.5.2 Pulse Instruction

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

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	No. of Instruction Steps
PT (Positive Transition)	S1 	Input	2 to 5
Ladder Instruction Name	Ladder Symbol	Feature	No. of Instruction Steps
NT (Negative Transition)	S1 	Input	2 to 5

◆ Operand Settings

The following table lists the configurable conditions for Operand (S1).

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	2	○	
	Word	Specify a bit in the word. (Example: [PLC1]D0000.00)	3	○	
Internal Address	Bit	—	2	○	
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	3	○	
Symbol	Bit	—	2	○	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	2	○	
		Specify bit array ([constant])	3	○	
		Specify bit array ([variable])	4	○	
	Integer	Arrays and modifiers are not specified		—	×
		Integer	Specify Integer Variable .X[constant]	3	○
			Specify Integer Variable .X[variable]	4	○
			Specify Integer Variable [constant/variable] .X [constant/variable]	5	○
	Float	—	—	×	
	Real	—	—	×	
	Timer	.Q / .TI / .R only	3	○	
	Counter	.R / .UP / .QU / .QD / .Q only	3	○	
	Date	—	—	×	
	Time	—	—	×	
PID	.Q / .UO / .TO / .PF / .IF only	3	○		

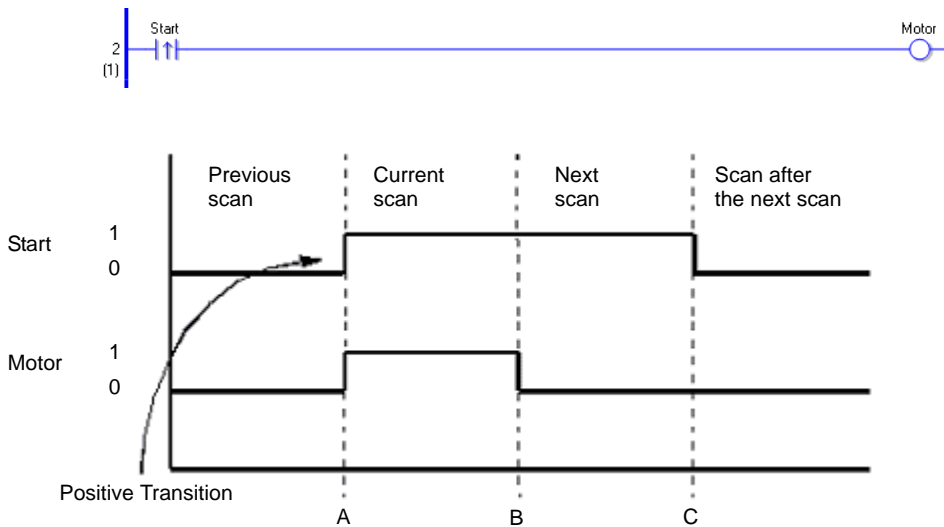
Continued

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×	
Address Format	X_	—	2	○	
	Y_	—	2	○	
	M_	—	2	○	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified		—	×
		D_****.X [constant]		3	○
		D_****.X [address]		4	○
	F_	—	—	×	
	R_	—	—	×	
	T_	.Q / .TI / .R only		3	○
	C_	.R / .UP / .QU / .QD / .Q only		3	○
	N_	—	—	×	
	J_	—	—	×	
U_	.Q / .UO / .TO / .PF / .IF only		3	○	

◆ **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

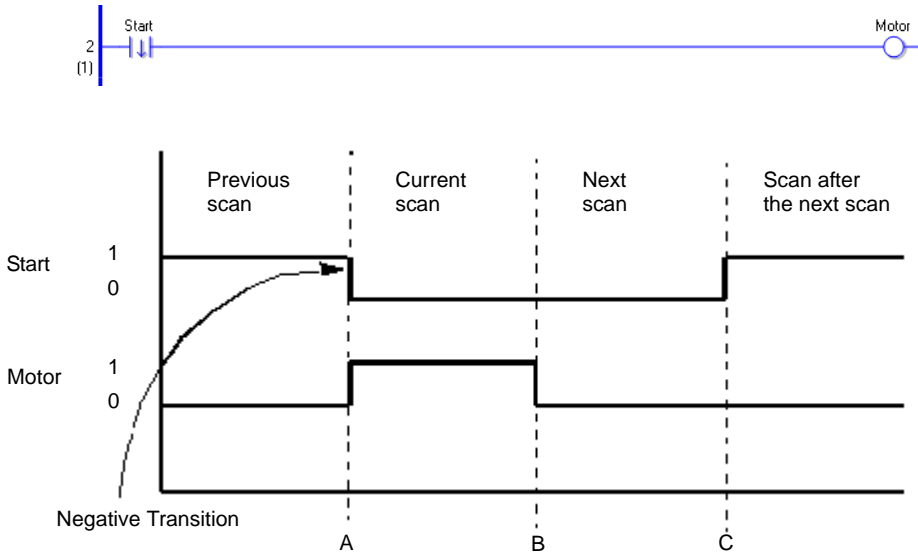


- Point A The variable (Start) turns ON, and then the variable motor turns ON.
- Point B After a scan is executed one time, the variable (Motor) turns OFF.
- Point C The variable (Motor) remains OFF because the upward transition of the variable (Start) is not detected.

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

- When an NT instruction is executed, if the variable that was ON during the previous scan turns 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 turns OFF.
- Point C The variable (Motor) remains OFF because the upward transition of the variable (Start) is not detected.

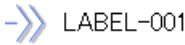
(Supplementary)

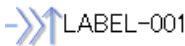
For the positive transition and negative transition instruction operands, you must pay attention when performing indirect addressing to each element, especially when an element is specifying an array or bit using variables. The variable in the operand of the previous execution is compared with the variable 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.

30.5.3 Program Control

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

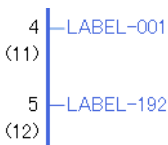
Symbols and Features

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.

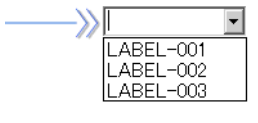
◆ Specifying Labels



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.

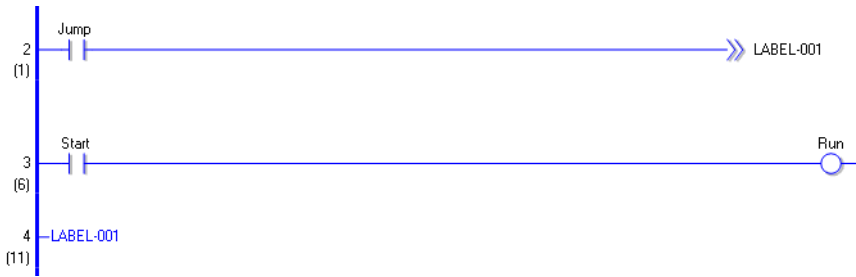


Only labels used in the program are displayed. The same label names cannot be used on the INIT, MAIN, and SUB screens.

When you execute a JMP instruction, the program jumps 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

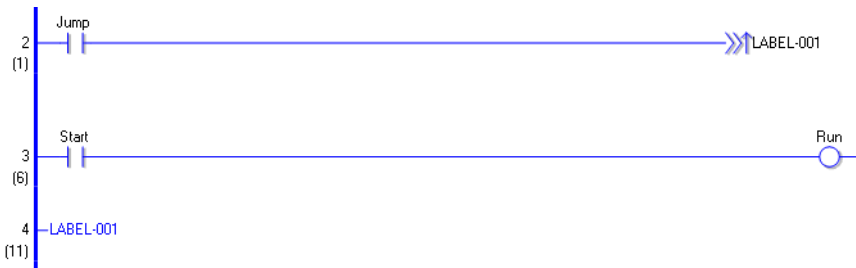
JMP



When the NO variable (Jump) turns ON, the JMP instruction is executed and the program jumps to the fourth rung set up 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

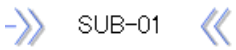
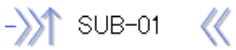
JMPP



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

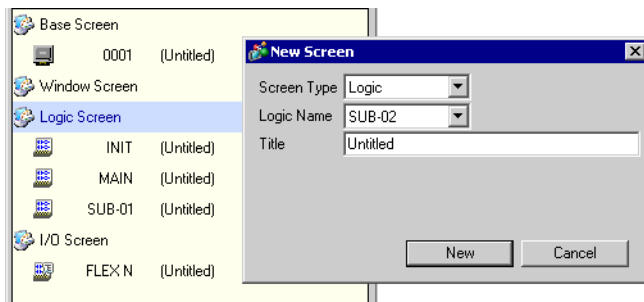
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JSR (Subroutine Call)	 SUB-01	Control	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JSRP (Positive Transition Subroutine Call)	 SUB-01	Control	2

Using JSR instructions you can specify up to 32 subroutines.

To specify the subroutine destination, first create a subroutine screen. If a subroutine screen has not been created, the subroutine destination cannot be specified. You can only specify subroutine screens as the subroutine destination.

◆ Specifying Subroutines

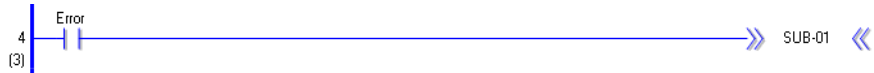
To create a subroutine screen, on the [Screen List Window] select [New Screen], or on the [Screen] menu click [New Screen].



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 turns ON to indicate a problem, the JSR instruction executes. The JSR instruction jumps to the subroutine screen “SUB-01” and executes the program. When “SUB-01” ends, the program returns to the rung after the JSR instruction and continues 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 a JSRP instruction is 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.

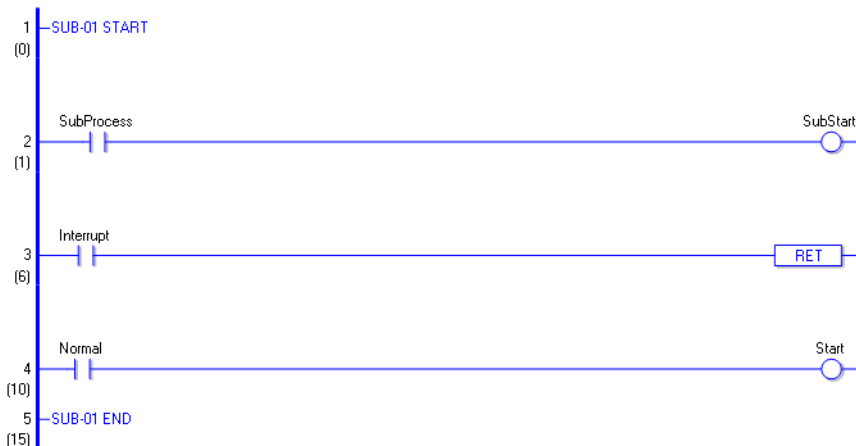
Use RET instructions to interrupt the subroutine and return to the MAIN program.

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	No. of Instruction Steps	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	2	○	
Internal Address	Bit	—	—	×	
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	2	○	
Symbol	Bit	—	—	×	
	Word	—	2	○	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer	Arrays and modifiers are not specified		2	○
		Integer	Specify integer variable [constant]	3	○
			Specify integer variable [variable]	4	○
			Specify Integer Variable [constant/variable] .X [constant/variable]	—	×
	Float	—	—	×	
	Real	—	—	×	
	Timer	.PT / .ET only	2	○	
	Counter	.PV / .CV only	2	○	
	Date	.YR / .MO / .DAY only	2	○	
	Time	.HR / .MIN / .SEC only	2	○	
PID	.KP / .TR / .TD / .PA / .BA / .ST only	2	○		

Continued

Explanations of Each Instruction

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	2	○
	Q_	—	2	○
	D_	Modifiers are not specified	2	○
		D_****.X [constant]	—	×
		D_****.X [address]	—	×
	F_	—	—	×
	R_	—	—	×
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY only	2	○
	J_	.HR/.MIN/.SEC only	2	○
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Constant	—	0 to 2147483647	2	○

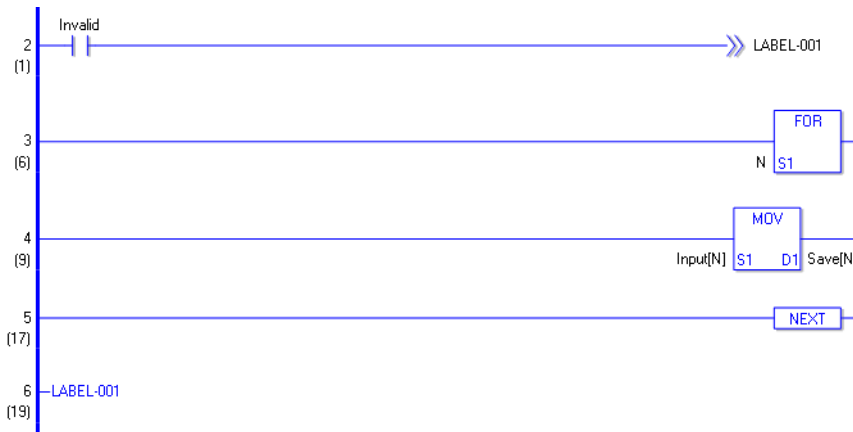
◆ **Explanation of FOR and NEXT Instructions**

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 is 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 as FOR and NEXT instructions.
(You cannot add conditions in the same rung as FOR and 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.

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

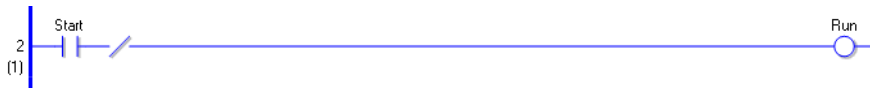
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
INV (Invert)	- / -	Control	1

◆ 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)	- [EXIT] -	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 turns ON, the EXIT instruction at the end of the rung is run. 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 table lists the configurable conditions for Operands (S1) and (D1) in the PBC instruction.

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Bit specifications (D1 operand only)	3	○	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer	Integer	Arrays and modifiers are not specified	—	×
			Specify integer variable [constant]	—	×
			Specify integer variable [variable]	—	×
			Specify integer variable [constant/variable] .X [constant/variable]	—	×
	Float	—	—	×	
	Real	—	—	×	
	Timer	.PT/.ET only	—	×	
	Counter	.PV/.CV only	—	×	
	Date	.YR/.MO/.DAY only	—	×	
	Time	.HR/.MIN/.SEC only	—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		

Continued

Explanations of Each Instruction

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	(D1 operand only)	3	○
	M_	(D1 operand only)	3	○
	I_	—	—	×
	Q_	—	—	×
	D_	Modifiers are not specified	—	×
		D_****.X [constant]	—	×
		D_****.X [address]	—	×
	F_	—	—	×
	R_	—	—	×
	T_	.PT/.ET only	—	×
	C_	.PV/.CV only	—	×
	N_	.YR/.MO/.DAY only	—	×
	J_	.HR/.MIN/.SEC only	—	×
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×	
Constant	—	0 to 7 (S1 operand only)	3	○

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in the PBR instruction.

Name	Type	Condition	Number of Steps	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify a bit in the word. (Example: [PLC1]D0000.00)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer	Arrays and modifiers are not specified		—	×
		Specify Integer Variable .X [constant]	—	×	
		Specify Integer Variable .X [variable]	—	×	
		Specify integer variable [constant/variable] .X [constant/variable]	—	×	
	Float	—	—	×	
	Real	—	—	×	
	Timer	.Q / .TI / .R only	—	×	
	Counter	.R / .UP / .QU / .QD / .Q only	—	×	
	Date	—	—	×	
	Time	—	—	×	
PID	.Q / .UO / .TO / .PF / .IF only	—	×		
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified		—	×
		D_****.X [constant]		—	×
		D_****.X [address]		—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.Q / .TI / .R only	—	×	
	C_	.R / .UP / .QU / .QD / .Q only	—	×	
	N_	—	—	×	
J_	—	—	×		
U_	.Q / .UO / .TO / .PF / .IF only	—	×		
Constant	—	0 to 7 (S1 operand only)	2	○	

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

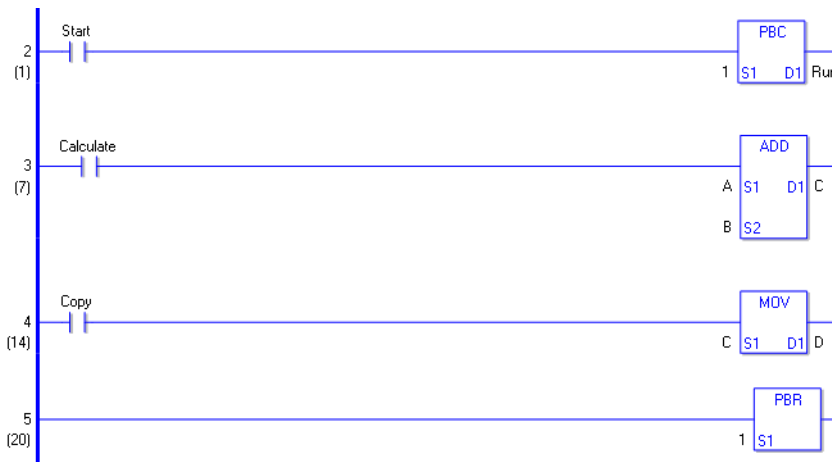
When a PBC instruction is executed, the program between PBC and PBR executes ON processing. PBC and PBR instructions can be used only in MAIN. They cannot be used in other parts of the program.

When the PBC instruction is ON, the bit variable in D1 turns 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.

Operand S1 of PBC and PBR instructions specifies the nesting level.

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.

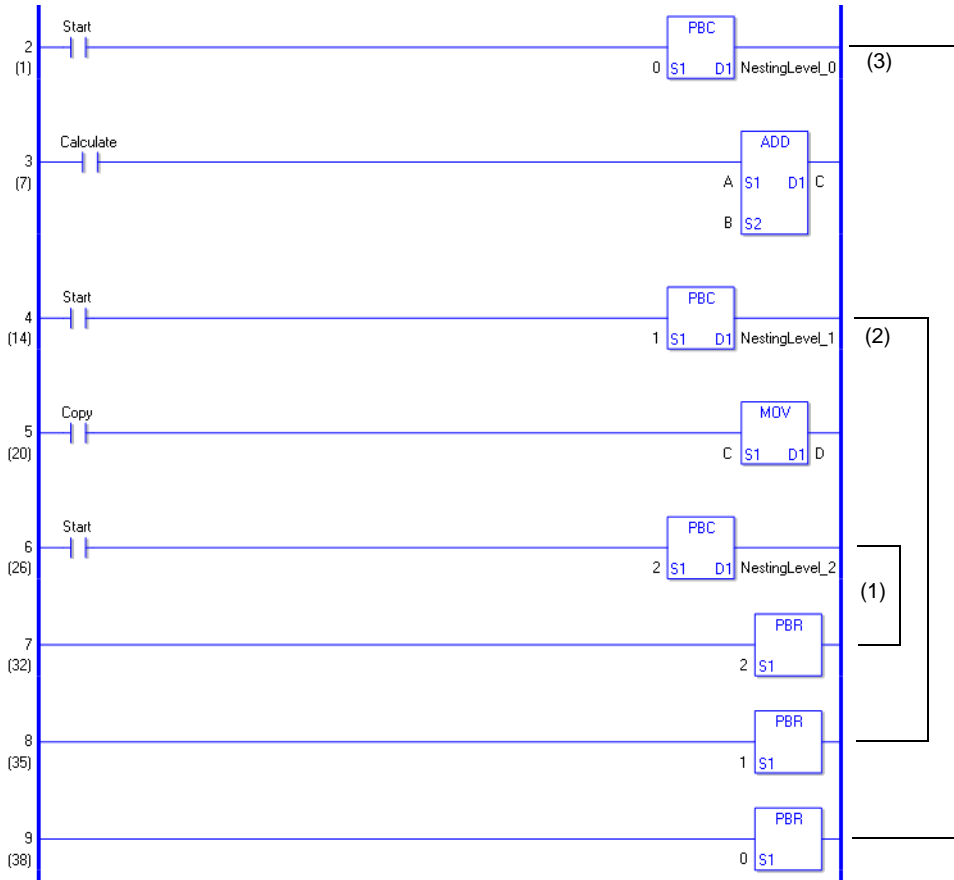
The MOV instruction is executed when the normally open instruction in the fourth rung turns ON.

◆ **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 nesting PBR 5 without before 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 table lists the specifiable contents of operand (S1).

Name	Type	Condition	Number of Steps	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify a bit in the word. (Example: [#INTERNAL]LS000000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify Integer Variable [constant/variable] .X [constant/variable]		—	×
	Float	—	—	×	
	Real	—	—	×	
	Timer	.PT/.ET only	—	×	
	Counter	.PV/.CV only	—	×	
	Date	.YR/.MO/.DAY only	—	×	
	Time	.HR/.MIN/.SEC only	—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		

Continued

Explanations of Each Instruction

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

◆ **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.

Setting range: 1 to 10 ms

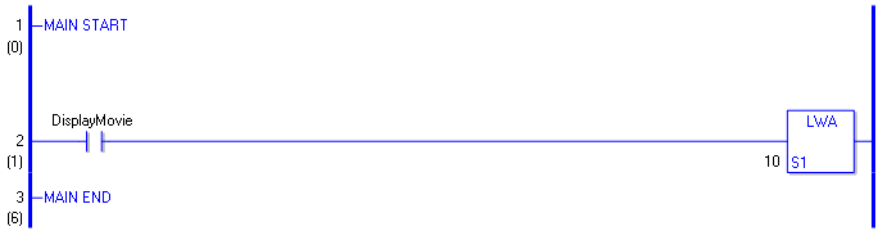
Notes

- (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.

Restrictions on use

- (1) Only one LWA instruction can be placed in one rung.
- (2) An LWA instruction must be the last instruction on the rung, just to the left of the right power bar.
- (3) An LWA instruction can be used only in MAIN and SUB. It cannot be used in INIT.

Program example



- (1) When the bit variable turns ON, the LWA instruction is 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.

30.5.4 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

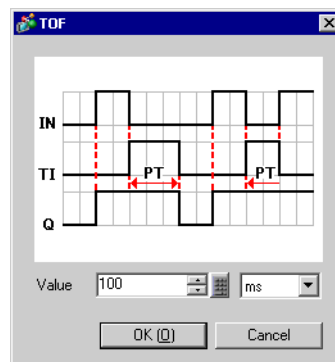
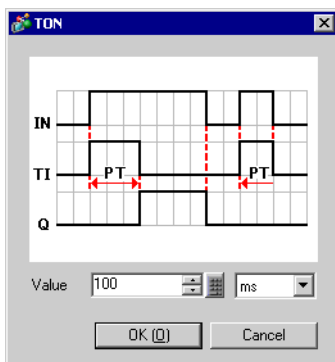
Timer variables used in TON and TOF instructions are structure variables. The following table lists the internal structures.

Timer Variable

Timer Variable	Variable Settings	Description
Variable Name.TI	Bit Variable	Turns ON when the timer begins.
Variable Name.Q	Bit Variable	Turns ON upon completion of the timer.
Variable Name.PT	Integer Variable	Preset Time (32 bits)
Variable Name.ET	Integer Variable	Elapsed Time (32 bits)

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.

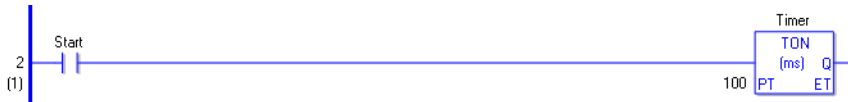


For time-based settings, double-click the timer instruction to display the setup dialog box.

Time base	Description	PT value/ET value
ms	Specify the time in units of ms. 0 to 2147483647×1 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 2147483647×1 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 214748364×10 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 21474836×100 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 $2147483 \times$ seconds

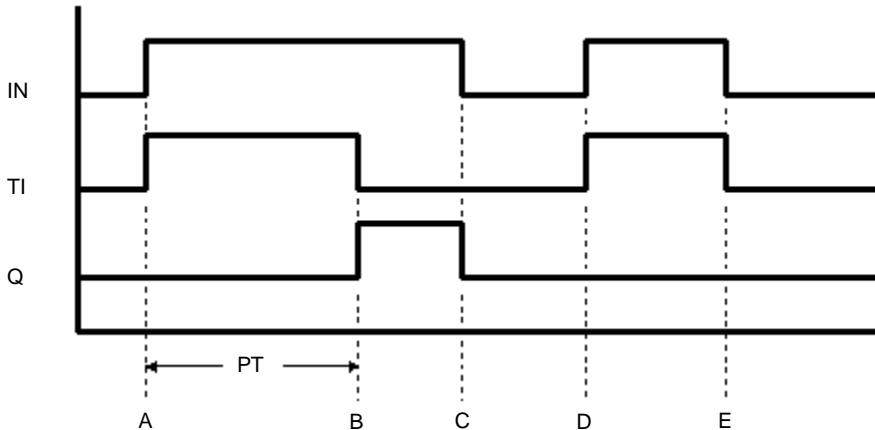
Program Example

TON



- (1) When the variable of the normally open instruction turns ON, because the TON 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 variable of the normally open instruction turns OFF , the elapsed time (.ET) resets to 0.
 - The timer measurement bit .TI turns OFF.
 - The timer output bit .Q turns OFF.

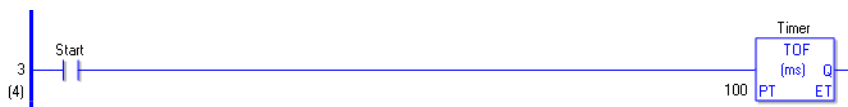
◆ Timing Chart for TON Instruction Operation



- 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.

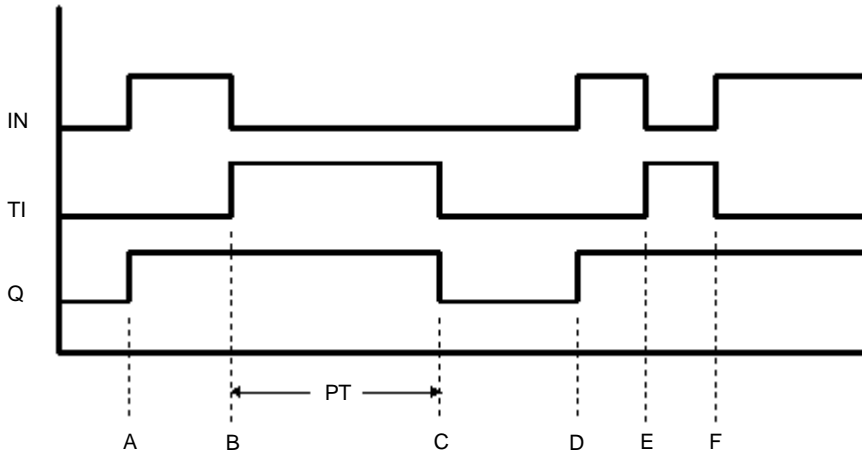
Program Example

TOF



- (1) When the variable for the NO instruction turns ON , because the TON instruction is triggered, the elapsed time .ET resets to 0.
 - 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 turns OFF, the elapsed time .ET increases in 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) When a value outside the setting range is input, an error occurs and error code “6706” is set in #L_CalcErrCode. To check details of the error, refer to #L_CalcErrCode. The instruction is not executed when a value outside the setting range is input.

■ **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**

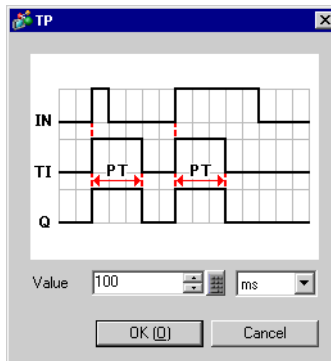
Timer variables used in TP instructions are structure variables. The following table lists the internal structures.

Timer Variable

Timer Variable	Variable Settings	Description
Variable Name.TI	Bit Variable	Turns ON when the timer begins.
Variable Name.Q	Bit Variable	Turns ON upon completion of the timer.
Variable Name.PT	Integer Variable	Preset Time (32 bits)
Variable Name.ET	Integer Variable	Elapsed Time (32 bits)

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

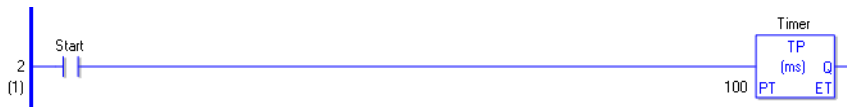
Enter the timer value and units.



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

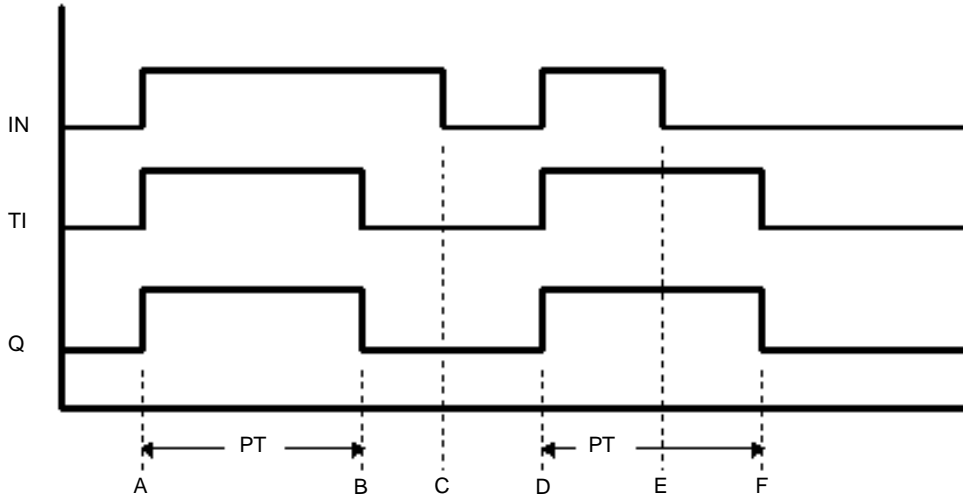
Time base	Description	PT value/ET value
ms	Specify the time in units of ms. 0 to 2147483647×1 ms	The PT value is specified and displayed in units of ms. The ET value is displayed in units of ms. Setting range = 0 to 2147483647×1 ms
10ms	Specify the time in units of 10 ms.	The PT value is set and displayed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

Program example



- (1) When the variable of the normally open instruction turns ON, the TP instruction is triggered. When the TP instruction is triggered, it starts the timer measurement to detect positive transition, regardless of the condition of the instruction before the timer instruction. The elapsed time .ET increases in the specified time-based units.
 - 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 increases to equal the preset time .PT, the TP instruction turns OFF. After the preset time elapses, the timer output bit .Q turns OFF regardless of the instruction conditions to the left of the TP instruction.
 - When $PT \leq ET$, .ET immediately resets to 0.
 - When the elapsed time .ET has become equal to the preset time .PT, the timer measurement bit .TI turns OFF.
 - If the TP instruction is disabled, the timer output bit .Q is turned 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) When a value outside the setting range is input, an error occurs and error code “6706” is set in #L_CalcErrCode. To check details of the error, refer to #L_CalcErrCode. The instruction is not executed when a value outside the setting range is input.

■ 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
TOFA (Accumulated OFF Delay Timer)		Timer	2

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

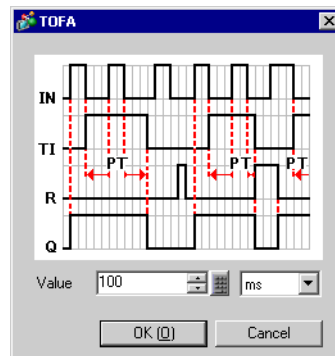
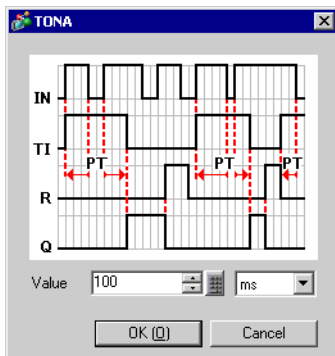
Timer variables in TONA and TOFA instructions are structure variables. The following table lists the internal structures.

Timer Variable

Timer Variable	Variable Settings	Description
Variable Name.TI	Bit Variable	Turns ON when the timer begins.
Variable Name.Q	Bit Variable	Turns ON upon completion of the timer.
Variable Name .R	Bit Variable	Resets the current timer. Clear (0).
Variable Name.PT	Integer Variable	Preset Time (32 bits)
Variable Name.ET	Integer Variable	Elapsed Time (32 bits)

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.

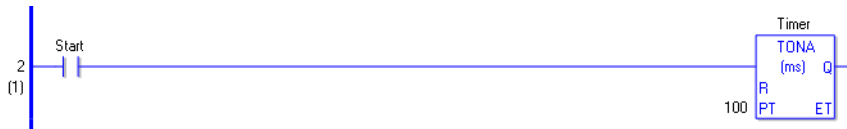


For time-based settings, double-click the timer instruction to display the setup dialog box.

Time base	Description	PT value/ET value
ms	Specify the time in units of ms. 0 to 2147483647×1 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 2147483647×1 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 214748364×10 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 21474836×100 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 $2147483 \times$ seconds

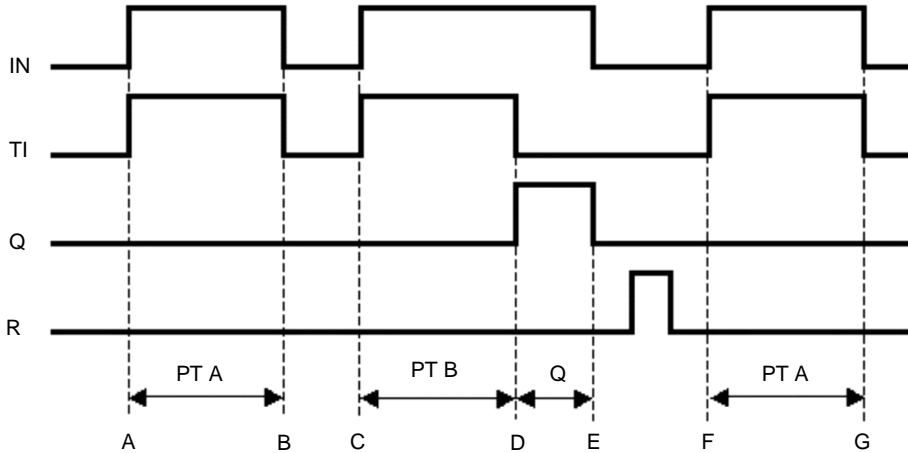
Program Example

TONA



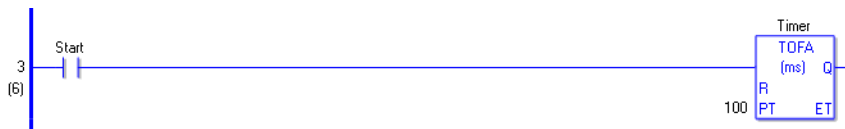
- (1) When the variable of the normally open instruction turns ON , because the TONA instruction is triggered, the elapsed time .ET increases in the specified time-based units.
 - The timer measurement bit .TI turns ON.
 - The timer output bit .Q turns OFF.
- (2) When the elapsed time .ET increases to 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.
- (4) The TONA instruction is a multiply instruction. The current value is not reset unless the R coil (reset) is turned ON.

◆ **Timing Chart for the TONA Instruction**



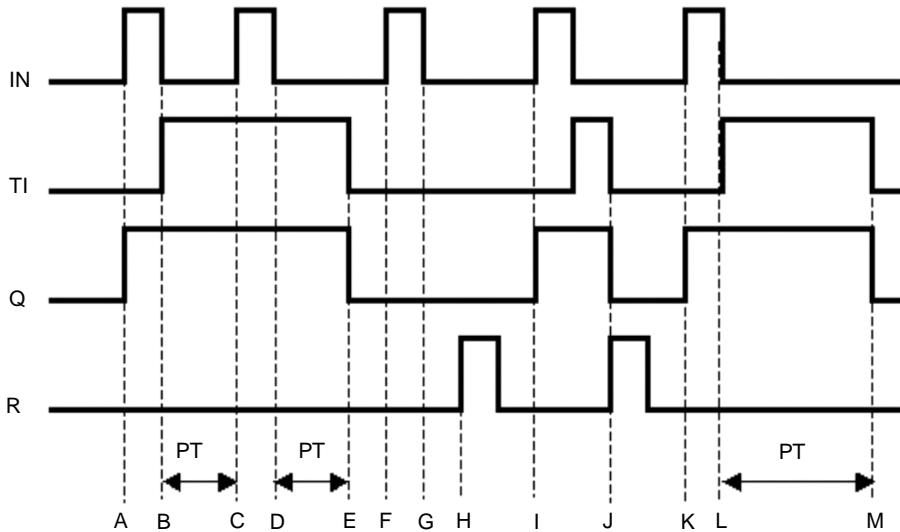
- Points A and F The timer input bit IN turns ON and the timer measurement bit TI turns ON. The timer starts and the elapsed time ET increases. The timer output bit Q remains OFF.
- Points B and G The timer input bit IN turns OFF, and if the elapsed time ET is less than the preset time PT, the timer 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 and G Even when IN (input) turns ON or OFF, Q (output) and TI (timer measurement) 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 measurement) are reset. The ET timer value is also reset and cleared to zero.
- 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 a value out of the setting range is entered, an error occurs and error code “6706” is set in #L_CalcErrCode. For details of the error, check the content of #L_CalcErrCode. The instruction is not executed when a value out of the setting range is entered.

30.5.5 Counter Instruction

■ CTU and CTUP (Up Counter)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
CTU (Up Counter - level transition)		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	Variable 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 executing CTU and CTUP instructions, if the counter reset bit variable .R is OFF and the current value .CV is less than the preset value .PV, the current value .CV increases 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 resets to 0. The counter output bit variable .Q also turns OFF.

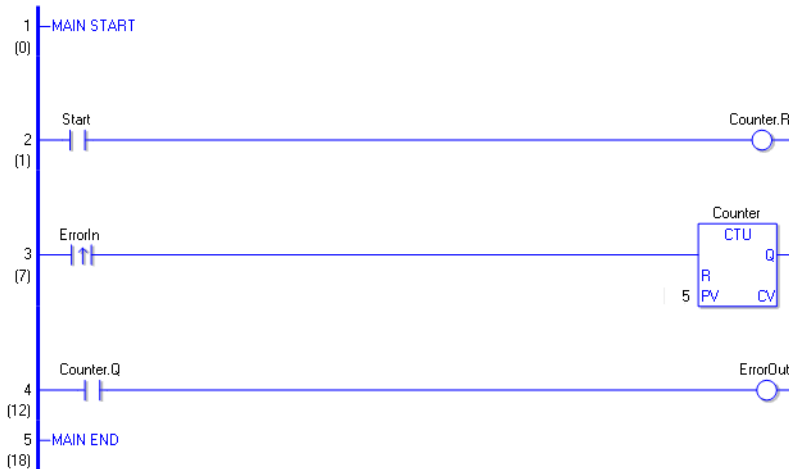
Program Example

CTU

In the following example, if five operation errors are counted within one minute, an error is 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 CTDP (Down Counters)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
CTD (Down Counter - level transition)		Counter	2
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
CTDP (Down Counter - positive transition)		Counter	2

◆ Explanation of the CTD and CTDP Instructions

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

Counter Variable

Counter Variable	Variable 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 CDT instructions are ON, if the counter reset bit variable .R is OFF, the current value .CV decreases 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.

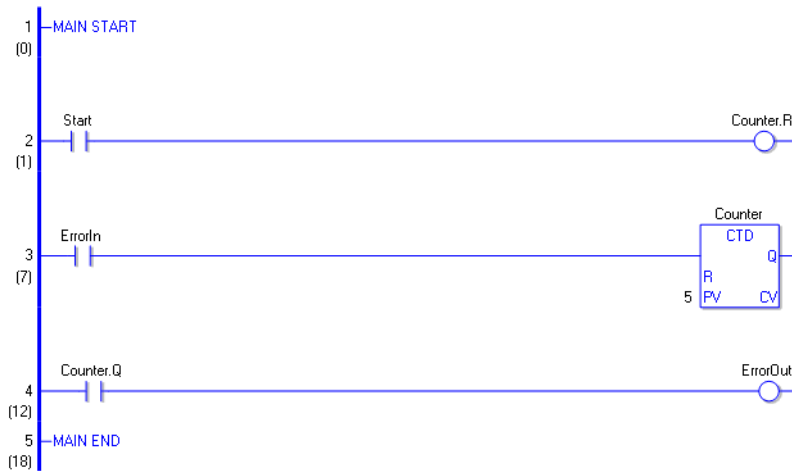
Program Example

CTD

In the following example, if five operation errors are counted within one minute, an error is displayed.

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

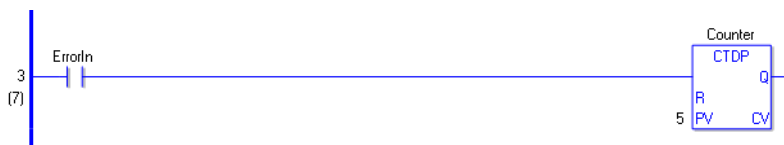
To count operation errors, create a separate error input trigger.



- (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 CTD 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 CTD instruction turns ON, and the OUT instruction in rung 4 outputs the error detection message.

Program Example

CTDP



The difference between the CTD and CTDP instructions is whether they decrease the .CV value based on the level or by detecting positive transition, as a counter instruction.

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 transition)		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	Variable 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.

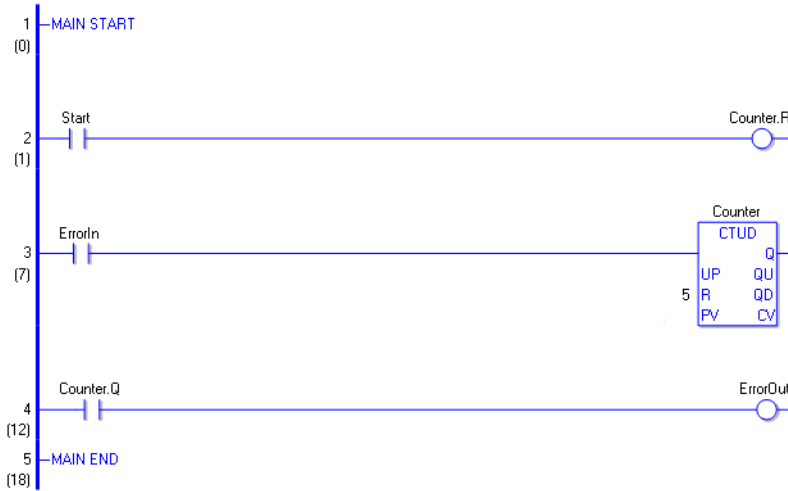
Program Example

CTUD

In the following example, if five operation errors are counted within one minute, an error is displayed.

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

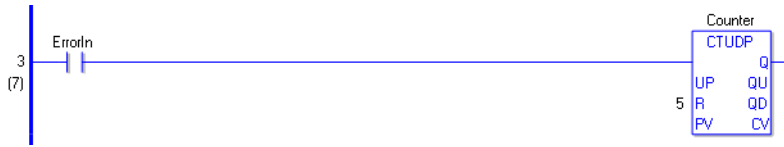
To count operation errors, create a separate error input trigger.



- (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 (enables up count while ON) is ON and the .CV value (current value) of the operation error counter becomes equal to the .PV value (preset value), .Q (turns ON when the current value reaches the preset value) and .QU (turns ON when the current value reaches the preset value while the Up/Down counter is used) turn ON. When .UP (enables up count while ON) is OFF and the .CV value (current value) becomes 0 or less, .Q (turns ON when the current value reaches the preset value) and .QD (turns ON when the current value reaches 0 or less while the Up/Down counter is used) 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 an 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.

30.5.6 R/W Instructions

■ JRD and JRDP (Time Read)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JRD (Time Read - level transition)		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	Variable Settings	Description
Variable Name.HR	Integer Variable	Hours are input in BCD.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.SEC	Integer Variable	Seconds are input in BCD.

When JRD and JRDP instructions receive power, the current time is 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.

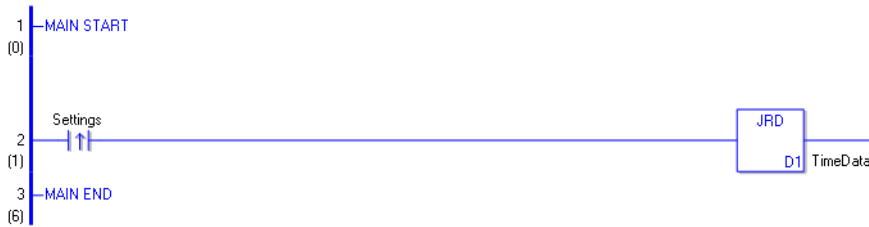
◆ **Confirming Execution Results**

- (1) When a value outside the setting range is input, an error occurs and error code “6706” is set in #L_CalcErrCode. To check details of the error, refer to #L_CalcErrCode.
- (2) #L_CalcZero turns ON when the value of the readout result D1 is 00 (h):00 (min):00 (s).

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 transition)		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

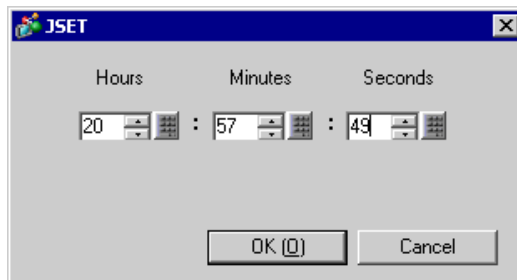
Time Variable	Variable Settings	Description
Variable Name.HR	Integer Variable	Hours are input in BCD.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.SEC	Integer Variable	Seconds are input in BCD.

When JSET and JSETP instructions receive power, the specified time is 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 the time variable in D1, the time variables .HR, .MIN, and .SEC are set to 12, 10, and 45 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

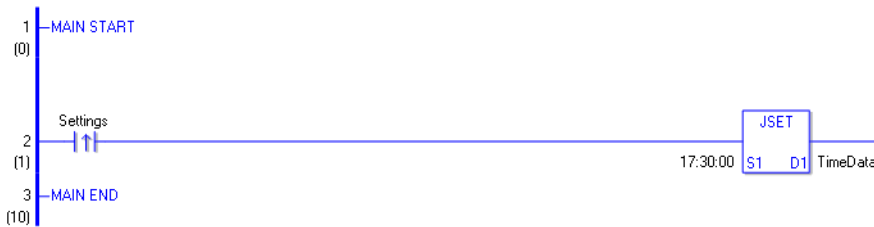
◆ **Confirming Execution Results**

- (1) When a value outside the setting range is input, an error occurs and error code “6706” is set in #L_CalcErrCode. To check details of the error, refer to #L_CalcErrCode.
- (2) #L_CalcZero turns ON when the value of the readout result D1 is 00 (h):00 (min):00 (s).

Program Example

JSET

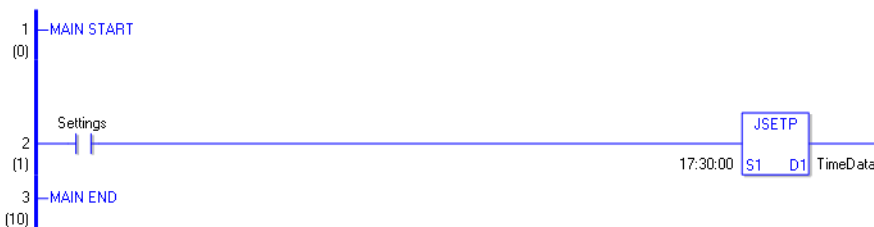
Stores the defined time in the time variable.



- (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 transition)		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	Variable Settings	Description
Variable Name .YR	Integer Variable	The year is input in BCD.
Variable Name .MO	Integer Variable	The month is input in BCD.
Variable Name DAY	Integer Variable	The day is input in BCD.

When the NRD and NRDP instructions receive power, the current date is stored in D1. The stored date variable can be extracted into year, month and day by specifying structure elements. When a date 10 (month) 20 (day), 2005 is stored in the date variable D1, the .HR date is 05, the .MO date is 10, and the .DAY date is 20.

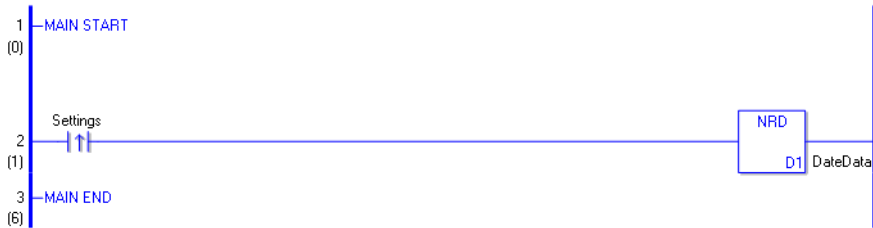
◆ **Confirming Execution Results**

- (1) When a value outside the setting range is input, an error occurs and error code “6706” is set in #L_CalcErrCode. To check details of the error, refer to #L_CalcErrCode.

Program Example

NRD

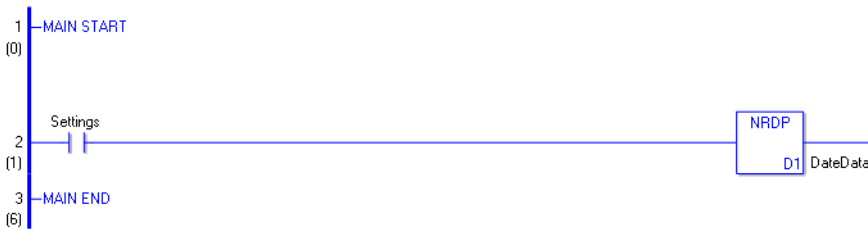
Stores the current date in the date variable.



- (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 transition)		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

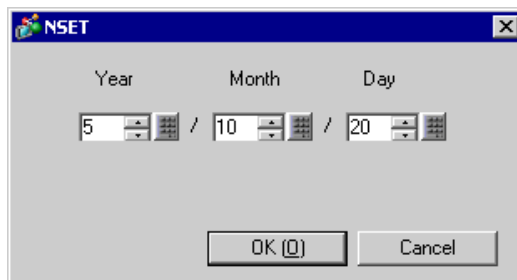
Date Variable	Variable Settings	Description
Variable Name .YR	Integer Variable	The year is input in BCD.
Variable Name .MO	Integer Variable	The month is input in BCD.
Variable Name .DAY	Integer Variable	The day is input in BCD.

When the NSET and NSETP instructions receive power, the specified date is 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 specified with the JSET instruction, October 20, 2005, is stored in the date variable in D1, the date variables .YR, .MO, and .DAY are set to 05, 10, and 20 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.)

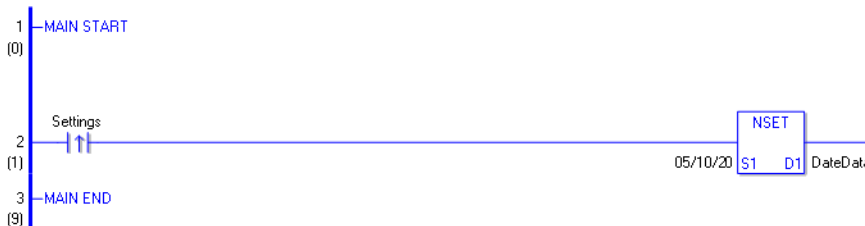
◆ **Confirming Execution Results**

- (1) When a value outside the setting range is input, an error occurs and error code “6706” is set in #L_CalcErrCode. To check details of the error, refer to #L_CalcErrCode.

Program Example

NSET

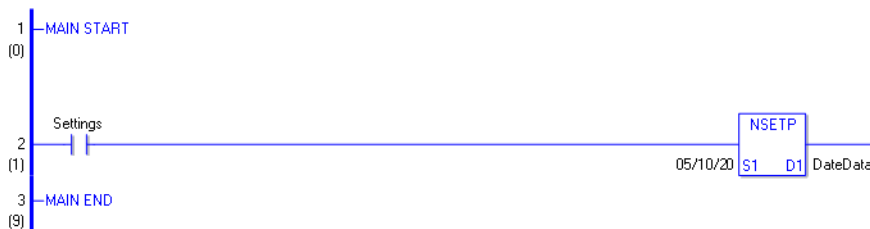
Stores the setup date in the date variable.



- (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

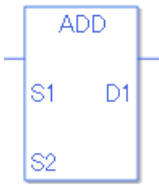


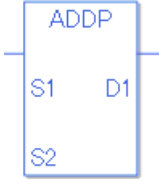
- (1) When the normally open instruction turns ON, the NSETP instruction will be executed. When the NSETP instruction is executed, the date 10 (month) 20 (day), 2005 specified in the dialog box is stored in the date variable in D1.

30.5.7 Operation (Arithmetic)

■ ADD and ADDP (Add)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ADD (Add - level transition)		Operation	4 to 13

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

◆ Operand Settings

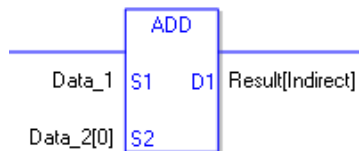
The following table lists the specifiable contents of operands (S1, S2, and D1) for the ADD and ADDP instructions.

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

e.g. 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 [indirect specification] = 3 Steps} + {1 Step} = 7 Steps.

The last 1 step is included in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1) and (S2) in the ADD and ADDP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
Time	.HR/.MIN/.SEC only	2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the ADD and ADDP instructions.

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

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (only output)		Arrays and modifiers are not specified	1	○
			Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
			Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
			Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float		—	1	○
			Specify float variable [constant]	2	○
			Specify float variable [variable]	3	○
	Real		—	1	○
			Specify real variable [constant]	2	○
			Specify real variable [variable]	3	○
	Timer		.PT/.ET only	2	○
	Counter		.PV/.CV only	2	○
	Date		.YR/.MO/.DAY only	2	○
Time		.HR/.MIN/.SEC only	2	○	
PID		.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	1	○	
	D_		Modifiers are not specified	1	○
			D_****.B/W [constant]	2	○
			D_****.B/W [address]	3	○
	F_	—	1	○	
	R_	—	1	○	
	T_		.PT/.ET only	2	○
	C_		.PV/.CV only	2	○
	N_		.YR/.MO/.DAY only	2	○
	J_		.HR/.MIN/.SEC only	2	○
U_		.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Constant	—	—	—	×	

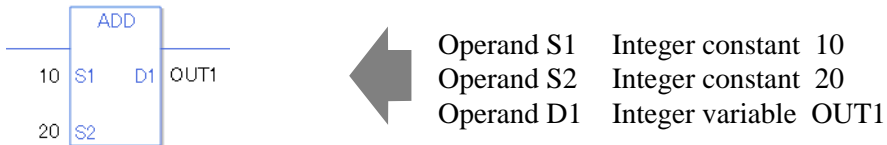
◆ **Explanation of the ADD and ADDP Instructions**

The ADD and ADDP instructions are add instructions. When an ADD instruction is executed, S1 is 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.

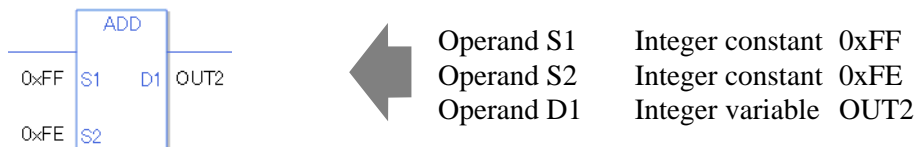
Refer to the following for specifying a constant.

When operand D1 is an integer variable



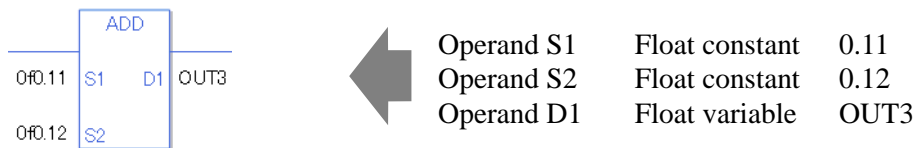
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

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



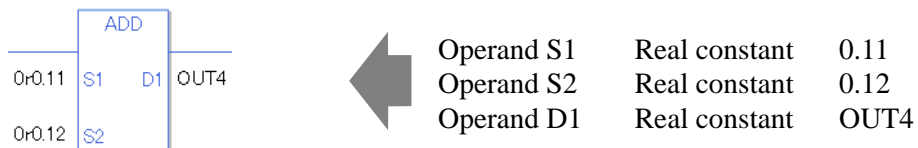
When operand D1 is a float variable

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



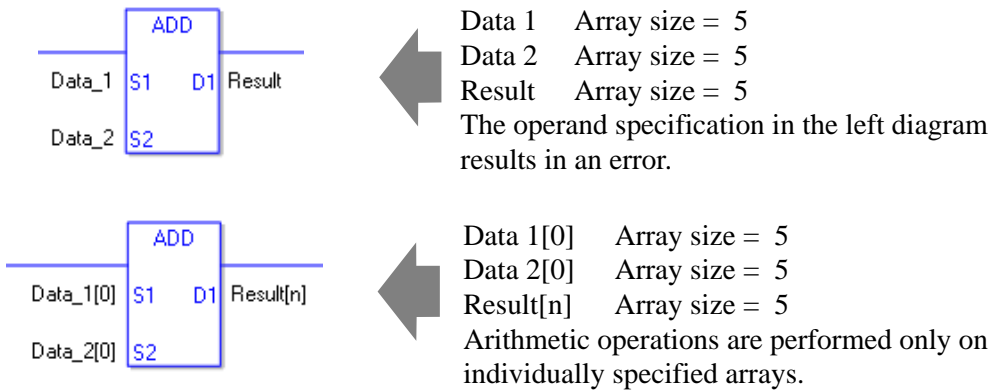
When operand D1 is a real variable

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



When 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.



◆ **Confirming Execution Results**

- (1) If an overflow occurs after the execution of the instruction, the system variable (bit) #L_CalcCarry turns ON.
- (2) If you use any numeric value that cannot be expressed in operands S1 and S2 (infinite or non-numeric value), the instruction will not be executed.
As error indication, error code “6706” is set in #L_CalcErrCode.
The output result D1 maintains the value from the last successfully executed instruction.
- (3) #L_Error turns ON and an error code (6706) is generated in #L_CalcErrCode.
- (4) When the result of the execution is 0, the system variable #L_CalcZero turns ON.

(Notes)

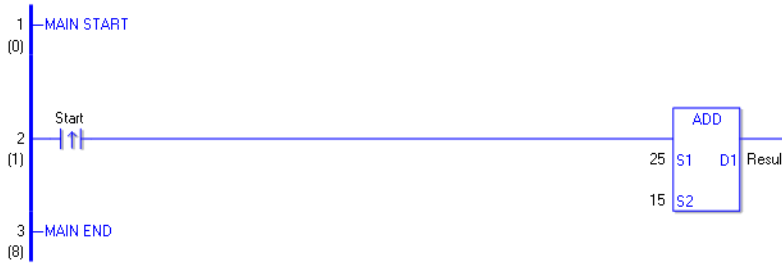
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

ADD

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

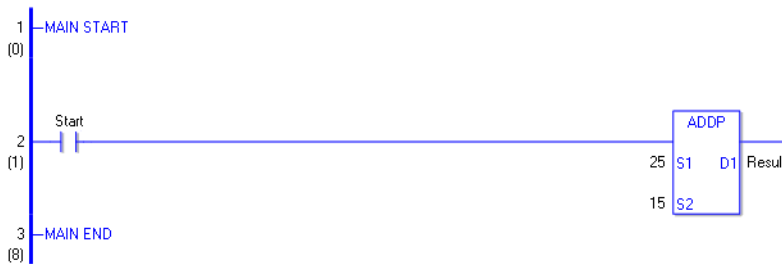


(1) When the positive transition instruction in the operation turns ON, the ADD instruction will be executed. When the ADD instruction is executed, the result value of 40, obtained from $25 + 15 = 40$, is stored in D1.

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

Program Example

ADDP



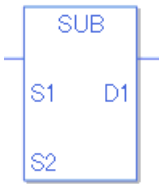
(1) When the normally open instruction turns ON, the ADDP instruction will be executed. When the ADDP instruction is executed, the result value of 40, obtained from $25 + 15 = 40$, is stored in D1.

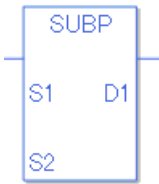
Even when the operation is a normally open instruction, only when the upward transition is detected, will the ADDP instruction execute.

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 transition)		Operation	4 to 13

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

◆ Operand Settings

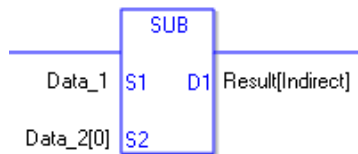
The following table lists the specifiable contents of the (S1, S2, and D1) operands for the SUB and SUBP instructions.

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

e.g. 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 [indirect specification] = 3 Steps} + {1 Step} = 7 Steps.

The last 1 step is the number of steps in an instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1) and (S2) in the SUB and SUBP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
Time	.HR/.MIN/.SEC only	2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the SUB and SUBP instructions.

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

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (output only)		Arrays and modifiers are not specified	1	○
			Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
			Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
			Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float		—	1	○
			Specify float variable [constant]	2	○
			Specify float variable [variable]	3	○
	Real		—	1	○
			Specify real variable [constant]	2	○
			Specify real variable [variable]	3	○
	Timer		.PT/.ET only	2	○
	Counter		.PV/.CV only	2	○
	Date		.YR/.MO/.DAY only	2	○
	Time		.HR/.MIN/.SEC only	2	○
PID		.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	1	○	
	D_		Modifiers are not specified	1	○
			D_****.B/W [constant]	2	○
			D_****.B/W [address]	3	○
	F_	—	1	○	
	R_	—	1	○	
	T_		.PT/.ET only	2	○
	C_		.PV/.CV only	2	○
	N_		.YR/.MO/.DAY only	2	○
	J_		.HR/.MIN/.SEC only	2	○
U_		.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Constant	—	—	—	×	

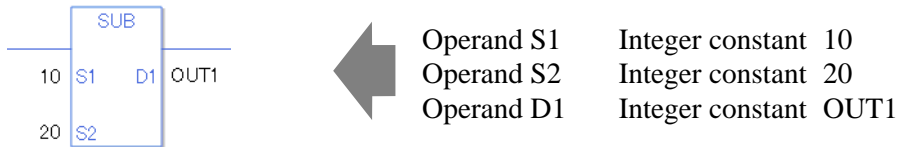
◆ **Explanation of the SUB and SUBP Instructions**

The SUB and SUBP instructions are subtraction instructions. When a SUB instruction is executed, S1 is 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.

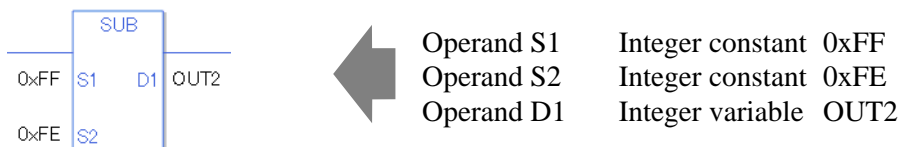
Refer to the following for specifying a constant.

When operand D1 is an integer variable



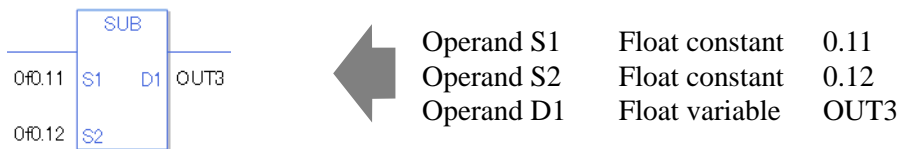
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

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



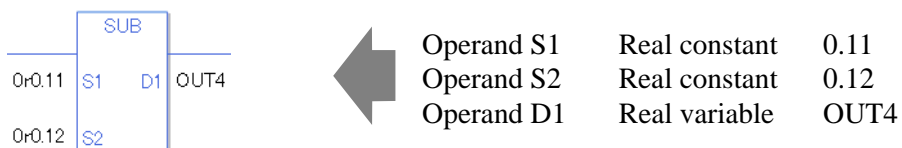
When operand D1 is a float variable

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



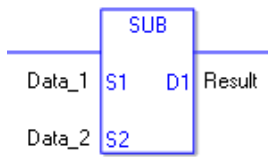
When operand D1 is a real variable

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

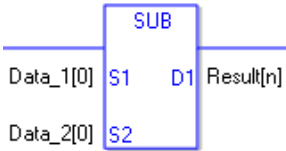


When subtracting specified array data (integer variable array) Specify the array using data [0] or data [N] (N indicates an integer variable).

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 after the execution of the instruction, the system variable (bit) #L_CalcCarry turns ON.
- (2) If you use any numeric value that cannot be expressed in operands S1 and S2 (infinite or non-numeric value), the instruction will not be executed.
 As error indication, error code “6706” is set in #L_CalcErrCode.
 The output result D1 maintains the value from the last successfully executed instruction.
- (3) #L_Error turns ON and an error code (6706) is generated in #L_CalcErrCode.
- (4) When the result of the execution is 0, the system variable #L_CalcZero turns ON.

(Notes)

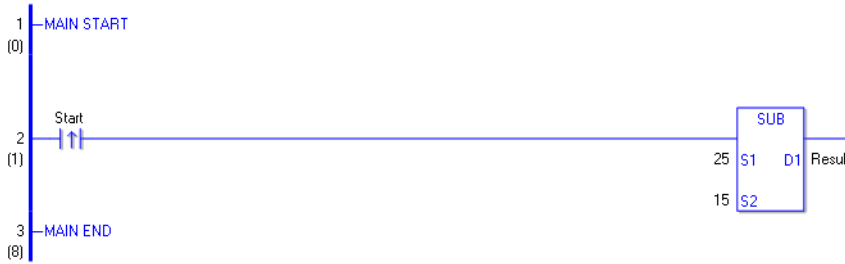
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

SUB

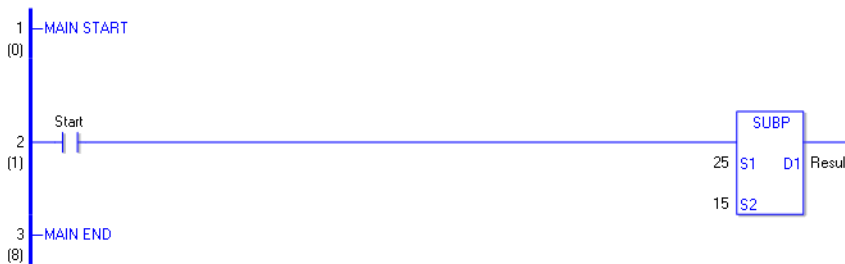
Subtracts one constant from another and stores the result in the integer variable.



- (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


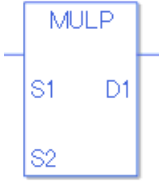
SUBP



- (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 transition)		Operation	4 to 13
MULP (Multiplication - positive transition)		Operation	4 to 13

◆ Operand Settings

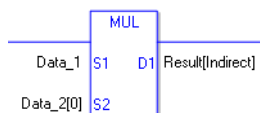
The following table lists the specifiable contents of the (S1, S2, and D1) operands for the MUL and MULP instructions.

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

e.g. 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 [indirect specification] = 3 steps} + {1 step} = 7 steps.

The last 1 step is the number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1) and (S2) in the MUL and MULP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the MUL and MULP instructions.

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

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (output only)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
Time	.HR/.MIN/.SEC only	2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	—	×
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	2	○
		D_****.B/W [address]	3	○
	F_	—	1	○
	R_	—	1	○
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY only	2	○
	J_	.HR/.MIN/.SEC only	2	○
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Constant	—	—	—	×

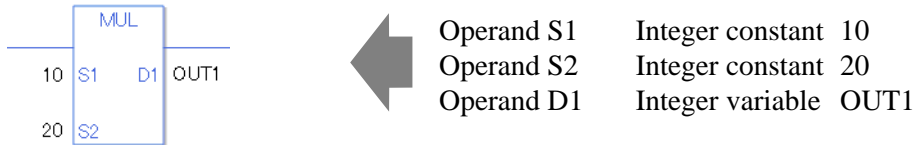
◆ **Explanation of the MUL and MULP Instructions**

The MUL and MULP instructions are multiplication instructions. When a MUL instruction is executed, S1 is 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.

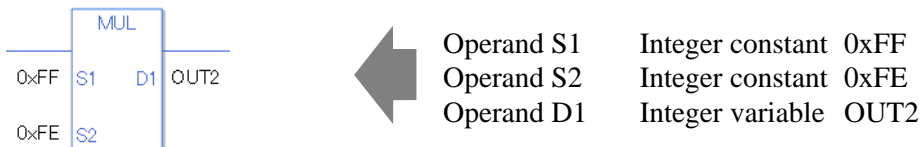
Refer to the following for specifying a constant.

When operand D1 is an integer variable



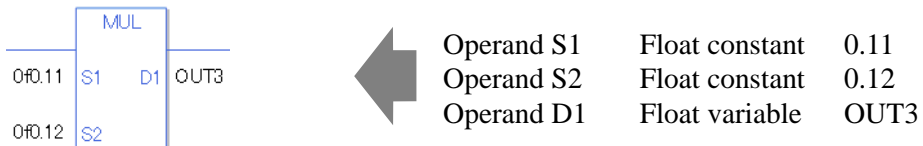
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

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



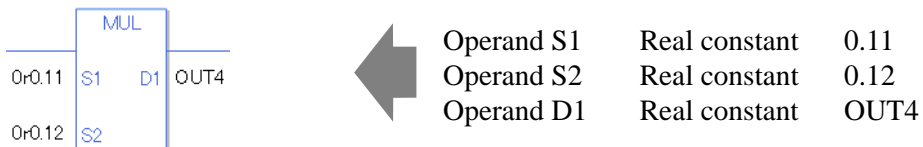
When operand D1 is a float variable

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



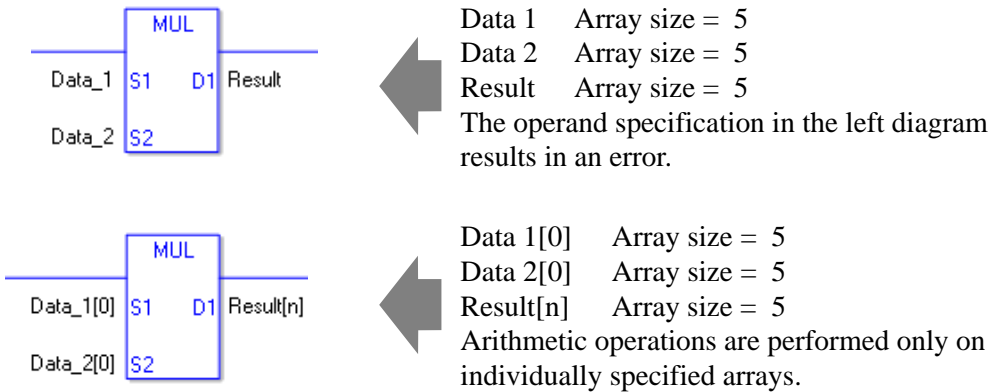
When operand D1 is a real variable

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



When 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 after the execution of the instruction, the system variable (bit) #L_CalcCarry turns ON.
- (2) If you use any numeric value that cannot be expressed in operands S1 and S2 (infinite or non-numeric value), the instruction will not be executed.
As error indication, error code “6706” is set in #L_CalcErrCode.
The output result D1 maintains the value from the last successfully executed instruction.
- (3) #L_Error turns ON and an error code (6706) is generated in #L_CalcErrCode.
- (4) When the execution result is 0, the system variable #L_CalcZero turns ON.

(Notes)

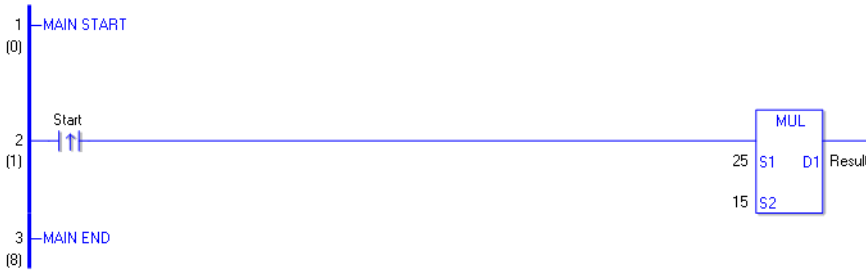
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

MUL

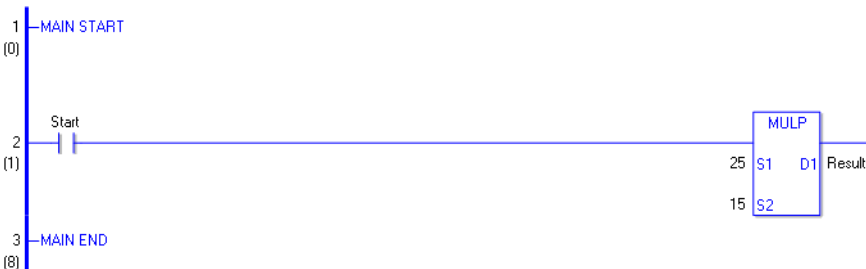
Multiplies one constant by another and stores the result in the integer variable.



- (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

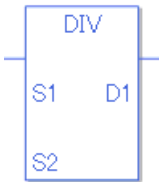
MULP

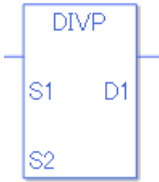


- (1) When the normally open instruction turns ON, the MULP instruction will be executed. When the MULP instruction is executed, the result value of 10, obtained from $25 \times 15 = 375$, is stored in D1. Even when using a normally open instruction, only the upward transition is detected, and the MULP instruction is executed. 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 transition)		Operation	4 to 13

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

◆ **Operand Settings**

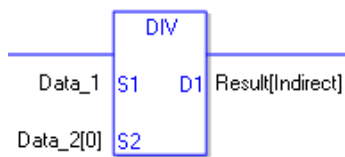
The following table lists the specifiable contents of the (S1, S2, and D1) operands for DIV and DIVP instructions.

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

e.g. 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 [indirect specification] = 3 Steps} + {1 Step} = 7 Steps.

The last 1 step is the number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1) and (D2) in the DIV and DIVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the DIV and DIVP instructions.

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

Continued

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (output only)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
Time	.HR/.MIN/.SEC only	2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	—	×
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	2	○
		D_****.B/W [address]	3	○
	F_	—	1	○
	R_	—	1	○
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY only	2	○
	J_	.HR/.MIN/.SEC only	2	○
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Constant	—	—	—	×

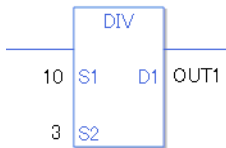
◆ **Explanation of the DIV and DIVP Instructions**

The DIV and DIVP instructions are division instructions. When a DIV instruction is executed, S1 is 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.

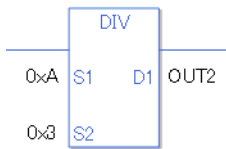
When operand D1 is an integer variable



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.
 Example: 10 (S1) ÷ 3 (S2) = 3 (D1)

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

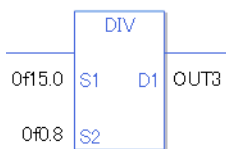
When 0x (zero and lower case “x”) is input, the following values are interpreted as hexadecimal values.



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

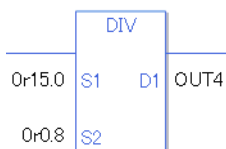
When 0f (zero and lower case “f”) is input, the following values become float values.



Operand S1 Float constant 15.0
 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

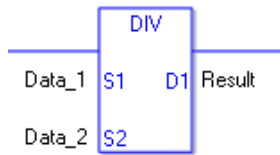
When 0r (zero and lower case “r”) is input, the following values become real values.



Operand S1 Float constant 15.0
 Operand S2 Float constant 0.8
 D1 Operand Float variable OUT4
 The operation result is a value including the decimal point.
 Example: 0r 15.0 (S1) ÷ 0r 0.8 (S2) = 18.75 (D1)

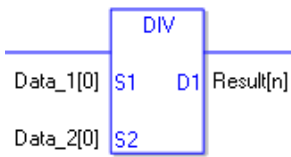
When dividing specified array data (integer variable array) Specify the array using data [0] or data [N] (N indicates an integer variable).

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 after the execution of the instruction, the system variable (bit) #L_CalcCarry turns ON.
- (2) If you use any numeric value that cannot be expressed in operands S1 and S2 (infinite or non-numeric value), the instruction will not be executed.
As error indication, error code “6706” is set in #L_CalcErrCode.
The output result D1 maintains the value from the last successfully executed instruction.
- (3) #L_Error turns ON and an error code (6706) is generated in #L_CalcErrCode.
- (4) When the execution result is 0, the system variable #L_CalcZero turns ON.

(Notes)

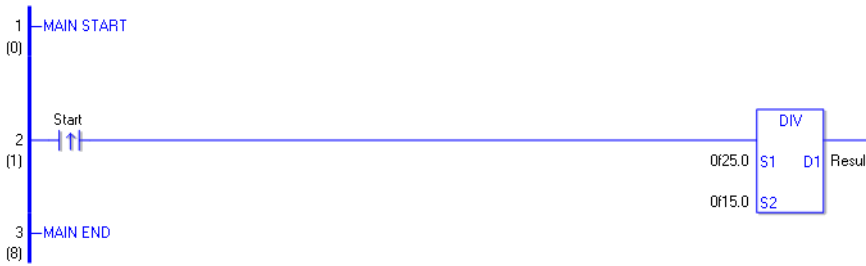
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

DIV

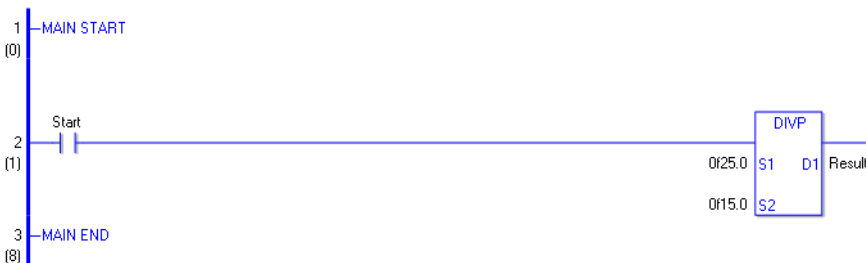
Divides one constant by another and stores the result in the float variable.



- (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 $25 \div 15 = 1.66666\dots$, 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

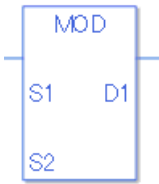
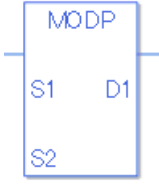
DIVP



- (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 $25 \div 15 = 1.66666\dots$, 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 (Modulus)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
MOD (Modulus - level transition)		Operation	4 to 13
MODP (Modulus - positive transition)		Operation	4 to 13

◆ Operand Settings

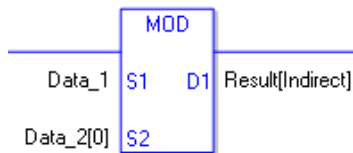
The following table lists the specifiable contents of operands (S1, S2, and D1) for the MOD and MODP instructions.

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

e.g. 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 [indirect specification] = 3 Steps} + {1 step} = 7 steps.

The last 1 step is included in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1) and (S2) in the MOD and MODP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the MOD and MODP instructions.

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

Continued

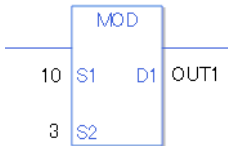
Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (output only)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
Date	.YR/.MO/.DAY only	2	○	
Time	.HR/.MIN/.SEC only	2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	—	×
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	2	○
		D_****.B/W [address]	3	○
	F_	—	—	×
	R_	—	—	×
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY only	2	○
	J_	.HR/.MIN/.SEC only	2	○
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Constant	—	—	—	×

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

The MOD and MODP instructions are modulus operations. When a MOD instruction is executed, S1 is 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.

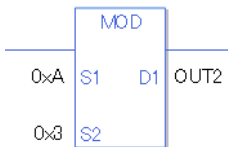
When operand D1 is an integer variable



Operand S1 Integer constant 10
 Operand S2 Integer constant 3
 Operand D1 Integer variable OUT1
 Example: $10 (S1) \div 3 (S2) = 3$ Remainder 1
 Therefore, $D1 = 1$

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

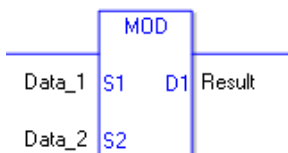
When 0x (zero and lower case “x”) is input, the following values are interpreted as hexadecimal values.



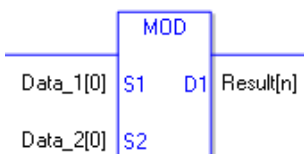
Operand S1 Integer constant 0xFF
 Operand S2 Integer constant 0xFE
 Operand D1 Integer variable OUT2
 Example: $10 (S1) \div 3 (S2) = 3$ Remainder 1
 Therefore, $D1 = 1$

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.



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 after the execution of the instruction, the system variable (bit) #L_CalcCarry turns ON.
- (2) #L_Error turns ON and an error code (6706) is generated in #L_CalcErrCode.
- (3) When the result of the execution is 0, the system variable #L_CalcZero turns ON.

(Notes)

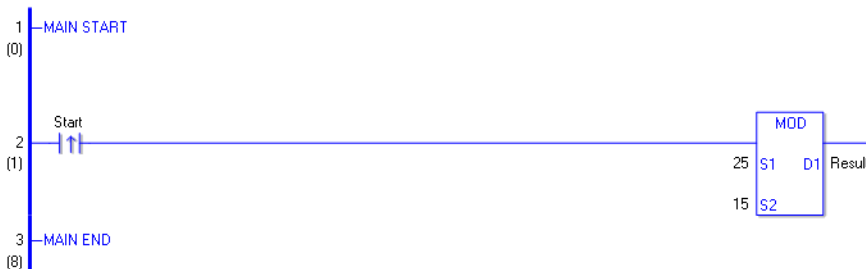
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

MOD

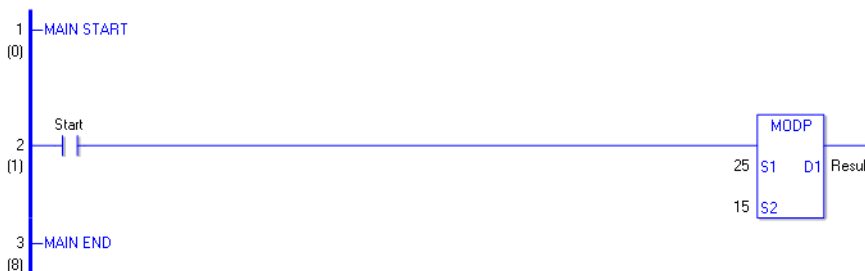
Performs modulus operation on two constants and stores the result in the integer variable.



- (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 $25 \div 15 = 1$ (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 $25 \div 15 = 1$ (remainder 10), is stored in D1.

Even when using a normally open instruction, only the upward transition is detected, and the MODP instruction is executed.

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 transition)		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 table lists the configurable conditions for Operand (D1) in the INC and INCP instructions.

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	2	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	2	○
Symbol	Bit	—	—	×
	Word	—	2	○

Continued

Explanations of Each Instruction

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (output only)	Arrays and modifiers are not specified	2	○
		Specify integer variable [constant]	3	○
		Specify integer variable [variable]	4	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	3	○
	Counter	.PV/ .CV only	3	○
	Date	.YR/ .MO/ .DAY only	3	○
	Time	.HR/ .MIN/ .SEC only	3	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	3	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	L_	—	—	×
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	—	×
		D_****.B/W [address]	—	×
	F_	—	—	×
	R_	—	—	×
	T_	.PT / .ET only	3	○
	C_	.PV / .CV only	3	○
	N_	.YR / .MO / .DAY only	3	○
	J_	.HR / .MIN / .SEC only	3	○
U_	.KP / .TR / .TD / .PA / .BA / .ST only	3	○	
Constant	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	—	×
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	×
	Integer	-2147483648 to 2147483647	—	×

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

The INC and INCP instructions are increment 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 after the execution 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)

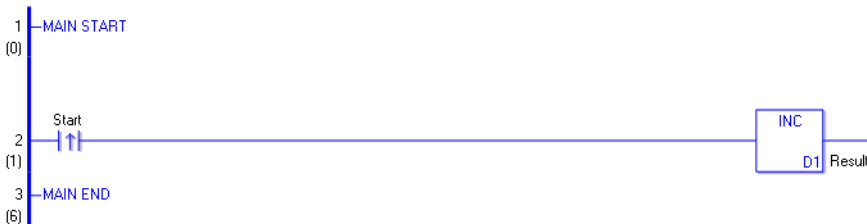
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

INC

1 is added to D1 every time the INC instruction is conducted.

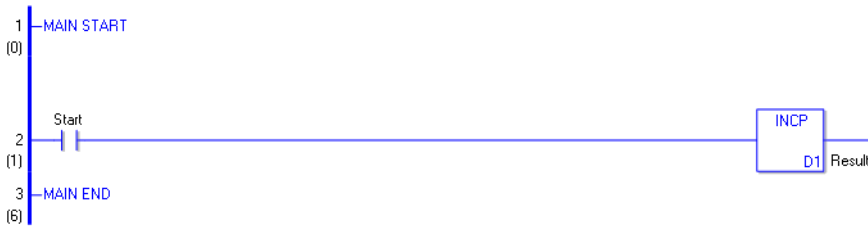


(1) When the positive transition instruction turns ON, the INC instruction will be executed. When the INC instruction is executed, 1 is added to the result data (integer variable) in D1.

When using a normally open instruction, as long as the variable is ON, the INC instruction is always executed, and 1 is added every time a scan is executed.

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, only the upward transition is detected, and the INCP instruction is executed. 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 transition)		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 table lists the configurable conditions for Operand (D1) in the DEC and DECP instructions.

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	2	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	2	○
Symbol	Bit	—	—	×
	Word	—	2	○

Continued

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (output only)	Arrays and modifiers are not specified	2	○
		Specify integer variable [constant]	3	○
		Specify integer variable [variable]	4	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	3	○
	Counter	.PV/ .CV only	3	○
	Date	.YR/ .MO/ .DAY only	3	○
	Time	.HR/ .MIN/ .SEC only	3	○
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	3	○

Continued

Explanations of Each Instruction

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	L_	—	—	×
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	—	×
		D_****.B/W [address]	—	×
	F_	—	—	×
	R_	—	—	×
	T_	.PT / .ET only	3	○
	C_	.PV / .CV only	3	○
	N_	.YR / .MO / .DAY only	3	○
	J_	.HR / .MIN / .SEC only	3	○
U_	.KP / .TR / .TD / .PA / .BA / .ST only	3	○	
Constant	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	—	×
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	×
	Integer	-2147483648 to 2147483647	—	×

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

The DEC and DECP instructions are decrement instructions. When the DEC instruction is executed, 1 is subtracted from D1.

The DEC and DECP instructions always pass power.

◆ **Confirming Execution Results**

(1) If an overflow occurs after the execution 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)

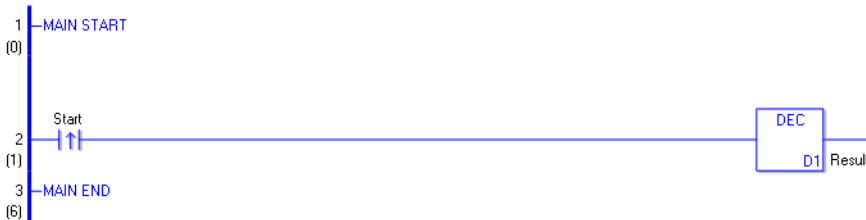
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

DEC

1 is subtracted from D1 every time the DEC instruction receives power.



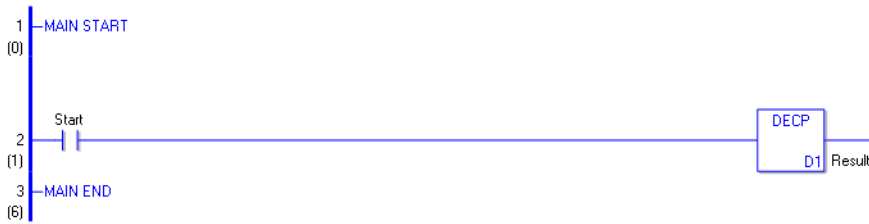
(1) When the positive transition instruction turns ON, the DEC instruction will be executed.

When the DEC instruction is executed, 1 is subtracted from D1.

When using a normally open instruction, as long as the variable is ON, the DEC instruction is always executed, and 1 is subtracted every time a scan is executed.

Program Example

DECP

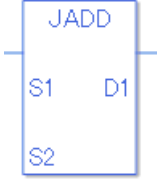


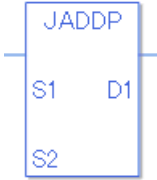
- (1) When the normally open instruction turns ON, the DECP instruction will be executed. When the DECP instruction is executed, 1 is subtracted from D1. Even when using a normally open instruction, only the upward transition is detected, and the DECP instruction is executed. Therefore, even when the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

30.5.8 Operation (Time)

■ JADD and JADDP (Time Addition)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JADD (Time Addition - level transition)		Operation	4

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

◆ Operand Settings

The following table lists the configurable conditions for Operands (S1, S2, and D1) in the JADD and JADDP instructions.

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	—	×
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×
Symbol	Bit	—	—	×
	Word	—	—	×

Continued

Explanations of Each Instruction

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (output only)	Arrays and modifiers are not specified	—	×
		Specify integer variable [constant]	—	×
		Specify integer variable [variable]	—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	—	×
	Counter	.PV/ .CV only	—	×
	Date	.YR/ .MO/ .DAY only	—	×
	Time	Other than .HR / .MIN / .SEC	4	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	No. of Instruction Steps	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	L_	—	—	×
	Q_	—	—	×
	D_	Modifiers are not specified	—	×
		D_****.B/W [constant]	—	×
		D_****.B/W [address]	—	×
	F_	—	—	×
	R_	—	—	×
	T_	.PT / .ET only	—	×
	C_	.PV / .CV only	—	×
	N_	.YR / .MO / .DAY only	—	×
	J_	Other than .HR / .MIN / .SEC	4	○
U_	.KP / .TR / .TD / .PA / .BA / .ST only	—	×	
Constant	Float	$\pm 1.175494351e - 38$ to $\pm 3.402823466e + 38$	—	×
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	×
	Integer	-2147483648 to 2147483647	—	×

◆ **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 is 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	Variable Settings	Description
Variable Name.HR	Integer Variable	Hours are input in BCD.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.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 exceeds 00 (h):00 (min):00 (s) after the instruction is executed, an overflow will occur. The system variable (bit) #L_CalcCarry turns ON.
- (2) If the operation result is 00 (h):00 (min):00 (s), the system variable #L_CalcZero turns ON.

(Notes)

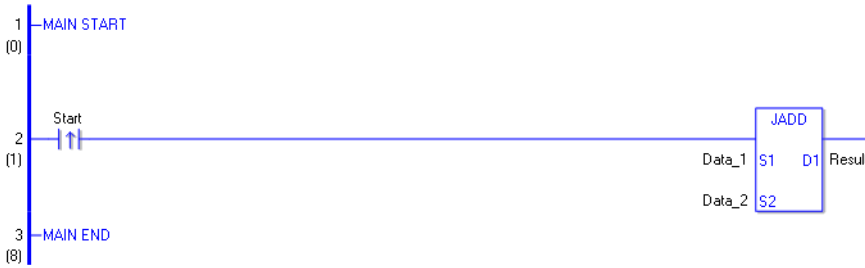
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

JADD

When the positive transition instruction turns ON, time addition is performed.

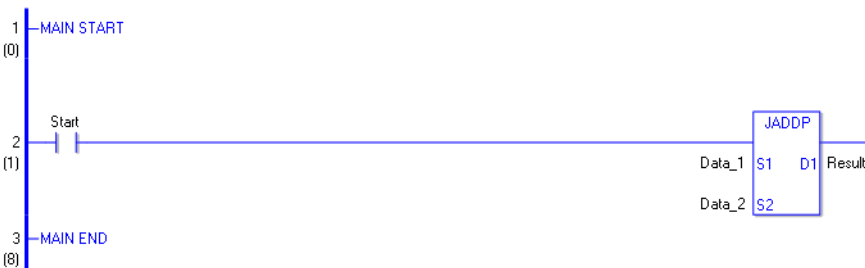


(1) When the positive transition instruction turns ON, the JADD instruction will be executed. When the JADD instruction is executed, data 1 (time variable) in operand S1 is added to data 2 (time variable), and the result of the addition is stored in operand D1. When using a Normally Open instruction, as long as the instruction variable is ON, the JADD instruction is always executed, and time addition is performed every time a scan is executed.

e.g. 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
<p>JSUB (Time Subtraction - level transition)</p>		Operation	4

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
<p>JSUBP (Time Subtraction - positive transition)</p>		Operation	4

◆ Operand Settings

The following table lists the configurable conditions for Operands (S1, S2, and D1) in the JSUB and JSUBP instructions.

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

Continued

Explanations of Each Instruction

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

Continued

Explanations of Each Instruction

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

◆ **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 is 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	Variable Settings	Description
Variable Name.HR	Integer Variable	Hours are input in BCD.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.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 is below 00 (h):00 (min):00 (s) after the instruction is executed, an overflow will occur. The system variable (bit) #L_CalcCarry turns ON.
- (2) If the operation result is 00 (h):00 (min):00 (s), the system variable #L_CalcZero turns ON.

(Notes)

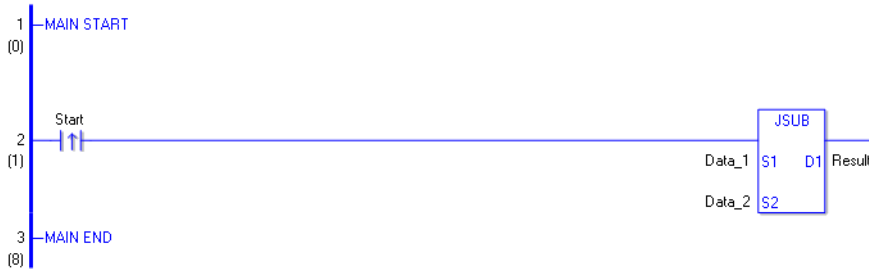
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

JSUB

When the positive transition instruction turns ON, time subtraction is performed.

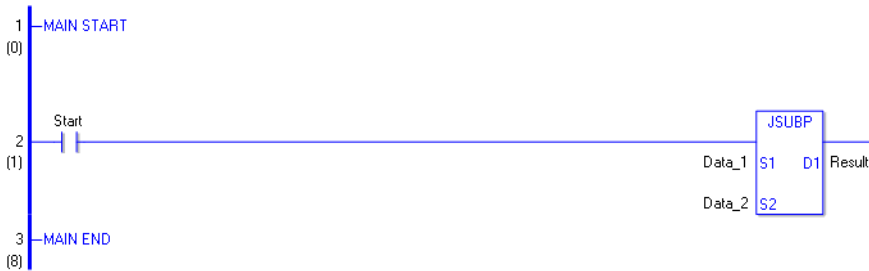


- (1) When the positive transition instruction turns ON, the JSUB instruction will be executed. When the JSUB instruction is executed, data 2 (time variable) in operand S2 is subtracted from data 1 (time variable) in operand S1, and the result of the subtraction is stored in operand D1. When using a Normally Open instruction, as long as the instruction variable is ON, the JSUB instruction is always executed, and time subtraction is performed every time a scan is executed.

e.g. 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




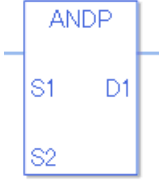
- (1) When the Normally Open instruction turns ON, the JSUBP instruction will be executed. When the JSUBP instruction is executed, data 2 (time variable) in operand S2 is subtracted from data 1 (time variable) in operand S1, and the result of the subtraction is stored in operand D1. Even when using a 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.

30.5.9 Operation (Logical)

■ AND and ANDP (Logical AND)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
AND (Logical AND - level transition)		Operation	4 to 13

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

◆ Operand Settings

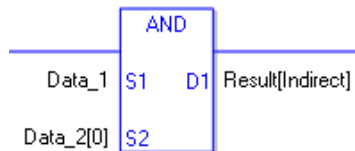
The following table lists the configurable conditions for Operands (S1, S2, and D1) in the AND and ANDP instructions.

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

e.g. 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 [indirect specification] = 3 Steps} + {1 step} = 7 steps.

The last 1 step is included in the total number of steps in the instruction. Be sure to add 1 step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1 and S2) in the AND and ANDP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	1	○	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○	
Symbol	Bit	—	—	×	
	Word	—	1	○	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant] Specify integer array (entire array)		2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		4	○
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		2	○
	Counter	.PV/.CV only		2	○
	Date	.YR/.MO/.DAY only		2	○
Time	.HR/.MIN/.SEC only		2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		2	○	

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the AND and ANDP instructions.

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

Continued

Explanations of Each Instruction

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

Continued

Explanations of Each Instruction

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

◆ **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 are logically ANDed and the result is stored in D1.

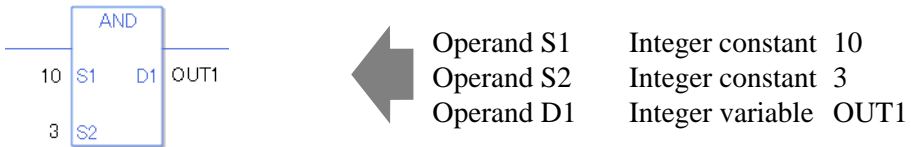
The AND and ANDP instructions always pass power. When using the AND and ANDP instructions, if the types of variables specified in the S1, S2, and D1 operands are not the same type, an error will occur. Specify the same variable type for the S1, S2, and D1 operands.

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

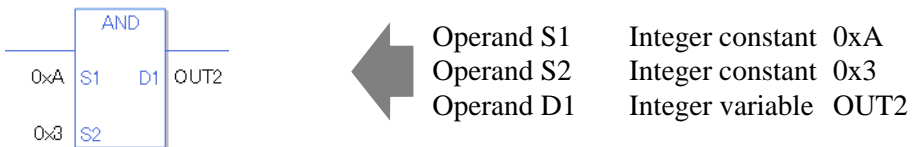
When an AND instruction is executed, the D1 bit turns ON only when S1 and S2 are ON. Otherwise, the D1 bit is OFF.

When operand D1 is an integer variable



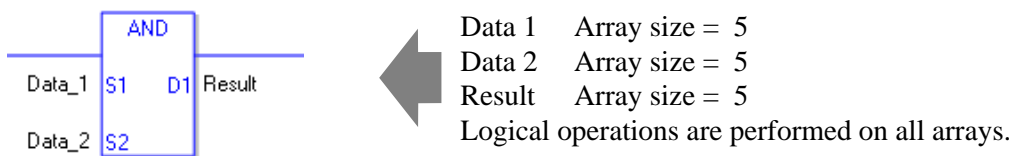
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

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

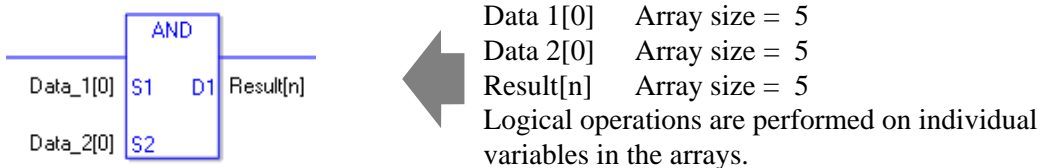


When Calculating Data in a Specified Array (Integer Variable Array)

Specifying the entire array



Individually Specifying Array Variables



◆ **Confirming Execution Results**

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

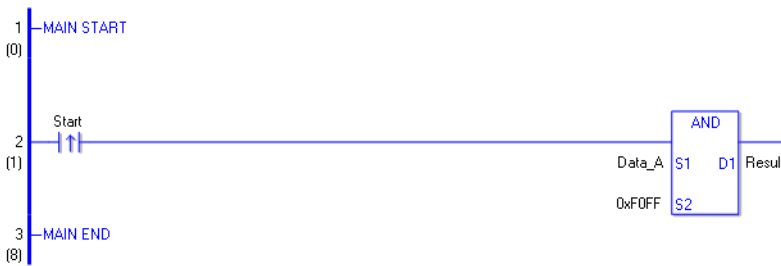
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

AND



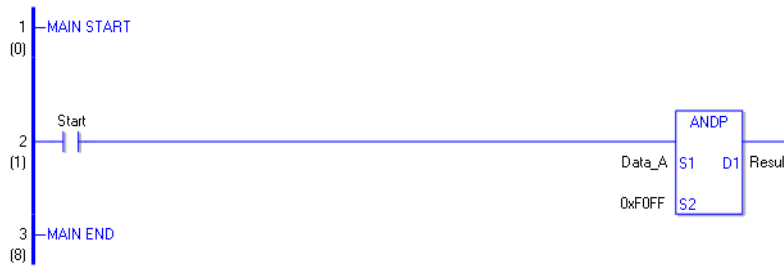
(1) When the positive transition instruction turns ON, the AND instruction will be executed.

When the AND instruction is executed, the result value obtained by ANDing data A with F0FF is stored in D1.

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

Program Example


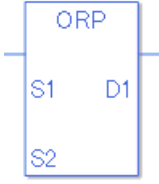
ANDP



- (1) When the normally open instruction turns ON, the ANDP instruction will be executed. When the ANDP instruction is executed, the result value obtained by ANDing data A with F0FF is stored in D1. Even when using a normally open instruction, only the upward transition is detected, and the ANDP instruction is executed. Therefore, even when the variable of the NO instruction is always ON, the ANDP instruction is executed only for one scan.

■ OR and ORP (Logical OR)

Symbols and Features

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

◆ Operand Settings

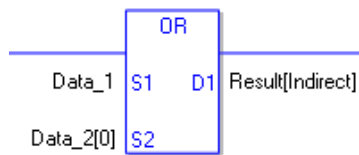
The following table lists the configurable conditions for Operands (S1, S2, and D1) in the OR and ORP instructions.

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

e.g. 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 [indirect specification] = 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 table lists the configurable conditions for Operands (S1 and S2) in the OR and ORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant] Specify integer array (entire array)	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
Time	.HR/.MIN/.SEC only	2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the OR and ORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (output only)		Arrays and modifiers are not specified	1	○
			Specify integer variable [constant] or Specify integer variable B/W [constant] Specify integer array (entire array)	2	○
			Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
			Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float		—	—	×
			Specify float variable [constant]	—	×
			Specify float variable [variable]	—	×
	Real		—	—	×
			Specify real variable [constant]	—	×
			Specify real variable [variable]	—	×
	Timer		.PT/.ET only	2	○
	Counter		.PV/ .CV only	2	○
	Date		.YR/ .MO/ .DAY only	2	○
	Time		.HR/ .MIN/ .SEC only	2	○
	PID		.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○
	Address Format	X_		—	×
Y_			—	×	
M_			—	×	
I_			—	×	
Q_			—	1	○
D_			Modifiers are not specified	1	○
			D_****.B/W [constant]	2	○
			D_****.B/W [address]	3	○
F_			—	—	×
R_			—	—	×
T_			.PT/.ET only	2	○
C_			.PV/ .CV only	2	○
N_			.YR/ .MO/ .DAY only	2	○
J_			.HR/ .MIN/ .SEC only	2	○
U_		.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	
Constant	Integer	-2147483648 to 2147483647	—	×	

◆ **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.

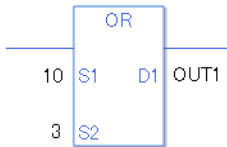
The OR and ORP instructions always pass power. When using the OR and ORP instructions, if the variables specified in operands S1, S2, and D1 are not the same type, an error will occur. Specify the same variable type in operands S1, S2, and D1.

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 turns ON only when S1 and S2 are ON. Otherwise, the D1 bit is OFF.

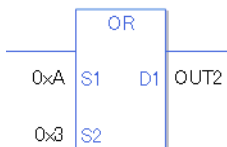
When operand D1 is an integer variable



← 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.

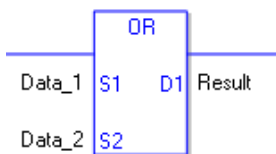
When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal values.



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

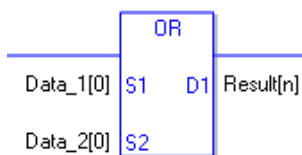
When Calculating Data in a Specified Array (Integer Variable Array)

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.

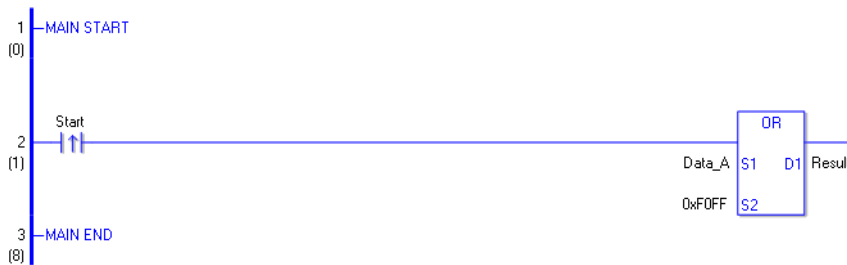
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

OR

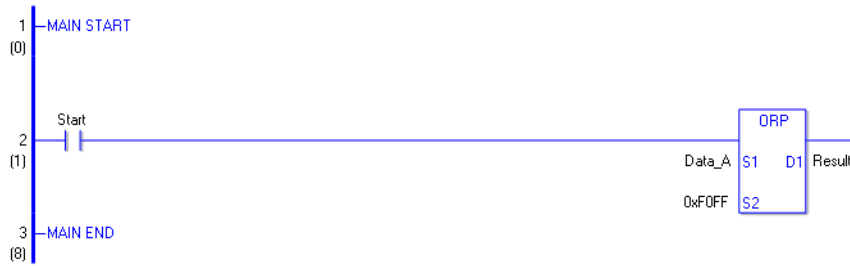


(1) When the positive transition instruction start turns ON, the OR instruction will be executed. When the OR instruction is executed, the result value obtained by ORing data A with F0FF is stored in D1.

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, only the upward transition is detected, and the ORP instruction is executed. 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 transition)		Operation	4 to 13
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
XORP (Logical XOR - positive transition)		Operation	4 to 13

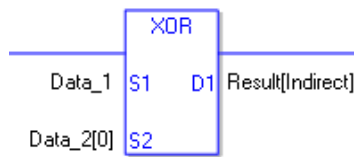
◆ **Operand Settings**

The following table lists the configurable conditions for 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

e.g. 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 [indirect specification] = 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 table lists the configurable conditions for Operands (S1 and S2) in the XOR and XORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○	
Symbol	Bit	—	—	×	
	Word	—	1	○	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant] Specify integer array (entire array)		2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		4	○
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		2	○
	Counter	.PV/.CV only		2	○
	Date	.YR/.MO/.DAY only		2	○
Time	.HR/.MIN/.SEC only		2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the XOR and XORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (output only)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant] Specify integer array (entire array)	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
Time	.HR/.MIN/.SEC only	2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	

Continued

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

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

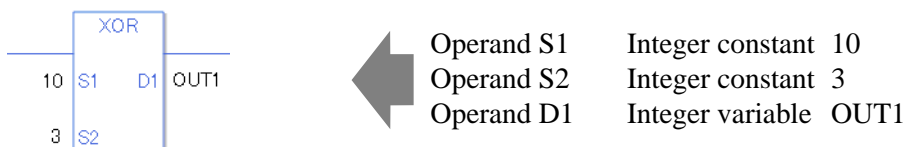
The XOR and XORP instructions are logical XOR instructions. When an XOR instruction is executed, S1 and S2 are logically XORed and the result is stored in D1. The XOR and XORP instructions always pass power. When using the XOR and XORP instructions, if the variables specified in operands S1, S2, and 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	XOR	OFF	OFF
ON		OFF	ON
OFF		ON	ON
ON		ON	OFF

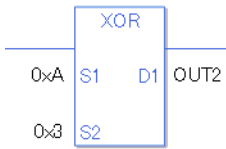
When an XOR instruction is executed, the D1 bit turns ON only when either S1 or S2 is ON. Otherwise, the D1 bit is OFF.

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.

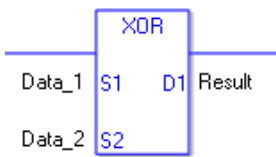
When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal values.



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

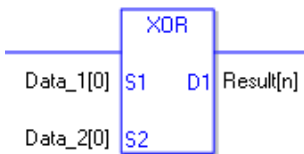
When Calculating Data in a Specified Array (Integer Variable Array)

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.

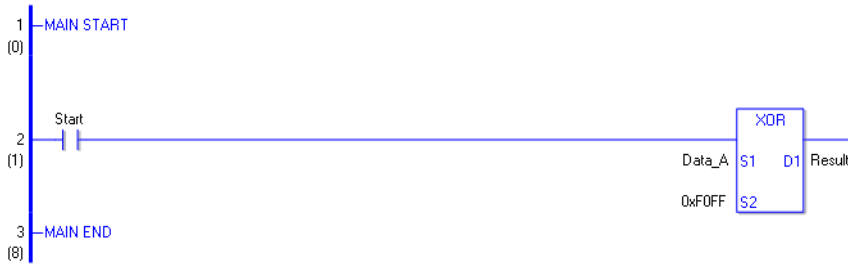
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

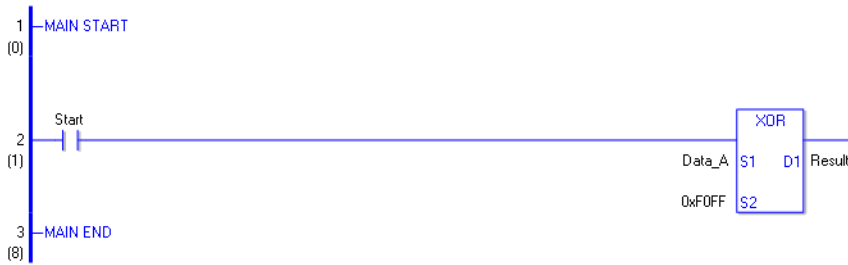
XOR



- (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, only the upward transition is detected, and the XORP instruction is executed. 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 transition)		Operation	3 to 9
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NOTP (Logical NOT - positive transition)		Operation	3 to 9

◆ Operand Settings

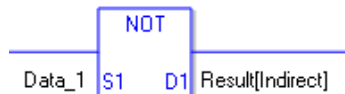
The following table lists the configurable conditions for Operands (S1 and D1) in the NOT and NOTP instructions.

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

e.g. 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.)



$$\{\text{Data 1} = 1 \text{ step}\} + \{\text{Result [indirect specification]} = 3 \text{ steps}\} + \{1 \text{ step}\} = 5 \text{ steps}$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in the NOT and NOTP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○	
Symbol	Bit	—	—	×	
	Word	—	1	○	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant] Specify integer array (entire array)		2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		4	○
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		2	○
	Counter	.PV/.CV only		2	○
Date	.YR/.MO/.DAY only		2	○	
Time	.HR/.MIN/.SEC only		2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		2	○	

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the NOT and NOTP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (output only)		Arrays and modifiers are not specified	1	○
			Specify integer variable [constant] or Specify integer variable B/W [constant] Specify integer array (entire array)	2	○
			Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
			Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float		—	—	×
			Specify float variable [constant]	—	×
			Specify float variable [variable]	—	×
	Real		—	—	×
			Specify real variable [constant]	—	×
			Specify real variable [variable]	—	×
	Timer		.PT/.ET only	2	○
	Counter		.PV/ .CV only	2	○
	Date		.YR/ .MO/ .DAY only	2	○
	Time		.HR/ .MIN/ .SEC only	2	○
	PID		.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○
	Address Format	X_		—	×
Y_			—	×	
M_			—	×	
I_			—	×	
Q_			—	1	○
D_			Modifiers are not specified	1	○
			D_****.B/W [constant]	2	○
			D_****.B/W [address]	3	○
F_			—	—	×
R_			—	—	×
T_			.PT/.ET only	2	○
C_			.PV/ .CV only	2	○
N_			.YR/ .MO/ .DAY only	2	○
J_			.HR/ .MIN/ .SEC only	2	○
U_		.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	
Constant	Integer	-2147483648 to 2147483647	—	×	

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

The NOT and NOTP instructions are logical NOT instructions. When the NOT instruction is executed, S1 is logically inverted and the inverted result is stored in D1. The NOT and NOTP instructions always pass power. When using the NOT and NOTP 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.

S1	Operator	D1
OFF	NOT	ON
ON		OFF

When an 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



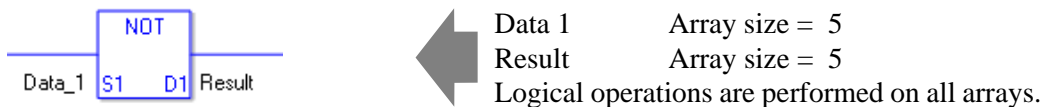
When operand D1 is an integer variable and you want to input a hexadecimal value in operand S1.

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



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

Specifying the entire array



Individually Specifying Array Variables



◆ **Confirming Execution Results**

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

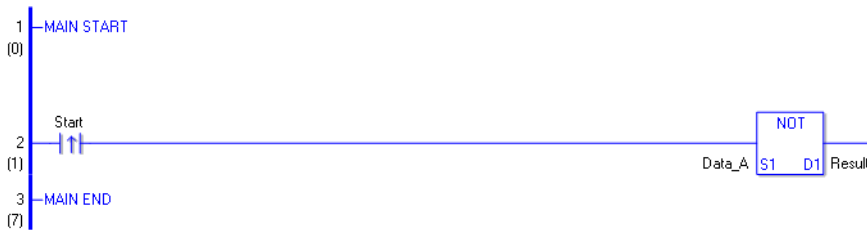
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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, only the upward transition is detected, and the NOTP instruction is executed.


Therefore, even when the variable of the NO instruction is always ON, the NOTP instruction is executed only for one scan.

30.5.10 Operation (Transfer)

■ MOV and MOVP (Transfer)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
MOV (Transfer - level transition)		Transfer	3 to 9

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

◆ Operand Settings

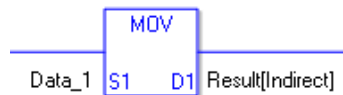
The following table lists the configurable conditions for Operands (S1, S2 and D1) in the MOV and MOVP instructions.

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

e.g. 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 step} + {Result [indirect specification] = 3 steps} + {1 step} = 5 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 table lists the configurable conditions for Operand (S1) in the MOV and MOVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the MOV and MOVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (output only)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
Time	.HR/.MIN/.SEC only	2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	—	×
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	2	○
		D_****.B/W [address]	3	○
	F_	—	1	○
	R_	—	1	○
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY only	2	○
J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Constant	—	—	—	×

◆ **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 are 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.



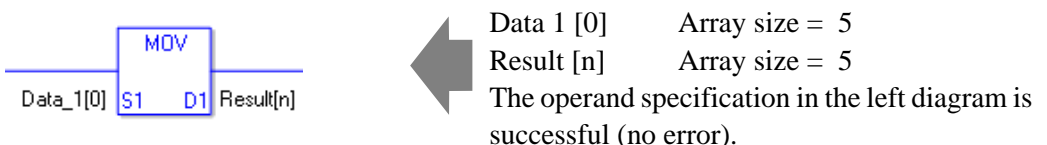
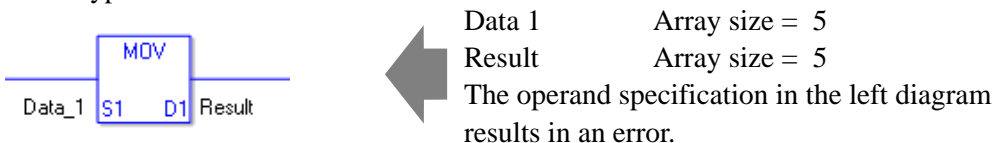
When operand D1 is a real variable

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.



◆ **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 execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

MOV

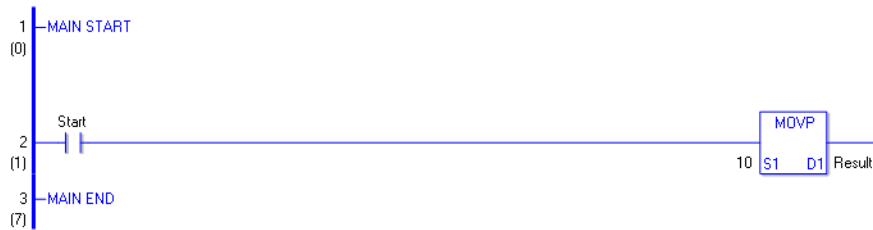
Stores the constant in the integer variable.



- (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, only the upward transition is detected, and the MOVP instruction is executed. 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
<p>BLMV (Block Transfer - level transition)</p>		Transfer	6 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
<p>BLMVP (Block Transfer - positive transition)</p>		Transfer	6 to 10

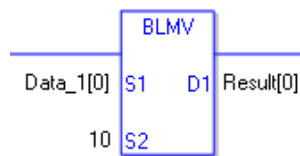
◆ **Operand Settings**

The following table lists the configurable conditions for 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 the Instruction

e.g. 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 table lists the configurable conditions for Operands (S1 and D1) in the BLMV and BLMVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	2	○	
		Specify bit array ([variable])	3	○	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		2	○
		Specify integer variable [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	—		—	×
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
	Time	.HR/.MIN/.SEC only		—	×
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S2) in for the BLMV and BLMVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○

Continued

Explanations of Each Instruction

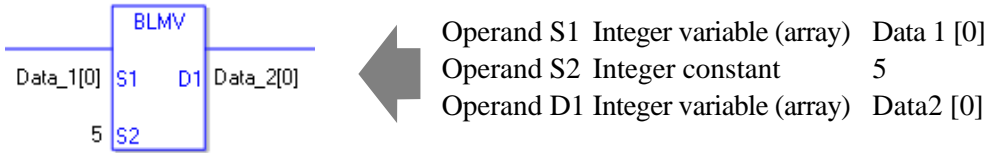
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)		Arrays and modifiers are not specified	1	○
			Specify integer variable [constant]	2	○
			Specify integer variable [variable]	3	○
			Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float		—	—	×
			Specify float variable [constant]	—	×
			Specify float variable [variable]	—	×
	Real		—	—	×
			Specify real variable [constant]	—	×
			Specify real variable [variable]	—	×
	Timer		.PT/.ET only	2	○
	Counter		.PV/ .CV only	2	○
Date		.YR/ .MO/ .DAY only	2	○	
Time		.HR/ .MIN/ .SEC only	2	○	
PID		.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	
Address Format	X_		—	×	
	Y_		—	×	
	M_		—	×	
	I_		—	1	○
	Q_		—	1	○
	D_		Modifiers are not specified	1	○
			D_****.B/W [constant]	—	×
			D_****.B/W [address]	—	×
	F_		—	—	×
	R_		—	—	×
	T_		.PT/.ET only	2	○
	C_		.PV/ .CV only	2	○
	N_		.YR/ .MO/ .DAY only	2	○
J_		.HR/ .MIN/ .SEC only	2	○	
U_		.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	
Constant		1 to 4096	1	○	

◆ **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.

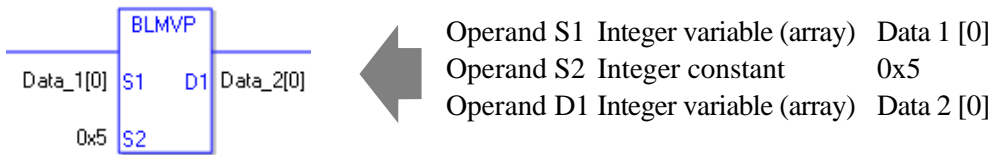
Refer to the following for specifying a constant.

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)

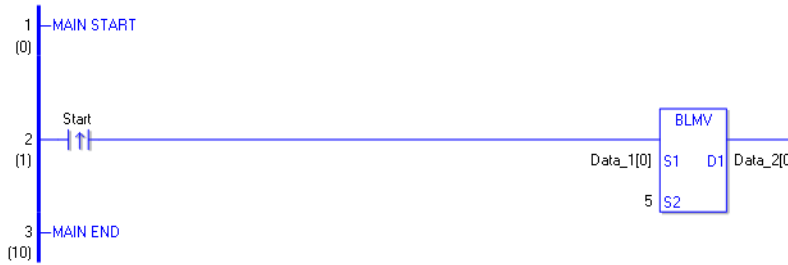
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

BLMV

Copies 1 through 5 from data 1 to data 2.

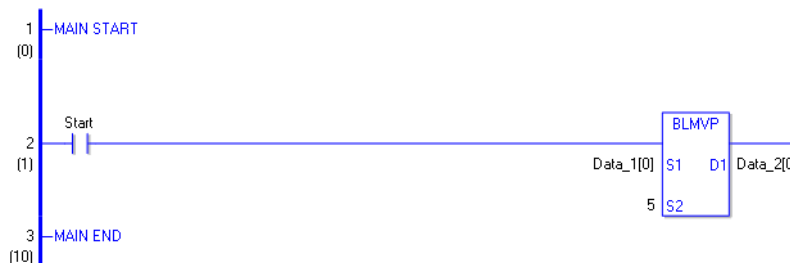


- (1) When the positive transition instruction turns ON, the BLMV instruction will be executed. When the BLMV instruction is executed, numbers 0 through 4 in data 1, stored in D1, are copied to numbers 0 through 4 in data 2. When the start is a normally open instruction, as long as the start is ON, the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

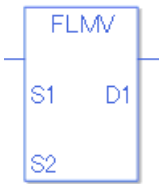
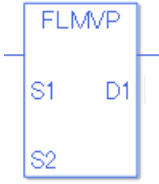
BLMVP



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

◆ Operand Settings

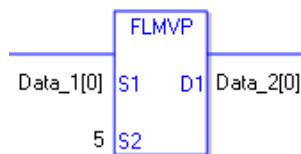
The following table lists the configurable conditions for Operands (S1, S2, and D1) in the FLMV and FLMVP instructions.

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 the Instruction

e.g. 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.)



{Data 1[0] = 2 steps} + {5 = 1 step} + {Data 2 [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 table lists the configurable conditions for Operand (S1) in the FLMV and FLMVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○

Continued

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S2) in the FLMV and FLMVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the FLMV and FLMVP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	×
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	×
Symbol	Bit	—	—	×
	Word	—	—	×
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (not including I/O)	Arrays and modifiers are not specified	—	×
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	—	×
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	—	×
	Counter	.PV/ .CV only	—	×
	Date	.YR/ .MO/ .DAY only	—	×
Time	.HR/ .MIN/ .SEC only	—	×	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	—	×	

Continued

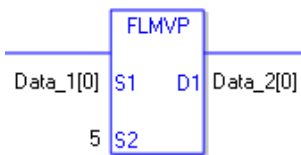
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified		1	○
		D_****.B/W [constant]		—	×
		D_****.B/W [address]		—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only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

◆ Explanation of the FLMV and FLMVP Instructions

The FLMV and FLMVP instructions are multipoint transfer instructions. When the FLMV instruction is executed, the data in S1 is stored in S2 elements of D1. The FLMV and FLMVP instructions always pass power. When using the FLMV and FLMVP 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.

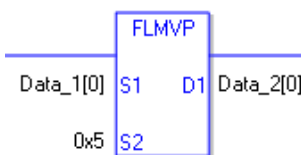
When operand D1 is an integer variable



Operand S1 Integer variable (array) Data 1 [0]
 Operand S2 Integer constant 5
 Operand D1 Integer variable (array) Data 2 [0]

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.



Operand S1 Integer variable (array) Data 1 [0]
 Operand S2 Integer constant 0x5
 Operand D1 Integer variable (array) Data 2 [0]

◆ **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)

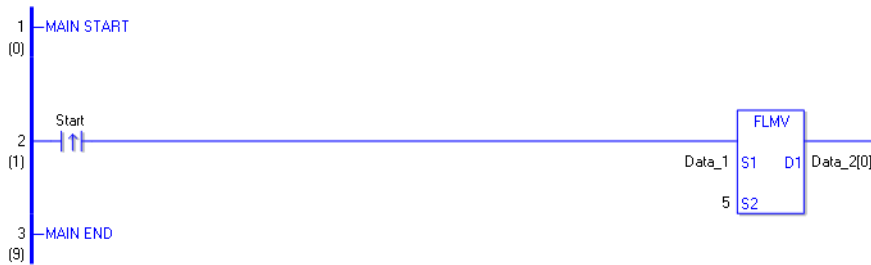
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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.

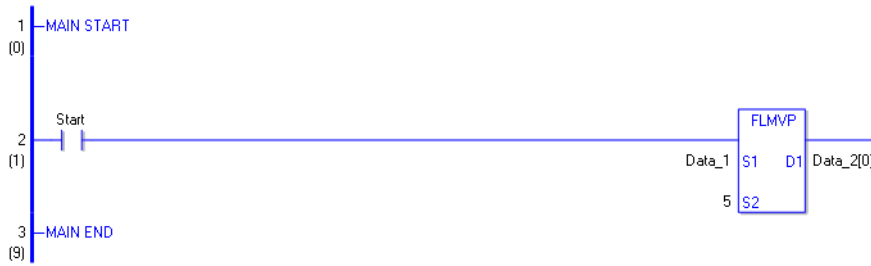
When the FLMV instruction is executed, data 1, stored in D1, is copied to elements 0 through 4 in data 2.

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

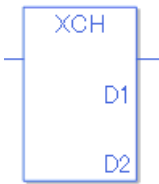
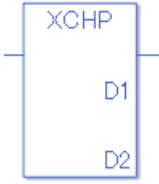
FLMVP



- (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, only the upward transition is detected, and the FLMVP instruction is executed. 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 transition)		Transfer	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
XCHP (Exchange - positive transition)		Transfer	3 to 7

◆ Operand Settings

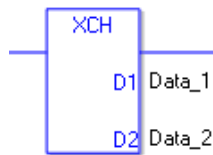
The following table lists the configurable conditions for Operands (D1 and D2) in the XCH and XCHP instructions.

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

e.g. 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 table lists the configurable conditions for Operands (S1 and D2) in the XCH and XCHP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (Output included)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

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

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

The XCH and XCHP instructions are data exchange instructions. When the XCH instruction is executed, the data in D1 and the data in D2 are exchanged.

The XCH and XCHP instructions always pass power. When using the XCH and XCHP instructions, if the variables specified in operands D1 and D2 are not the same type, an error will occur. Specify the same variable type for operands D1 and D2.

◆ **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 results in D1 and D2 keep their previous values with which the instruction was executed successfully.

(Notes)

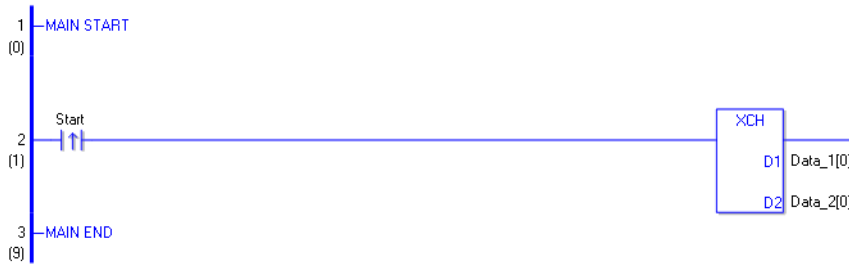
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

XCH

Exchanges the contents of data 1 and data 2.

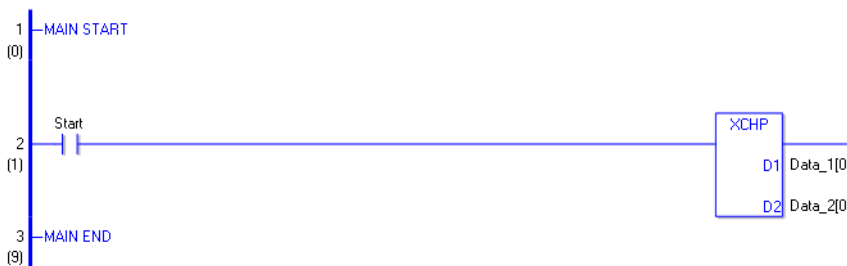


- (1) When the positive transition instruction turns ON, the XCH instruction will be executed. When the XCH instruction is executed, the contents of data 1[0] in D1 is exchanged with the contents of data 2[0]. When using a normally open instruction, as long as start is ON, the FLMV instruction is always 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

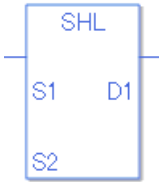
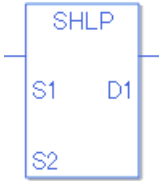


- (1) When the normally open instruction turns ON, the XCHP instruction will be executed. When the XCHP instruction is executed, the contents of data 1[0] in D1 is exchanged with the contents of data 2[0]. Even when using a normally open instruction, only the upward transition is detected, and the XCHP instruction is executed. Therefore, even when the variable of the NO instruction is always ON, the XCHP instruction is executed only for one scan.

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

◆ Operand Settings

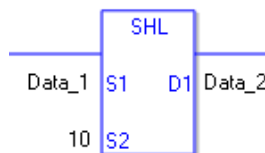
The following table lists the specifiable content of operands S1, S2, and D1 for the SHL and SHLP instructions.

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

e.g. 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.)



$$\{\text{Data 1} = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{\text{Data 2} = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in SHL and SHLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or entire array	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S2) in SHL and SHLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	—
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in SHL and SHLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (Output included)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or entire array	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	—
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ Explanation of the SHL and SHLP Instructions

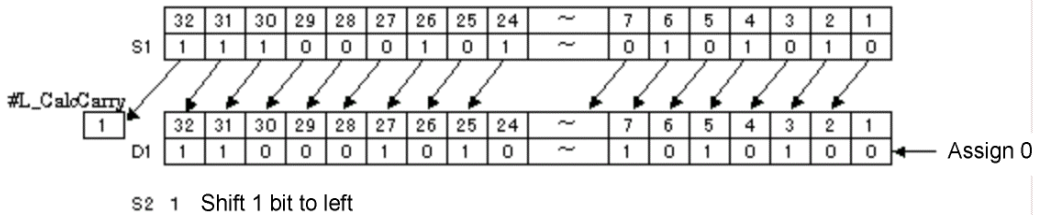
When an SHL or SHLP instruction is executed, the S1 bits are shifted to the left S2 number of bits. Every time 1 bit is shifted, the leftmost bit (the most significant bit) is lost. 0 is stored in the rightmost empty bit. The results are stored in D1.

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.

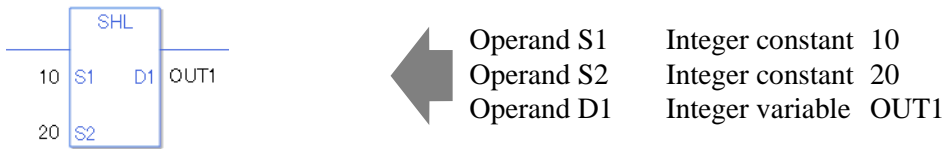
Refer to the following for specifying a constant.

- S1: Shift address Specifies an address whose bits are shifted.
- S2: Number of bits to be shift Specifies the number of bits to shift.
- D1: Storage address Specifies an address for storing the results after bits are shifted.

e.g. When 1 bit is shifted left

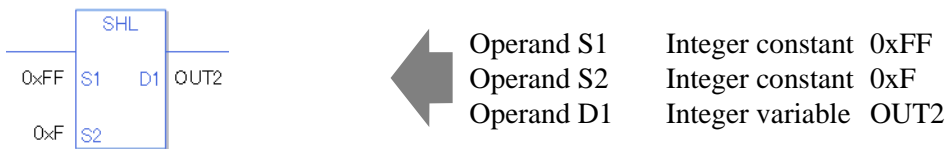


When operand D1 is an integer variable



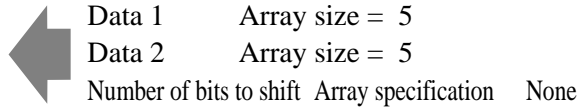
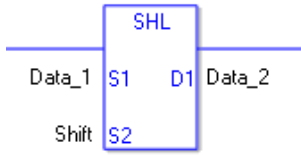
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

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



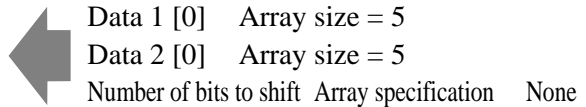
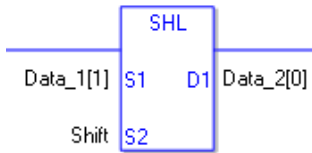
Use the same format when shifting data in a specified array (integer variable array) and when specifying an array element.

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 ($32 \times \text{array size} - 1$).



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 result of the execution is 0, #L_CalcZero turns ON.

The final bit value which overflowed as a result of the shift is stored in #L_CalcCarry.

If the execution results in an error, #L_Status stores the error information.

If the execution results in an error, #L_CalcErrCode stores the error code.

(Notes)

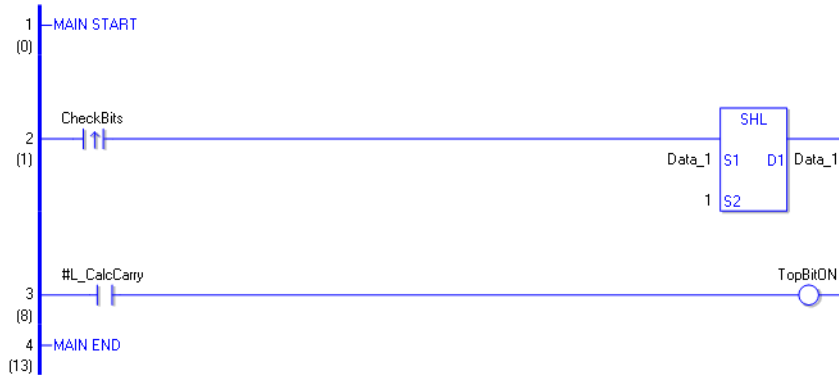
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

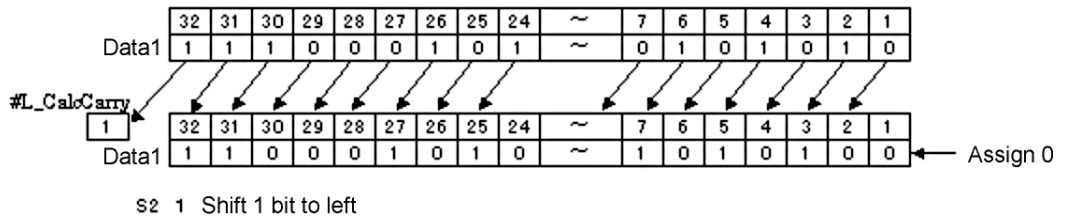
Program Example

SHL

Determines whether the most significant bit is ON or OFF.

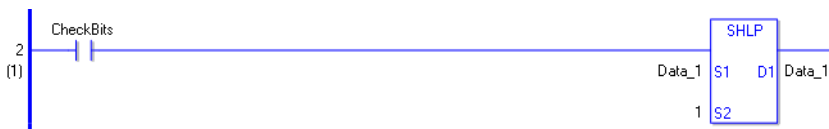


- (1) When the positive transition instruction turns ON, an SHL instruction will be executed. When the SHL instruction is executed, the result from shifting 1 bit to the left is stored in D1.
- (2) When 1 bit is shifted to the left, you can check whether the most significant bit before data shifting 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

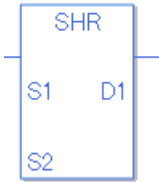
SHLP




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 transition)		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.

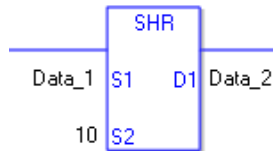
The number of steps in the SHR and SHRP instructions depends on the specified operand.

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

e.g. 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.)



$$\{\text{Data 1} = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{\text{Data 2} = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in SHR and SHRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or entire array	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S2) in SHR and SHRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in SHR and SHRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (Output included)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or entire array	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
Time	.HR/.MIN/.SEC only	2	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Explanation of SHR and SHRP Instructions**

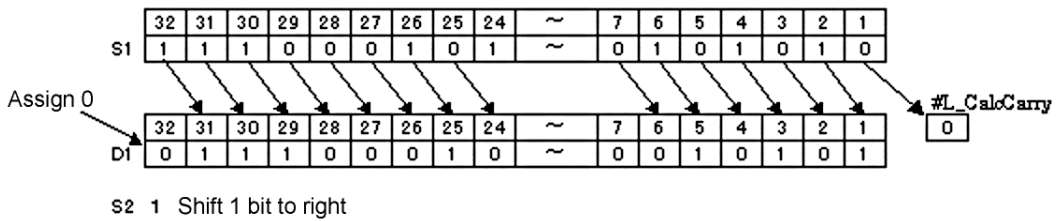
When the SHR or SHRP instruction is executed, the S1 bits are shifted to the right S2 number of bits. Every time 1 bit is shifted, the rightmost bit (least significant bit) is lost. 0 is stored in the empty topmost bit positions. The result is stored in D1.

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.

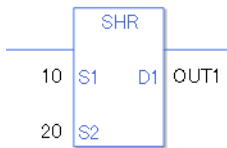
Refer to the following for specifying a constant.

- S1: Shift address Specifies an address whose bits are shifted.
- S2: Number of bits to be shift Specifies the number of bits to shift.
- D1: Storage address Specifies an address for storing the results after bits are shifted.

e.g. When 1 bit is shifted to the right



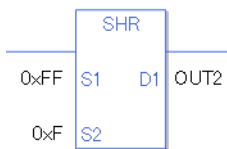
When operand D1 is an integer variable



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

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

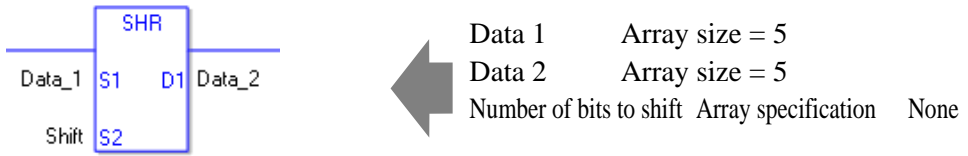
When 0x (zero and lower case x) is input, the following values are interpreted as hexadecimal values.



- Operand S1 Integer constant 0xFF
- Operand S2 Integer constant 0xF
- Operand D1 Integer variable OUT2

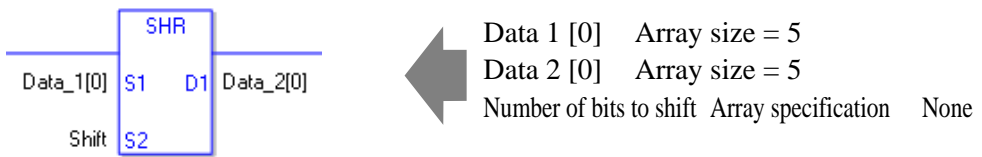
Use the same format when shifting data in a specified array (integer variable array) and when specifying an array element.

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 $(32 \times \text{array size} - 1)$.



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 result of the execution is 0, #L_CalcZero turns ON.

The final bit value which overflowed as a result of the shift is stored in #L_CalcCarry.

If the execution results in an error, #L_Status stores the error information.

If the execution results in an error, #L_CalcErrCode stores the error code.

(Notes)

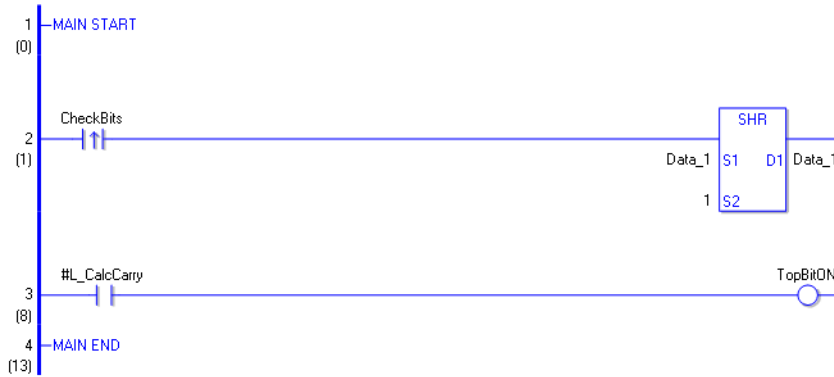
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

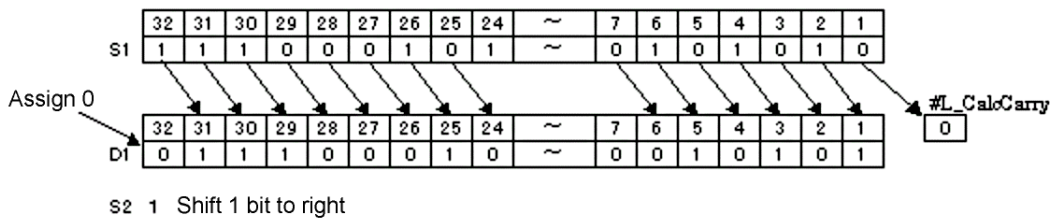
Program Example

SHR

Determines whether the least significant bit is ON or OFF.

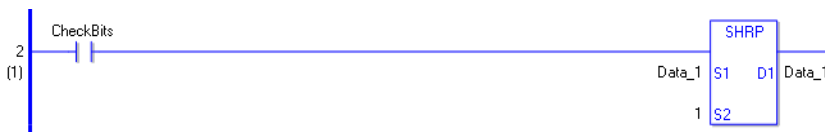


- (1) When the variable of the positive transition instruction turns ON, the SHR instruction is run. The SHR instruction shifts bits in Data1 one bit to the right and stores the result in D1.
- (2) After the bit shift operation is complete, you can check the previous value of the least significant bit in Data1 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 transition)		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.

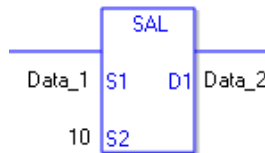
The number of steps in the SAL and SALP instructions depends on the specified operand.

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

e.g. 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.)



$$\{\text{Data 1} = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{\text{Data 2} = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in SAL and SALP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S2) in SAL and SALP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV only	2	○	
	N_	.YR/.MO/.DAY only	2	○	
	J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○		
Constant	Float	±1.175494351e-38 to ±3.402823466e+38	—	×	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	×	
	Integer	0 to 31	1	○	

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in SAL and SALP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (Output included)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ Explanation of the SAL and SALP Instructions

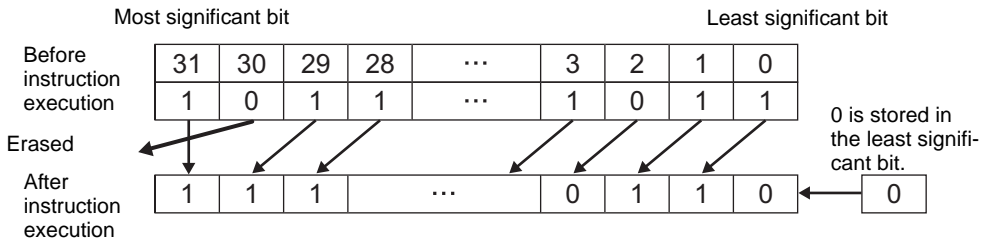
When the SAL or SALP instruction is executed, the S1 bits are shifted to the left S2 number of bits. Every time 1 bit is shifted, the 30th bit is lost. 0 is stored in the bottom-most empty bit. The result is stored in D1.

The SAL and SALP instructions always pass power. When using SAL and SALP 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.

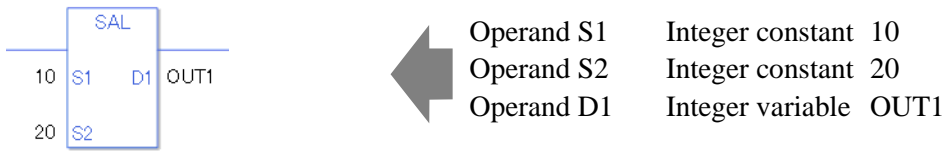
Refer to the following for specifying a constant.

- S1: Shift address Specifies an address whose bits are shifted.
- S2: Number of bits to be shift Specifies the number of bits to shift.
- D1: Storage address Specifies an address for storing the results after bits are shifted.

e.g. When 1 bit is shifted left

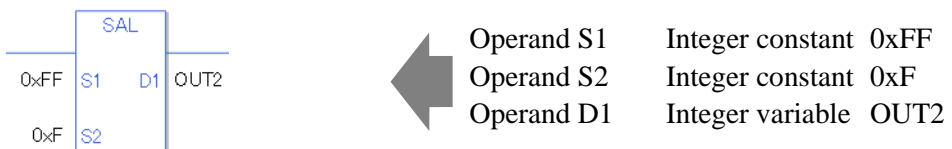


When operand D1 is an integer variable

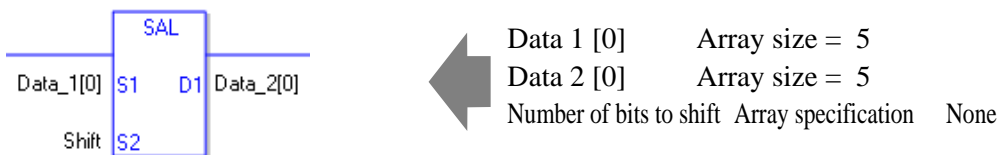


When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values are 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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_Status stores the error information.

If the execution results in an error, #L_CalcErrCode stores the error code.

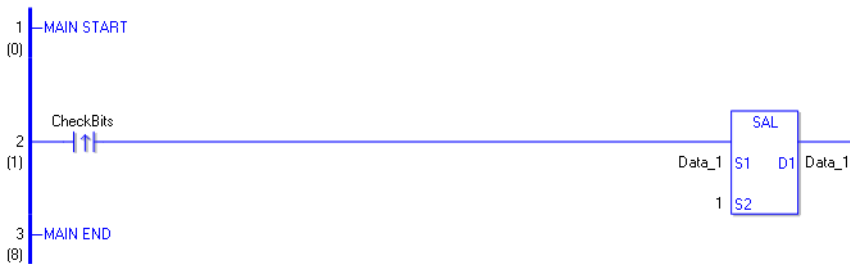
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

SAL

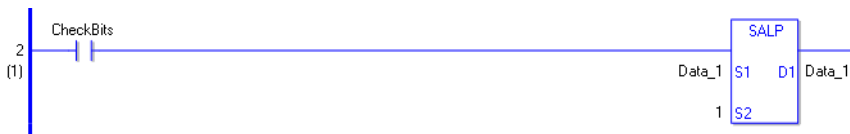


(1) When the positive transition instruction turns ON, the SAL instruction is executed. When the SAL instruction is executed, the result of the bit shift is stored in D1. The most significant bit is not shifted, and zero is stored in the least significant bit.

(Supplementary) When the using a normally open instruction, the SHR instruction is always executed as long as the normally open bit is ON.

Program Example

SALP




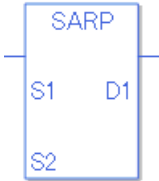
The SALP and SAL instructions have different ways of detecting when to execute. In the SALP instruction, even when using a normally open instruction, only the upward transition is detected, and the SALP instruction is executed.

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 transition)		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

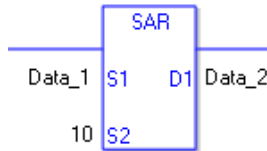
The following table lists the specifiable content of operands S1, S2, and D1 for the SAR and SARP instructions.

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

e.g. 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.)



$$\{Data\ 1 = 1\ step\} + \{10 = 1\ step\} + \{Data\ 2 = 1\ step\} + \{1\ step\} = 4\ steps$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in SAR and SARP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S2) in SAR and SARP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV only	2	○	
	N_	.YR/.MO/.DAY only	2	○	
	J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○		
Constant	Float	±1.175494351e-38 to ±3.402823466e+38	—	×	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	×	
	Integer	0 to 31	1	○	

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in SAR and SARP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (Output included)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

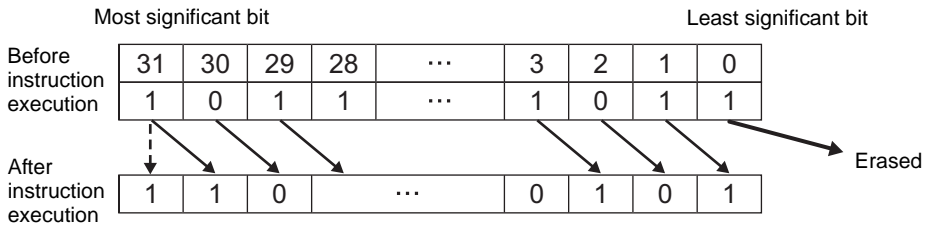
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV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

◆ **Explanation of the SAR and SARP Instructions**

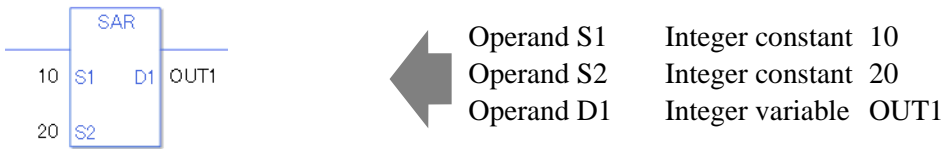
When the SAR or SARP instruction is executed, the S1 bits are shifted to the right of the S2 number of bits. For each bit shift, the bottom-most bit (the least significant bit) is lost, and the most significant bit is stored in the topmost empty bit. The result is stored in D1. The SAR and SARP instructions 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 an address whose bits are shifted.
- S2: Number of bits to shift Specifies the number of bits to shift.
- D1: Storage address Specifies an address for storing the results after bits are shifted.

e.g. When 1 bit is shifted to the right

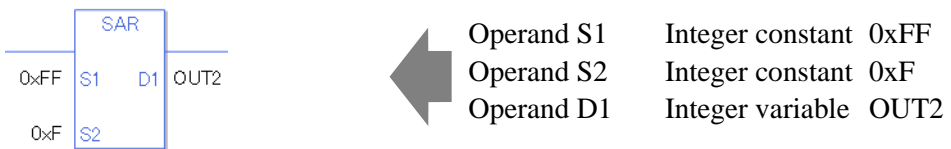


When operand D1 is an integer variable

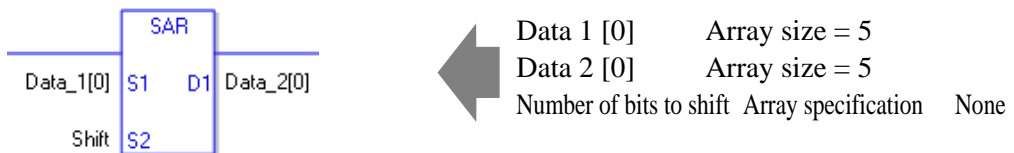


When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

When 0x (zero and lower case x) is input, the following values are 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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_Status stores the error information.

If the execution results in an error, #L_CalcErrCode stores the error code.

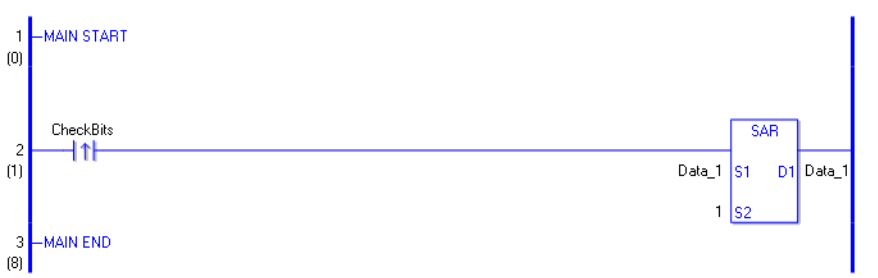
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

SAR



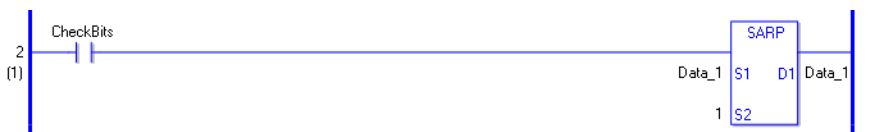
(1) When the positive transition instruction turns ON, the SAR instruction will be executed.

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.

30.5.12 Operation (Rotation Instruction)

■ ROL and ROLP (Rotate Left)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ROL (Rotate Left - level transition)		Rotate	4 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ROLP (Rotate Left - positive transition)		Rotate	4 to 10

◆ Operand Settings

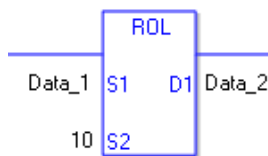
The following table lists the specifiable content of operands S1, S2, and D1 for the ROL and ROLP instructions.

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 the Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ step\} + \{10 = 1\ step\} + \{Data\ 2 = 1\ step\} + \{1\ step\} = 4\ steps$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in ROL and ROLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or entire array	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S2) in ROL and ROLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in ROL and ROLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (Output included)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or entire array	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

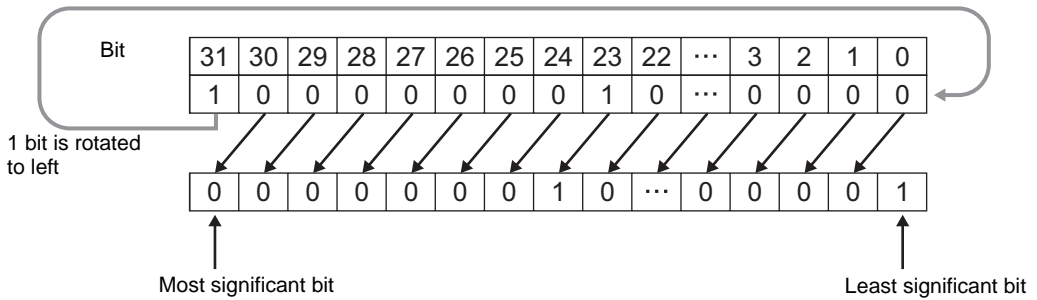
◆ **Explanation of the ROL and ROLP Instructions**

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.

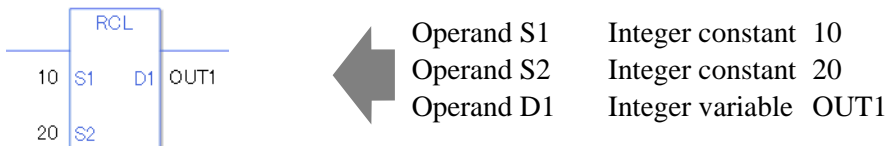
Refer to the following for specifying a constant.

- S1: Rotation device Specifies a device to rotate bits.
- S2: Number of bits to rotate Specifies the number of bits to rotate.
- D1: Storage device Specifies a device for storing the result after bits are rotated.

e.g. When 1 bit is rotated to left

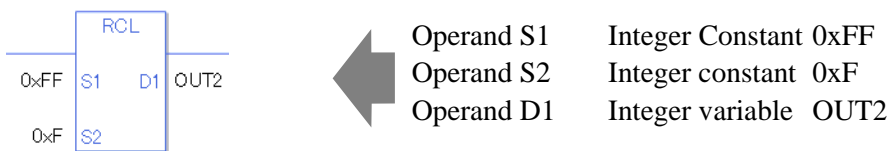


When operand D1 is an integer variable



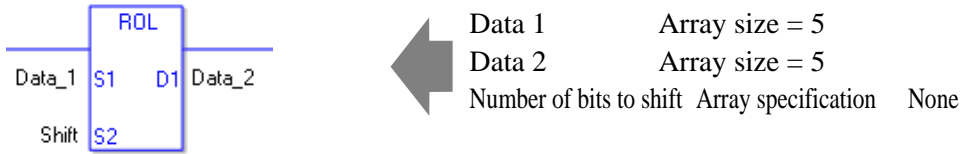
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

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



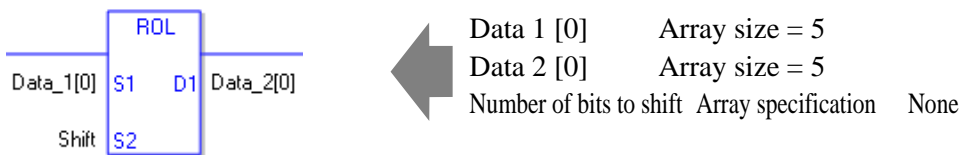
Use the same format when rotating data in a specified array (integer variable array) and when specifying an array element.

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 $(32 \times \text{array size} - 1)$.



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 result of the execution is 0, #L_CalcZero turns ON.

The final bit value which overflowed as a result of the rotation is stored in #L_CalcCarry.

If the execution results in an error, #L_Status stores the error information.

If the execution results in an error, #L_CalcErrCode stores the error code.


(Notes)


When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

■ ROR and RORP (Rotate Right)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ROR (Rotate Right - level transition)		Rotate	4 to 10

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
RORP (Rotate Right - positive transition)		Rotate	4 to 10

◆ Operand Settings

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

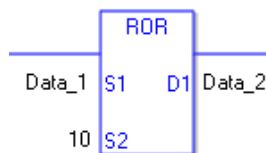
The number of steps in the ROR and RORP instructions depends on the specified operand.

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 the Instruction

e.g. 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.)



$$\{\text{Data 1} = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{\text{Data 2} = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in ROR and RORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or entire array	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S2) in ROR and RORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in ROR and RORP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (Output included)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or entire array	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

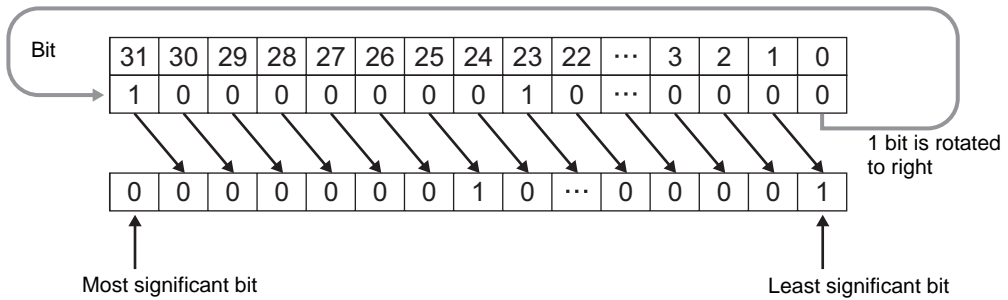
◆ **Explanation of the ROR and RORP Instructions**

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.

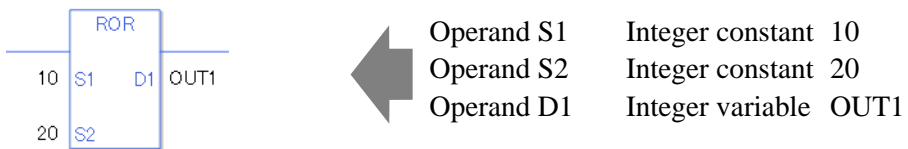
The result is stored in D1. The ROR and RORP instructions always pass power. When using the ROR and RORP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in the S1 and D1 operands. Refer to the following for specifying a constant.

- S1: Rotation device Specifies a device to rotate bits.
- S2: Number of bits to rotate Specifies the number of bits to rotate.
- D1: Storage device Specifies a device for storing the result after bits are rotated.

e.g. When 1 bit is rotated to right

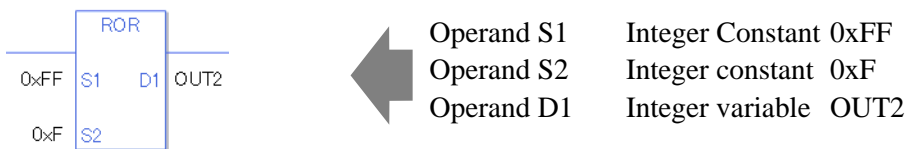


When operand D1 is an integer variable



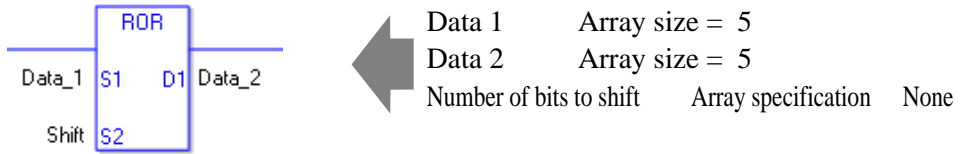
When operand D1 is an integer variable and you want to input hexadecimal values in operands S1 and S2.

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



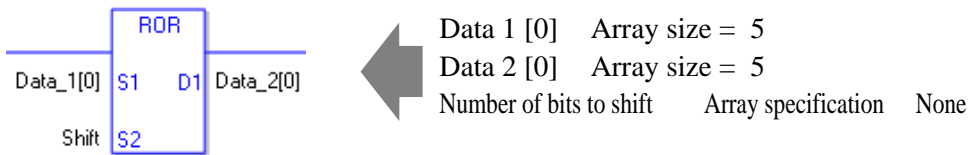
Use the same format when rotating data in a specified array (integer variable array) and when specifying an array element.

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 × 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 result of the execution is 0, #L_CalcZero turns ON.

The final bit value which overflowed as a result of the rotation is stored in #L_CalcCarry.

If the execution results in an error, #L_Status stores the error information.

If the execution results in an error, #L_CalcErrCode stores the error code.

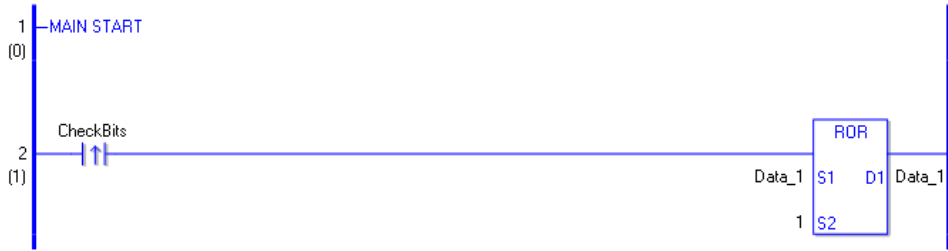
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

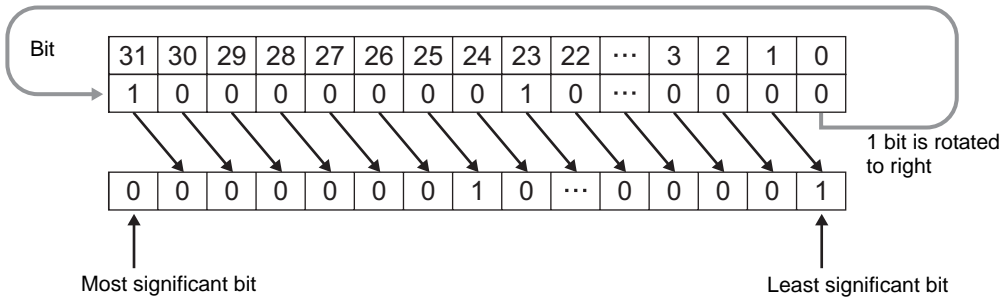
Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

ROR

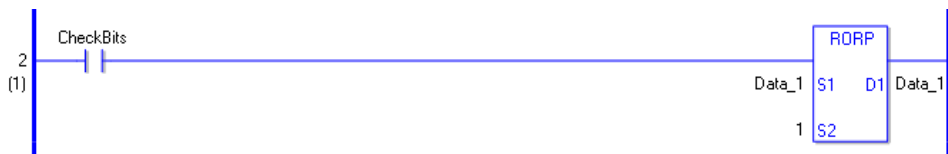


(1) When the positive transition instruction turns ON, the ROR instruction is executed. When the ROR instruction is executed, the result of rotating 1 bit to the right is stored in D1. (Supplementary) When using a normally open instruction, the ROR instruction is always executed as long as the bit is ON.



Program Example


RORP

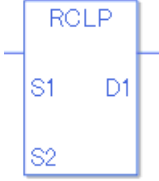


The RORP and ROR instructions have different ways of detecting when to execute. For RORP, even when using a normally open instruction, only the upward transition is detected, and the RORP instruction is executed. Therefore, the RORP instruction is executed only for one scan, even when the bit confirmation 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 transition)		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

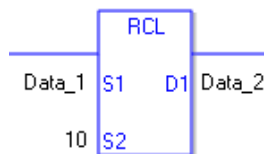
◆ Operand Settings

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 the Instruction

e.g. 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.)



$$\{\text{Data 1} = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{\text{Data 2} = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in RCL and RCLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S2) in RCL and RCLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV only	2	○	
	N_	.YR/.MO/.DAY only	2	○	
	J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○		
Constant	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	×	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	×	
	Integer	0 to 32	1	○	

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in RCL and RCLP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (Output included)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

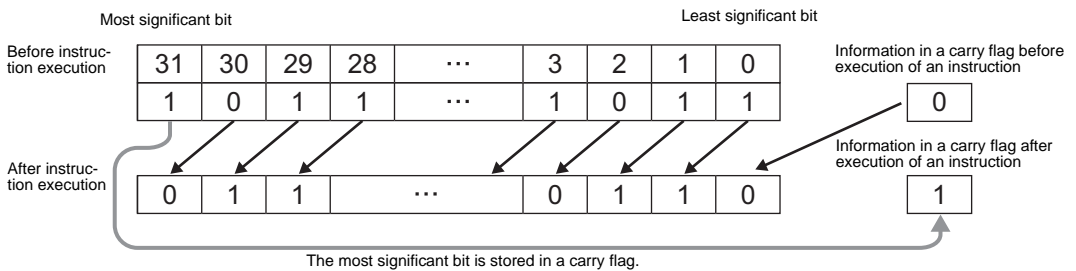
◆ **Explanation of the RCL and RCLP Instructions**

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).

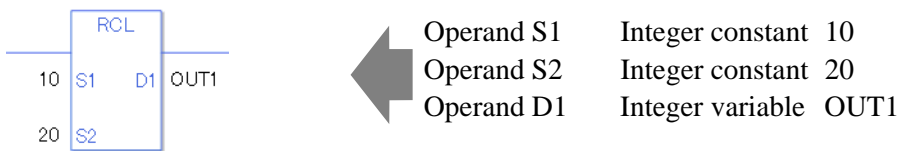
The result is stored in D1. The RCL and RCLP instructions always pass power. When using the RCL and RCLP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in the S1 and D1 operands. Refer to the following for specifying a constant.

- S1: Rotation device Specifies a device to rotate bits.
- S2: Number of bits to rotate Specifies the number of bits to rotate.
- D1: Storage device Specifies a device for storing the result after bits are rotated.

e.g. When 1 bit is rotated to left (with carry over)

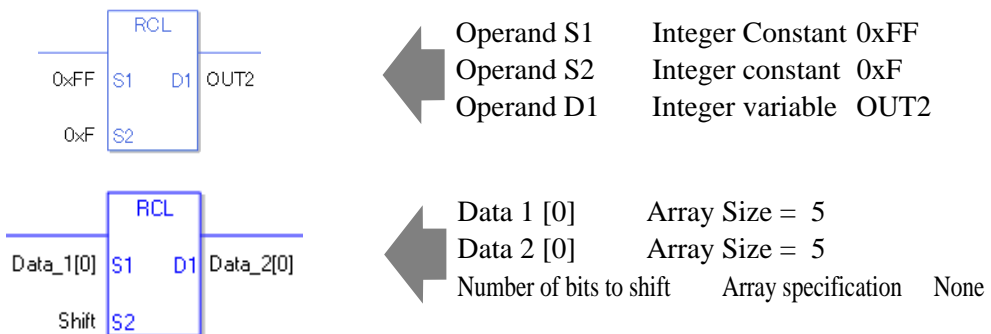


When operand D1 is an integer variable



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 result of the execution is 0, #L_CalcZero turns ON.

The final bit value which overflowed as a result of the rotation is stored in #L_CalcCarry.

If the execution results in an error, #L_Status stores the error information.

If the execution results in an error, #L_CalcErrCode stores the error code.

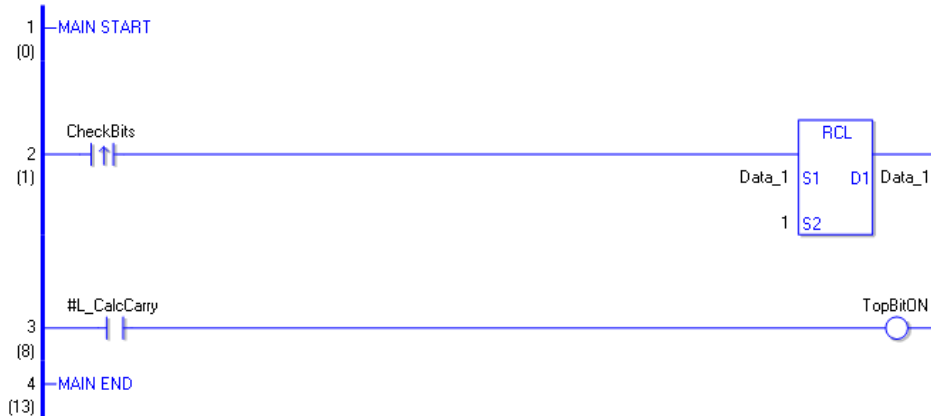
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

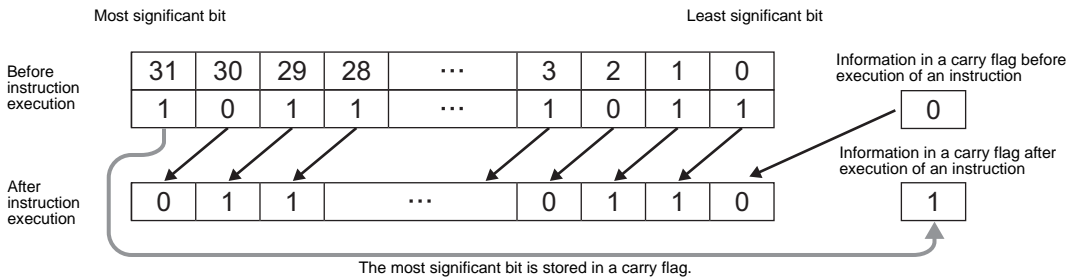
Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

RCL

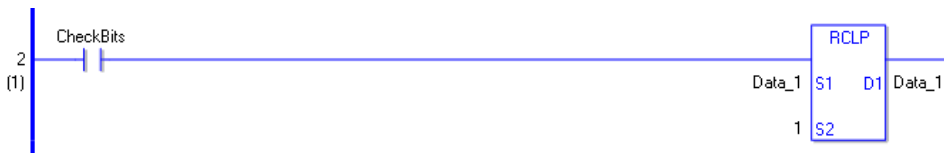


- (1) When the positive transition instruction turns ON, the RCL instruction is executed. When the RCL instruction is executed, the result from rotating 1 bit with carry over is stored in D1.
- (2) When 1 bit is shifted to the left with carry over, you can use #L_CalcCarry to check the value of the most 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, only the upward transition is detected, and the RCLP instruction is executed. 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 transition)		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

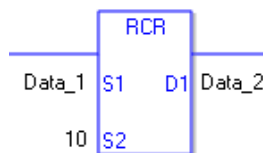
The following table lists the specifiable content of operands S1, S2, and D1 for the RCR and RCRP instructions.

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 the Instruction

e.g. 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.)



$$\{\text{Data 1} = 1 \text{ step}\} + \{10 = 1 \text{ step}\} + \{\text{Data 2} = 1 \text{ step}\} + \{1 \text{ step}\} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in RCR and RCRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S2) in RCR and RCRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV only	2	○	
	N_	.YR/.MO/.DAY only	2	○	
	J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○		
Constant	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	×	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	×	
	Integer	0 to 32	1	○	

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in RCR and RCRP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (Output included)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

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

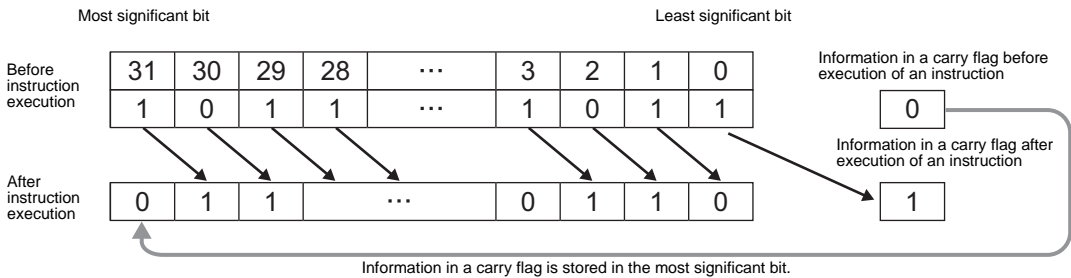
◆ **Explanation of the RCR and RCRP Instructions**

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).

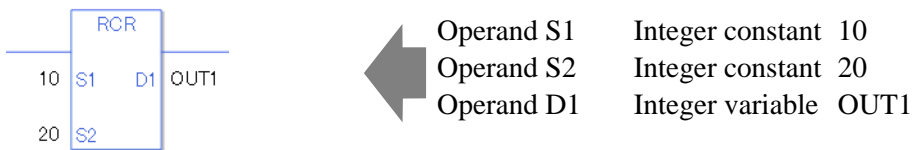
The result is stored in D1. The RCR and RCRP instructions always pass power. When using the RCR and RCRP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same type. Specify the same variable type in the S1 and D1 operands. Refer to the following for specifying a constant.

- S1: Rotation device Specifies a device to rotate bits.
- S2: Number of bits to rotate Specifies the number of bits to rotate.
- D1: Storage device Specifies a device for storing the result after bits are rotated.

e.g. When 1 bit is rotated to right (with carry over)

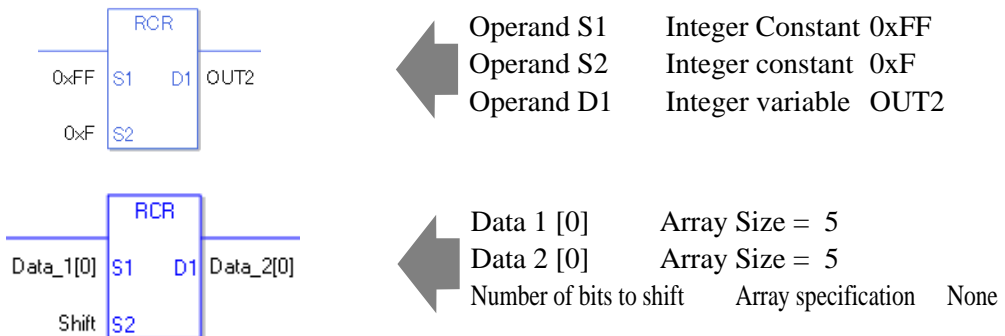


When operand D1 is an integer variable



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 result of the execution is 0, #L_CalcZero turns ON.

The bit value which overflowed as a result of the shift is stored in #L_CalcCarry.

If the execution results in an error, #L_Status stores the error information.

If the execution results in an error, #L_CalcErrCode stores the error code.

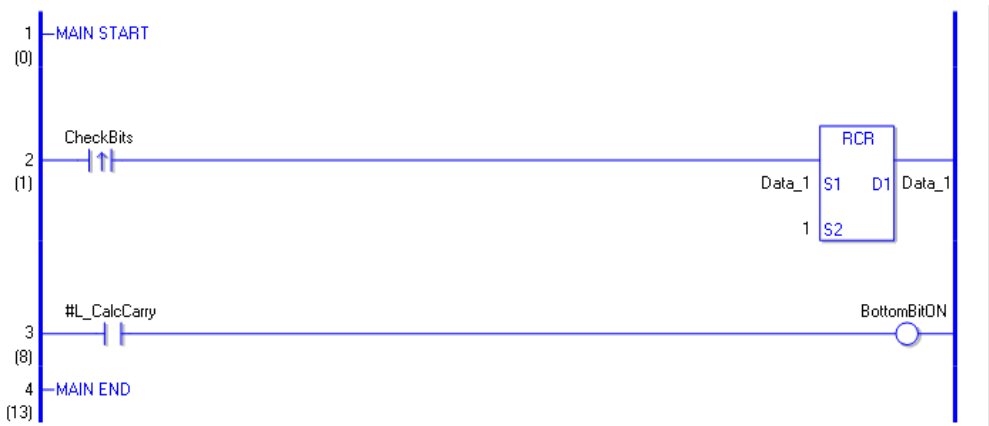
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

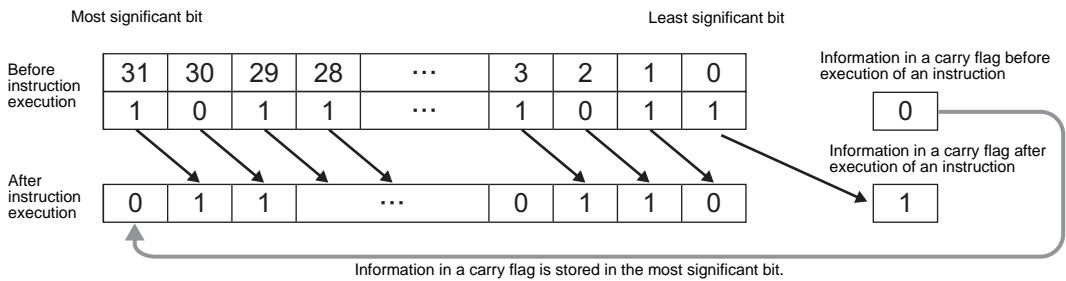
Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

RCR

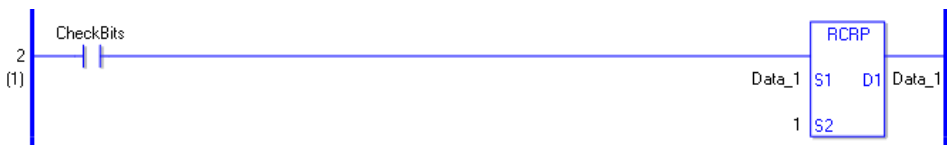


- (1) When the positive transition instruction turns ON, the RCR instruction will be executed. When the RCR instruction is executed, the result of rotating 1 bit with carry over is stored in D1.
- (2) When 1 bit is shifted to the right with carry over, you can use #L_CalcCarry to check the value of the 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.

30.5.13 Function Instruction

■ SUM/SUMP (Total)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SUM (Total - level transition)		Operation	6 to 10
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SUMP (Total - positive transition)		Operation	6 to 10

◆ Operand Settings

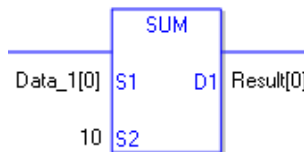
The following table lists the specifiable contents of operands S1, S2, D1 for the SUM/SUMP instructions.

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 the Instruction

e.g. 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 Steps} + {10= 1 Step} + {Result [0]= 2 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 lists the configurable conditions for Operand S1 in the SUM/SUMP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		2	○
		Specify integer variable [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	—		—	×
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand S2 in the SUM/SUMP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV only	2	○	
	N_	.YR/.MO/.DAY only	2	○	
	J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○		
Constant	—	1 to 4096	1	○	

◆ **Operand Settings**

The following table lists the configurable conditions for Operand D1 in the SUM/SUMP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (Output included)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		2	○
		Specify integer variable [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	—		—	×
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

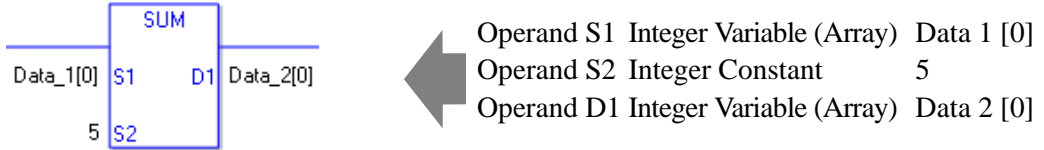
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	—	×
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	—	×
		D_****.B/W [address]	—	×
	F_	—	1	○
	R_	—	1	○
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY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

◆ **Explanation of the SUM 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.

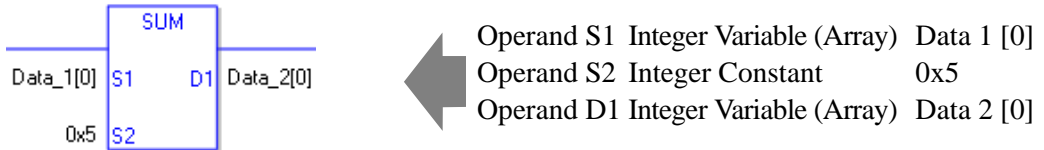
Refer to the following for specifying a constant.

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) If you use any numeric value that cannot be expressed in operands S1 and S2 (infinite or nonnumeric value), the instruction will not be executed.
 - As error indication, error code “6706” is set in #L_CalcErrCode.
 - The output result D1 maintains the value from the last successfully executed instruction.

◆ **System Variables Indicating Execution Results**

If the execution results in an error, #L_CalcErrCode stores the error code.

(Notes)

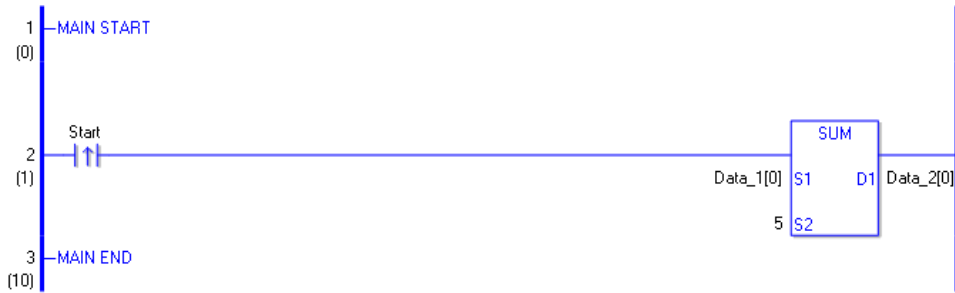
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

SUM

Totals 1 through 5 in Data 1 and saves the total in Data 2.

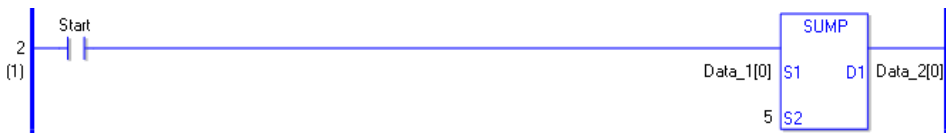


- (1) When the positive transition instruction turns ON, the SUM instruction will be executed. When the SUM instruction is executed, array elements 0 through 4 of Data 1 are totaled, and the result at D1 is stored in Data 2. When using a normally-open instruction, as long as the instruction variable is ON, the SUM 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]	+		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 transition)		Operation	6 to 10
AVEP (Average - positive transition)		Operation	6 to 10

◆ Operand Settings

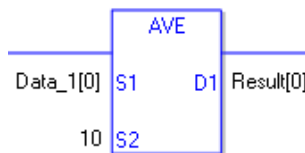
The following table lists the specifiable contents of operands S1, S2 and D2 for the AVE/AVEP instructions.

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 the Instruction

e.g. 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.)



$$\{Data\ 1\ [0] = 2\ Steps\} + \{10 = 1\ Step\} + \{Result\ [0] = 2\ Steps\} + \{1\ Step\} = 6\ Steps$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operand S1 in the AVE/AVEP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		2	○
		Specify integer variable [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	—		—	×
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand S2 in the AVE/AVEP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	1	○
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	—	×
		D_****.B/W [address]	—	×
	F_	—	—	×
	R_	—	—	×
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY only	2	○
	J_	.HR/.MIN/.SEC only	2	○
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Constant	—	1 to 4096	1	○

◆ **Operand Settings**

The following table lists the configurable conditions for Operand D1 in the AVE/AVEP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (Output included)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]	2	○	
		Specify integer variable [variable]	3	○	
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×	
	Float	—		—	×
		Specify float variable [constant]	2	○	
		Specify float variable [variable]	3	○	
	Real	—		—	×
		Specify real variable [constant]	2	○	
		Specify real variable [variable]	3	○	
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

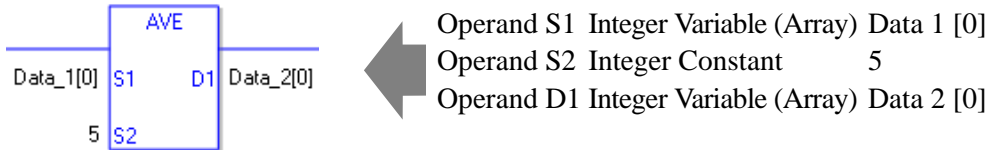
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	—	×
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	—	×
		D_****.B/W [address]	—	×
	F_	—	1	○
	R_	—	1	○
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY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

◆ **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.

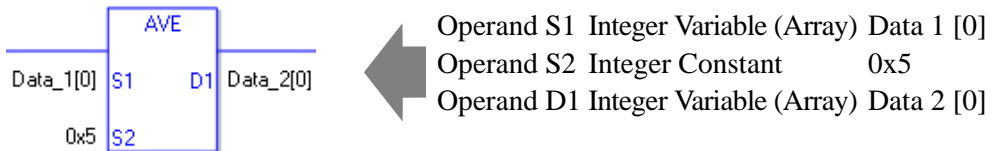
Refer to the following for specifying a constant.

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) If you use any numeric value that cannot be expressed in operands S1 and S2 (infinite or nonnumeric value), the instruction will not be executed.
 As error indication, error code “6706” is set in #L_CalcErrCode.
 The output result D1 maintains the value from the last successfully executed instruction.

◆ **System Variables Indicating Execution Results**

If the execution results in an error, #L_CalcErrCode stores the error code.

(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

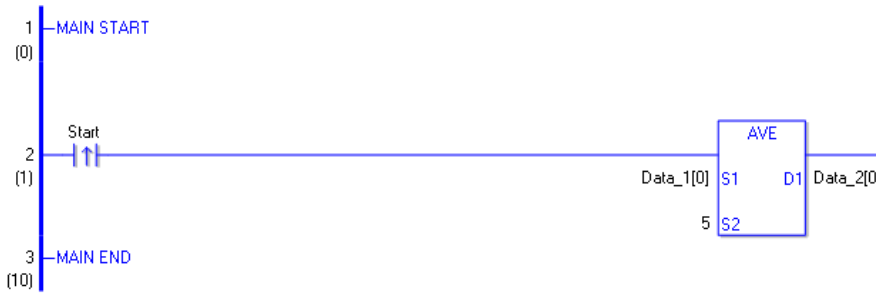
Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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.

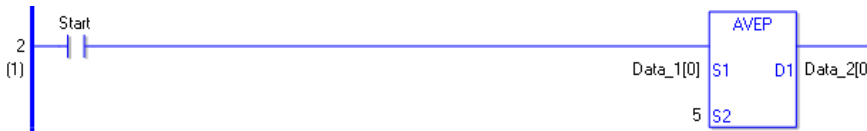


- (1) When the positive transition instruction turns ON, the AVE instruction will be executed. When the AVE instruction is executed, the average of array elements 0 through 4 of Data 1 are calculated and the result at D1 is stored in Data 2. When using a normally-open instruction, as long as the instruction 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]	+ ÷ 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 transition)		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 table lists the specifiable contents of operands S1 and D1 for 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.

e.g. 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 table lists the configurable conditions for Operands (S1 and D1) in the SQRT/SQ RTP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant (Cannot use for D1.)	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	○	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of the SQRT and SQRTP Instructions**

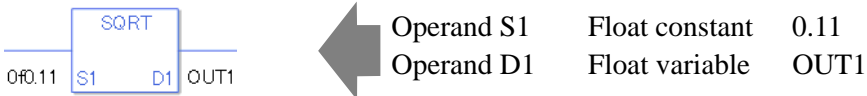
The SQRT/SQRTP instructions both calculate the square root. When the SQRT instruction is executed, the square root of S1 is calculated and the value is saved in D1.

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.

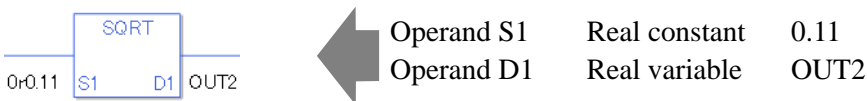
When operand D1 is a float variable

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



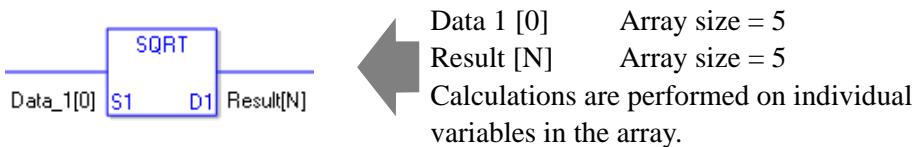
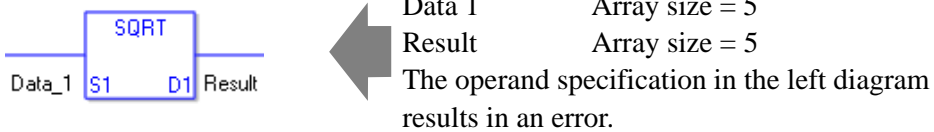
When operand D1 is a real variable

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) If you use any numeric value that cannot be expressed in operands S1 and S2 (infinite or nonnumeric value), the instruction will not be executed.

As error indication, error code “6706” is set in #L_CalcErrCode.

The output result D1 keeps the previous value of the successfully executed instruction.

◆ System Variables Indicating Execution Results

When the result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

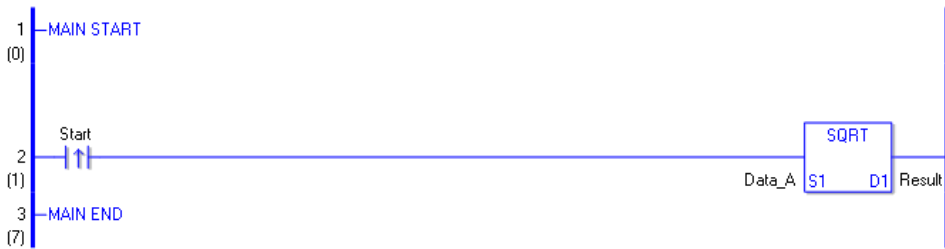
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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 transition)		Operation	3 to 9
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BCNTP (Bit Count - positive transition)		Operation	3 to 9

◆ Operand Settings

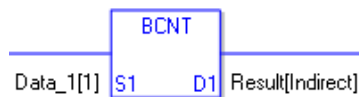
The following table lists the specifiable contents of operands S1 and D1 for the BCNT/BCNTP instructions.

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

e.g. 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 Step} + {Result [indirectly specify] = 3 Step} + {1 Step} = 6 Step

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: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○	
Symbol	Bit	—	—	×	
	Word	—	1	○	
Variable Format *(Notes 1) S1 = IO Enabled D1 = IO Disabled	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer *(Notes 1)	Arrays and modifiers are not specified		1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]		2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		4	○
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		2	○
	Counter	.PV/ .CV only		2	○
	Date	.YR/ .MO/ .DAY only		2	○
Time	.HR/ .MIN/ .SEC only		2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Address Format *(Notes 2) D1 = Disabled	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_ *(Notes 2)	—	1	○
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	2	○
		D_****.B/W [address]	3	○
	F_	—	—	×
	R_	—	—	×
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY only	2	○
	J_	.HR/.MIN/.SEC only	2	○
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Constant *(Notes 3) D1 = Disabled	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	—	×
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	—	×
	Integer *(Notes 3)	-2147483648 to 2147483647	1	○

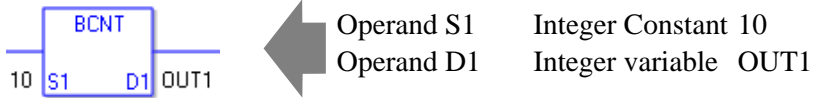
◆ **Explanation of the BCNT and BCNTP Instructions**

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.

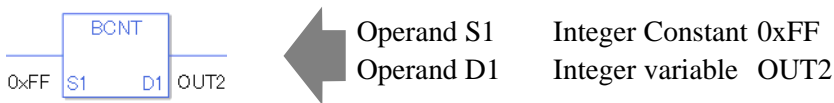
Refer to the following for specifying a constant.

When operand D1 is an integer variable



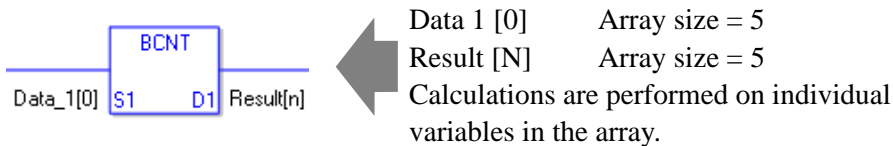
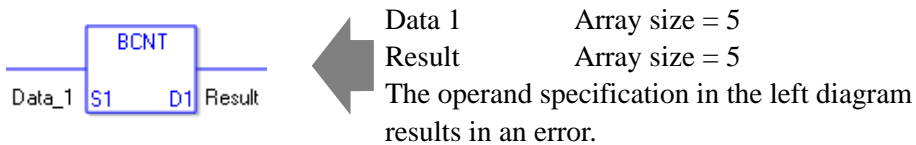
When operand D1 is an integer variable and you want to input hexadecimal values in operand S2.

When 0x (zero and lower case x) is input, the following values are 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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

BCNT

Counts the number of bits that are ON, and saves the number in an integer variable.



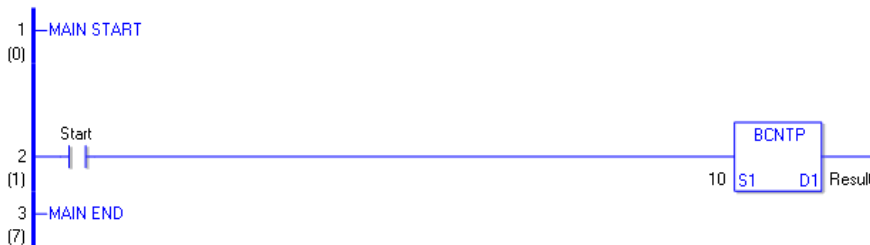
(1) When the positive transition instruction turns ON, the BCNT instruction will be executed.

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 transition)		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 below table.

PID Variable

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

Other operands are as follows.

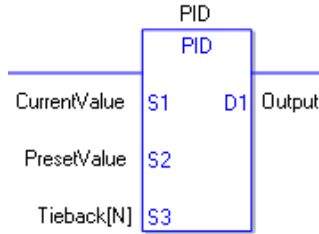
- S1: Setpoint
- S2: Present Value
- S3: Tieback Value (The setting value is output when an instruction is disabled)
- D1: Output Value

◆ **Operand Settings**

The following table lists the specifiable contents 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 of Steps in operand HP + 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

e.g. 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 (the PID variables for the HP operand are fixed in 1 Step)} + {Current Value = 1 Step} + {Setting Value = 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 table lists the configurable conditions for Operands (S1, S2, S3, and D1) in the PID instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) S1, S2, S3 = IO Enabled D1 = IO Disabled	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
Date	.YR/ .MO/ .DAY only	2	○	
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Address Format *(Notes 2) D1 = Disabled	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_ *(Notes 2)	—	1	○
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	2	○
		D_****.B/W [address]	3	○
	F_	—	—	×
	R_	—	—	×
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY only	2	○
	J_	.HR/.MIN/.SEC only	2	○
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	
Constant *(Notes 3) D1, S2 = Disabled	Float	±1.175494351e-38 to ±3.402823466e+38	—	×
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	×
	Integer *(Notes 3)	-2147483648 to 2147483647	1	○

◆ **Basic Function**

The PID instruction compares the measured values (current values) and the setting 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 below 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

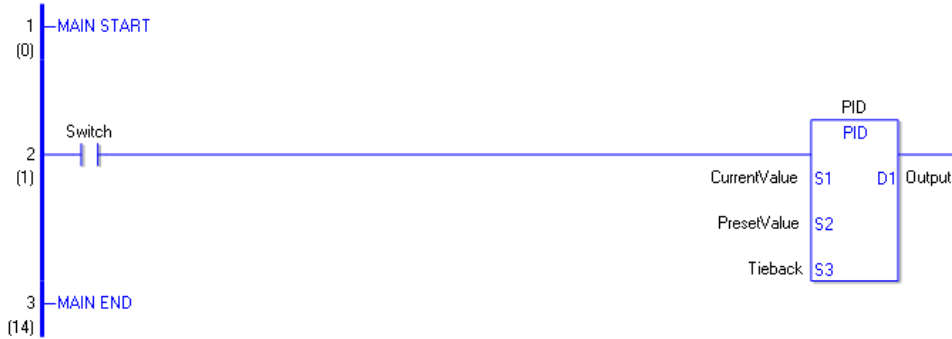
Using the [Tuning] tab that will be described later, adjust the sampling time to reduce the effect of noise on deviation. The below formula shows the result of filtering on the deviation.

$$EF_n = EF_{n-1} + \frac{T_{Loop}}{T_{Filter}} (E_n - EF_{n-1})$$

- 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 to the integer variable operands S1, S2, S3 and D1.

PID Variable

After a variable has been allocated to the PID instruction, the relevant members are automatically allocated to the variables.

PID Variable

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

- Values substituted into the proportional constant, integral calculus time, and differential calculus time appear differently between when they are entered from the “PID monitor window” and when entered to each PID variable member on the program. To enter the value on the program, multiply the values of the proportional constant, integral calculus time, and differential calculus time by 1000.
Example: Proportional constant $0.1 \times 1000 \rightarrow 100$

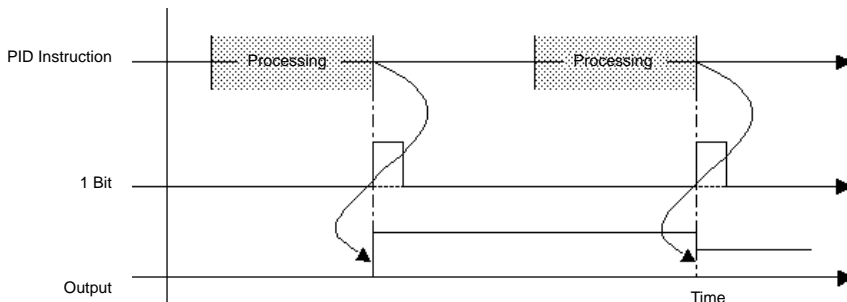
(Notes)

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

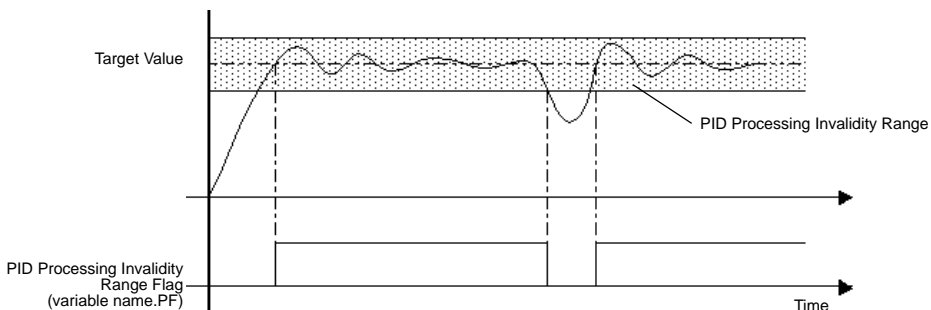
PID Instruction Processing Completion Flag (variable name.Q)

.Q turns ON when the calculation processing is completes and the calculated value is output to operand D1. The PID Instruction Processing Completion Flag remains ON during the scan.



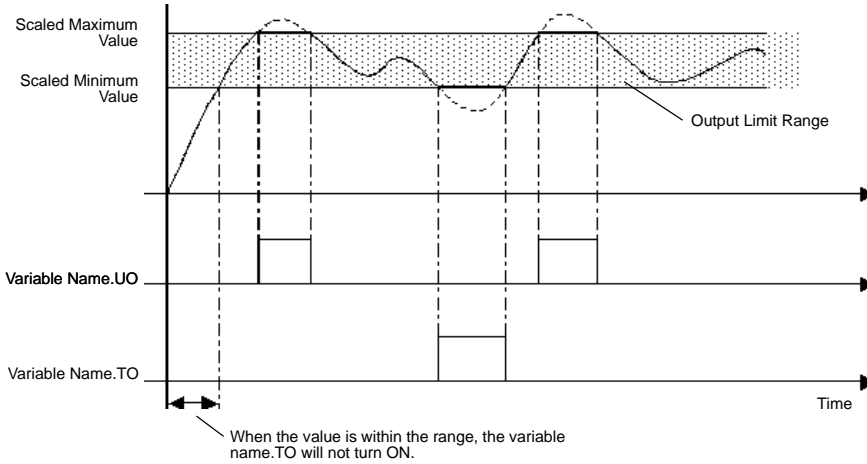
Processing Invalidity Range Flag (variable name.PF)

The flag turns ON when the current value reaches the target value in the range specified in the PID variable members (processing invalidity range, variable name.PF). When the current value exceeds the range, the flag turns OFF.



Output Values over the Upper/Lower Limits (variable name.UO, variable name.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 variable name.UO turns ON. When the result is below the specified lower limit, the variable name.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.



Exceeding Processing of Integral Cycles (variable name.IF)

Double-clicking the PID instruction displays a dialog box for specifying the execution range of the PID instruction. When the value exceeds the range for the integral settings, the variable name.IF turns ON. The PID continues even when the status bits turn ON and the calculated value is output as the specified upper limit. The range for the integral settings in each status is a setting item that executes integration only within the specified range.

Proportional Constant (variable name.KP)

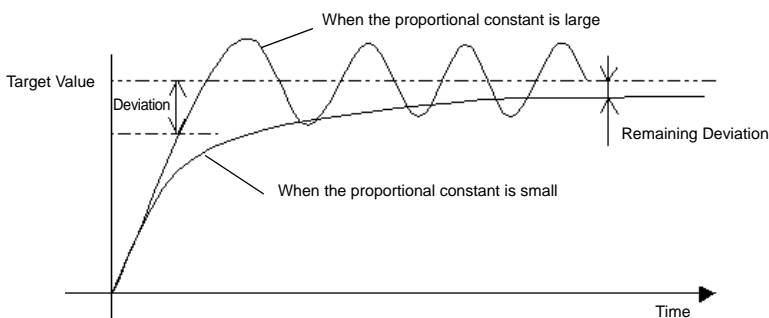
Specify a proportional constant (variable name.KP) to output a value corresponding to the deviation between the target and current values.

A smaller proportional constant produces a smaller output value to reach the target value, and eliminates overshoot but may increase the remaining 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.

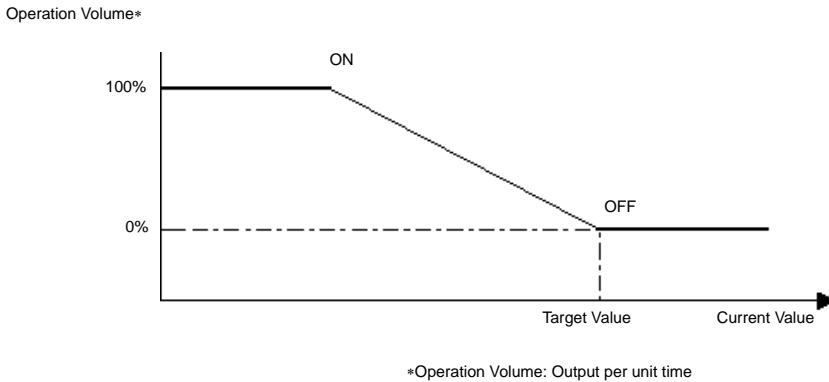
Settings range from 0.01 to 1000.00 Internal data are integer variables. Decimals cannot be used.

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).



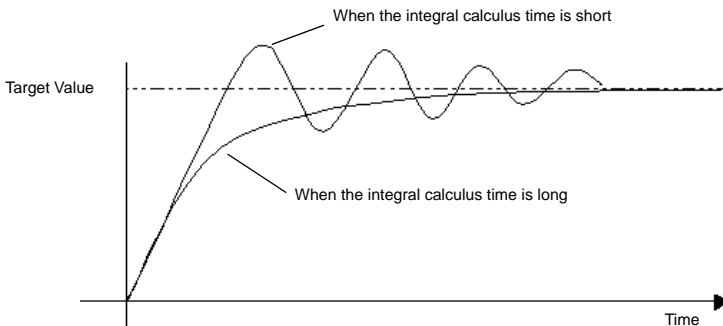
Integral Calculus Time (variable name.TR)

Specifying the integral calculus time (variable name.TR) can eliminate deviation in reaching the target value.

If you only use proportional control, the operation volume will decrease too much as it reaches the target value and will not be large enough to even out the deviation (control output). This small gap is called the “remaining deviation” and can be eliminated by integral control. Integral control is a control method that accumulates deviations over time and adjusts (increases) the operation volume to eliminate deviations once the volume reaches a certain size. A shorter integral calculus time requires a larger operation volume and shorter time to reach the target, but results in overshoot and hunting. A longer integral calculus time requires a smaller operation volume to reach the target and reduces overshoot and hunting, but requires more time to reach the target.

The integral calculus time specifies an interval time (in seconds) for executing integral processing.

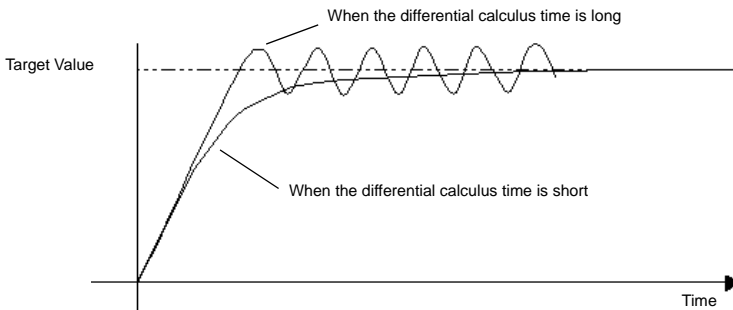
Settings range from 0.100 to 3000.000 Internal data are integer variables. Decimals cannot be used. To set 0.1, specify $0.1 \times 1000 = 100$. Specify variable.TR as the value multiplied by 1000.



Differential Calculus Time (variable name.TD)

By specifying the differential calculus time (variable name.TD), it will immediately respond to the change.

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.

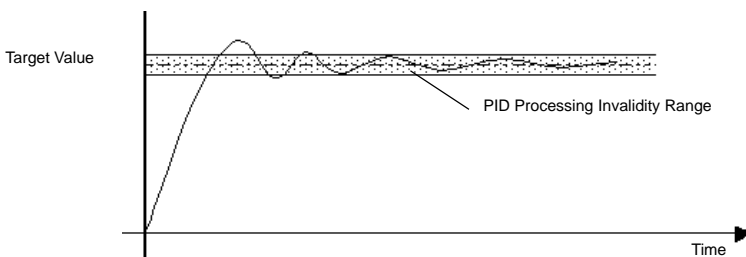


Settings range from 0.00 to 3000.00

Internal data are integer variables. Decimals cannot be used.
 To set 0.1, specify $0.1 \times 1000 = 100$.
 Specify variable.TD as the value multiplied by 1000.

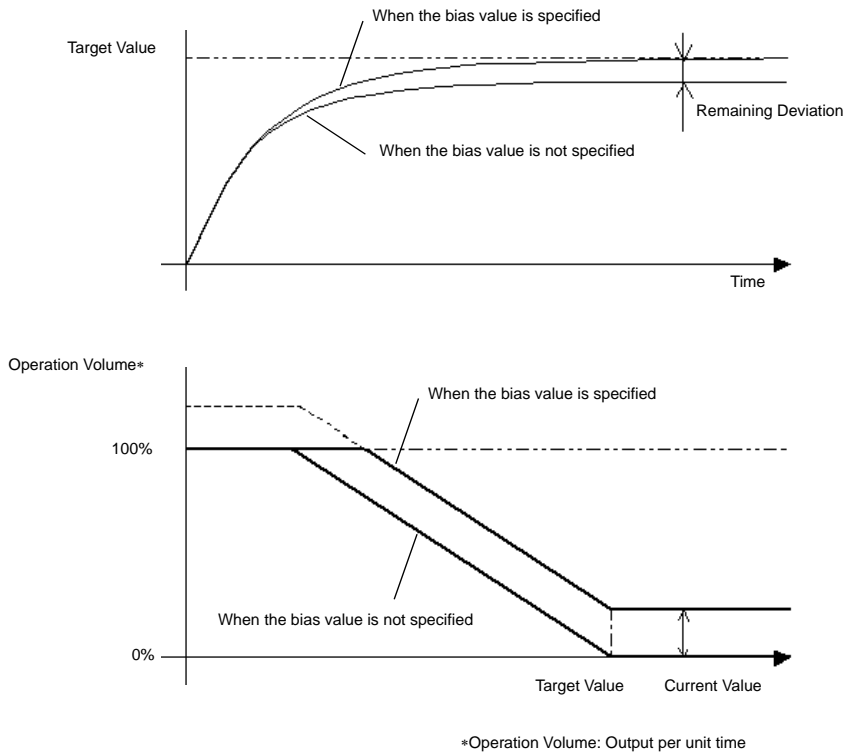
Processing Invalidity Range (variable name.PA)

In the “processing invalidity range,” PID control does not occur and the minimum value is output for smooth control without hunting.



Bias (variable name.BA)

Sets the bias value (offset). This reduces any remaining deviation incurred in the proportional control.

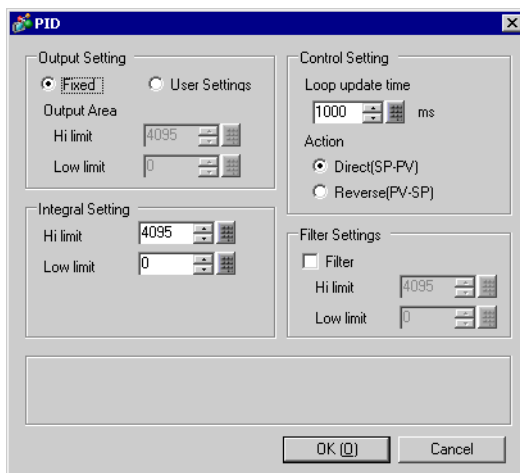


Sampling Frequency (variable name.ST)

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.

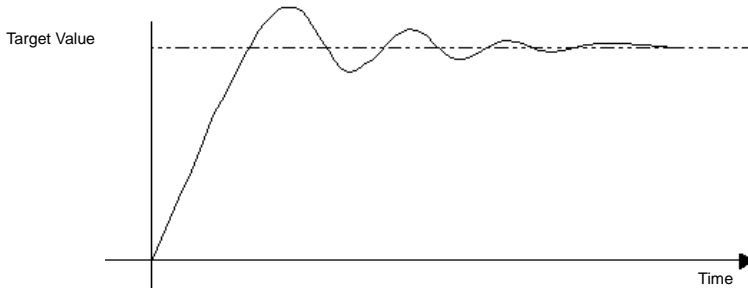
You can use the filtering feature to specify the frequency. The sampling frequency must be larger than the frequency of obtaining data.

Settings range from 10 to 65535 ms

Action:

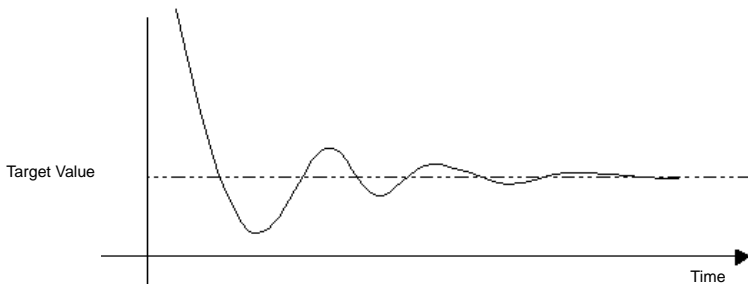
Direct (D1–D2)

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. (cooling, and so on)



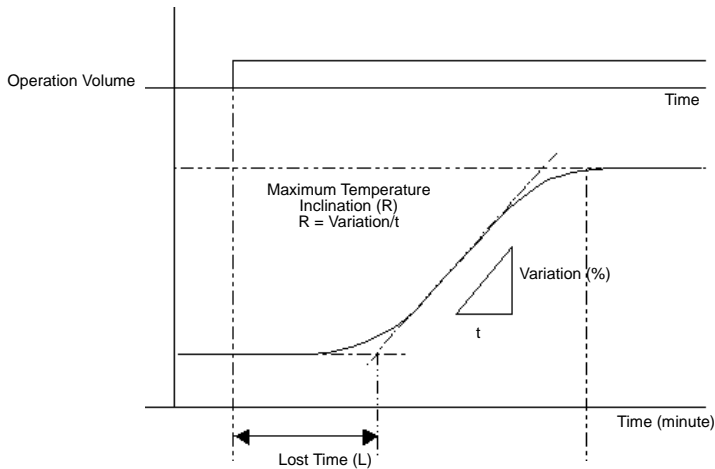
Filter Settings

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 (variable name.UO, variable name.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 below 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]

30.5.14 Function Instruction

■ SIN and SINP (Sine)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
SIN (Sine - level transition)		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 table lists the specifiable contents 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

e.g. 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 table lists the configurable conditions for Operands (S1) and (D1).

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant (Cannot use for D1.)	Float	±1.175494351e-38 to ±3.402823466e+38	1	○	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of the SIN and SINP Instructions**

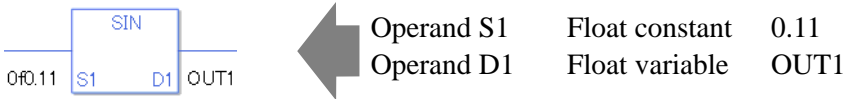
The SIN and SINP instructions are sine instructions for trigonometric functions. The SIN instruction calculates the sine of S1 and stores the result in D1.

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.

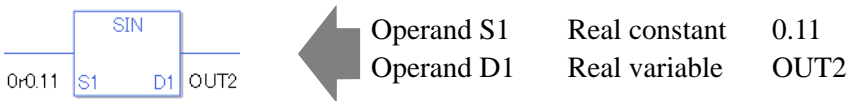
When operand D1 is a float variable

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



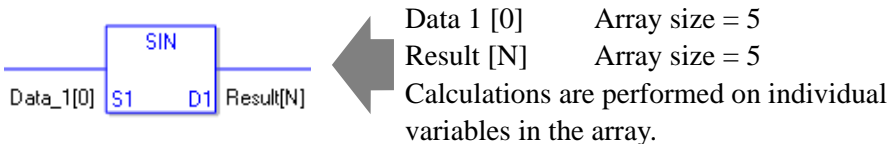
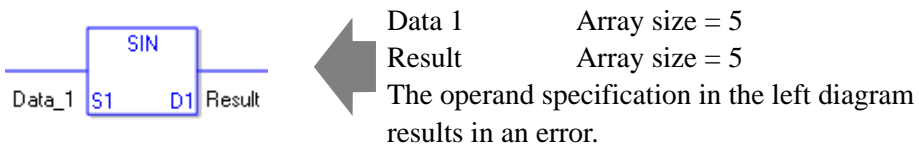
When operand D1 is a real variable

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 variables are the same type.



System Variables Indicating Execution Results

When the result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

SIN

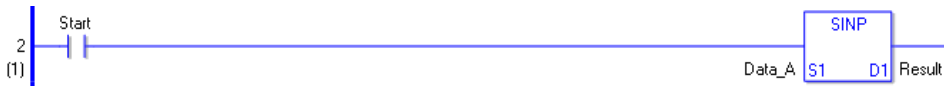


(1) The SIN instruction is executed when the positive transition instruction turns ON. The SIN instruction calculates the sine of DataA and stores the result in D1.

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, only the upward transition is detected, and the SINP instruction is executed. 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 transition)		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 table lists the specifiable contents 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

e.g. 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 table lists the specifiable contents of operands S1 and D1 for the COS and COSP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant (Cannot use for D1.)	Float	±1.175494351e-38 to ±3.402823466e+38	1	○	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of the COS and COSP Instructions**

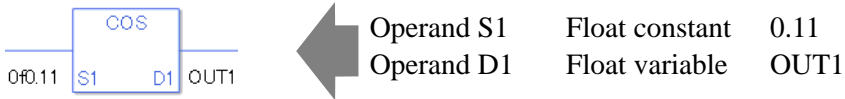
The COS and COSP instructions are cosine instructions for trigonometric functions. The COS instruction calculates the cosine of S1 and stores the result in D1. Enter the number of radians in S1 to get the result in D1 as a real value between -1.0 and 1.0.

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.

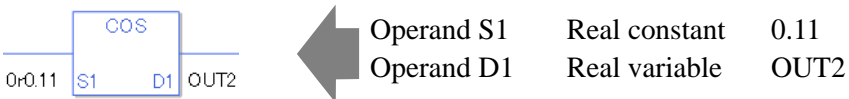
When operand D1 is a float variable

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



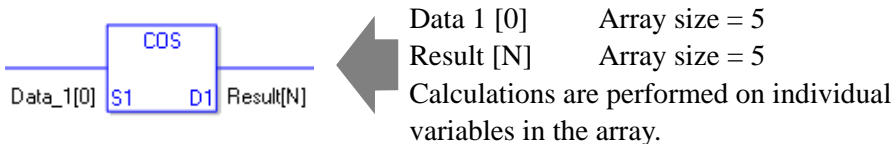
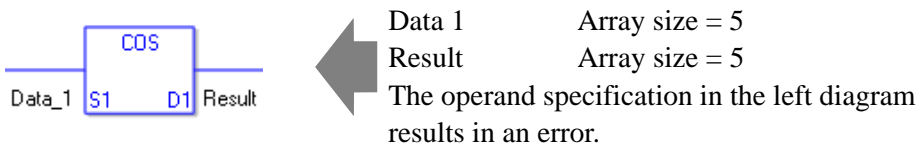
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

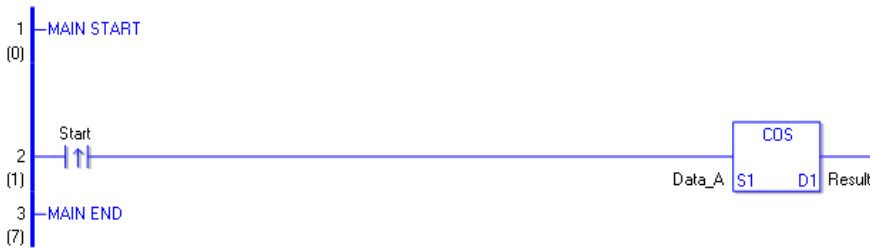
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

COS

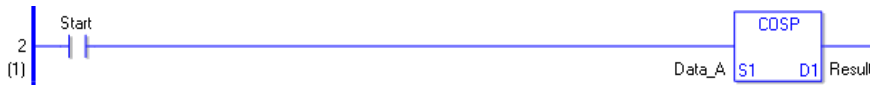


(1) The COS instruction is executed when the positive transition instruction turns ON. The COS instruction calculates the cosine of DataA and stores the result in D1.

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, only the upward transition is detected, and the COSP instruction is executed. 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 transition)		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 table lists the specifiable contents 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 D + 1 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{ \text{Data 1 [0]} = 2 \text{ Steps} \} + \{ \text{Result [N]} = 3 \text{ Steps} \} + \{ 1 \text{ Step} \} = 6 \text{ Steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the specifiable contents of operands S1 and D1 for the TAN and TANP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

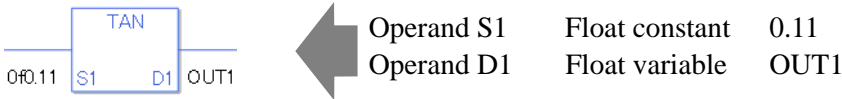
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV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

◆ **Explanation of the TAN and TANP Instructions**

The TAN and TANP instructions are tangent instructions for trigonometric functions. The TAN instruction calculates the tangent of S1 and stores the result in D1. Enter the number of radians in S1. The result in D1 stays within the range of real variables or float variables. 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.

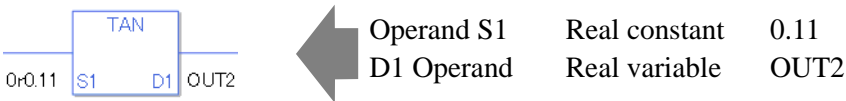
When operand D1 is a float variable

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



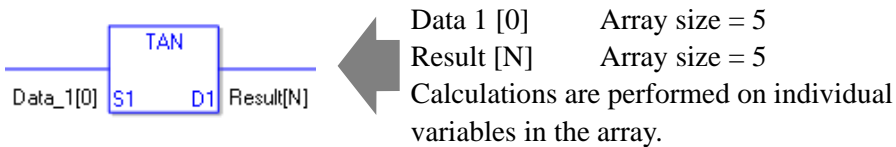
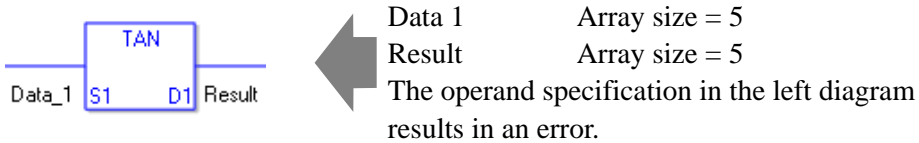
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

TAN



(1) The TAN instruction is executed when the positive transition instruction turns ON. The TAN instruction calculates the tangent of DataA 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, only the upward transition is detected, and the TANP instruction is executed. 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 transition)		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 table lists the specifiable contents 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

e.g. 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.)



{Data 1 [0] = 2 Steps} + {Result [N] = 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 lists the specifiable contents of operands S1 and D1 for the ASIN and ASINP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant (Cannot use for D1.)	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	○	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of the ASIN and ASINP Instructions**

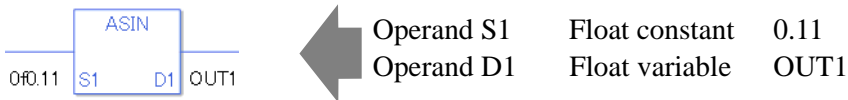
The ASIN and ASINP instructions are arc sine instructions for trigonometric functions. The ASIN instruction calculates the arc sine of S1 and stores the result in D1. $\text{Sin}^{-1}(S1)$ is stored in D1. Enter a value between -1.0 and 1.0 for S1. The result in D1 is in radians as a real value between $-\pi/2$ and $1\pi/2$. π is approximately 3.1415926535897 .

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.

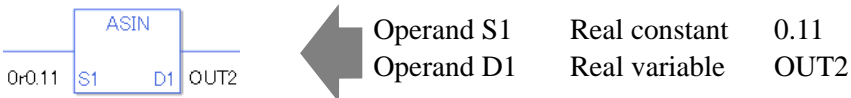
When operand D1 is a float variable

When Of (zero and lower case “f”) is input, the following values become float values.



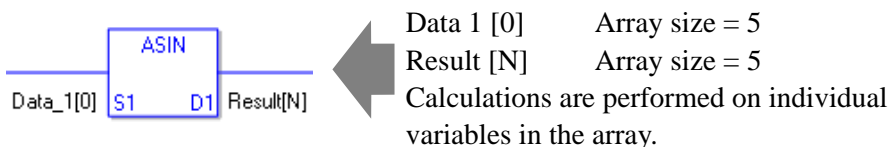
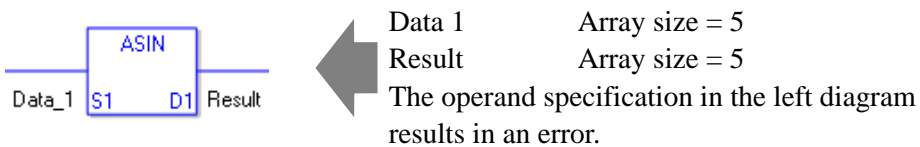
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

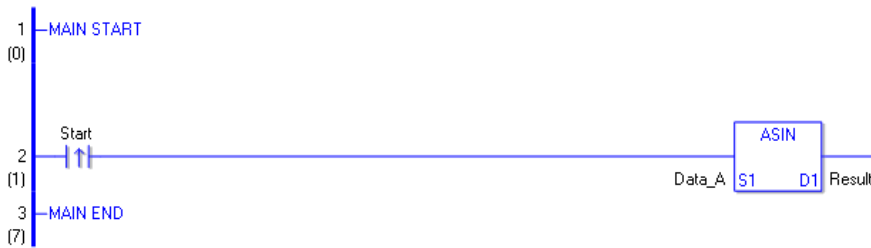
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

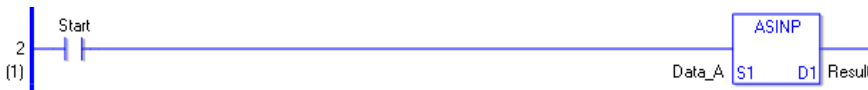
ASIN



- (1) The ASIN instruction is executed when the positive transition instruction turns ON. The ASIN instruction calculates the arc sine of DataA 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 transition)		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 table lists the specifiable contents 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

e.g. 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 table lists the specifiable contents of operands S1 and D1 for the ACOS and ACOSP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
	Time	.HR/.MIN/.SEC only		—	×
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant (Cannot use for D1.)	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	○	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of the ACOS and ACOSP Instructions**

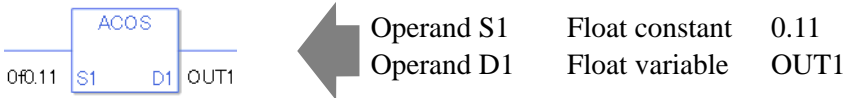
The ACOS and ACOSP instructions are arc cosine instructions for trigonometric functions. The ACOS instruction calculates the arc cosine of S1 and stores the result in D1. $\text{COS}^{-1}(S1)$ is stored in D1. Enter a value between -1.0 and 1.0 for S1. The result in D1 in radians is a real value between 0 and π . π is approximately 3.1415926535897 (real).

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.

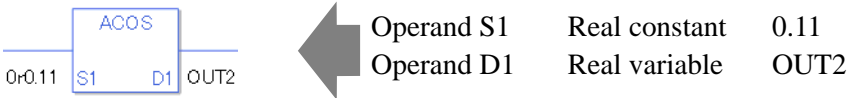
When operand D1 is a float variable

When Of (zero and lower case “f”) is input, the following values become float values.



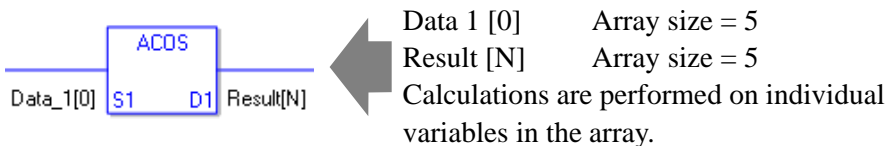
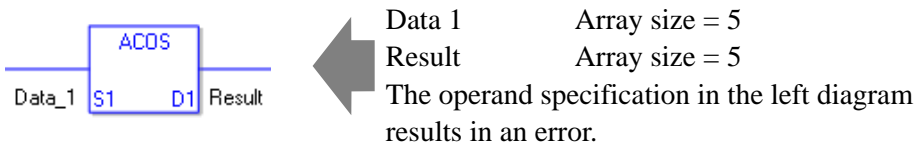
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

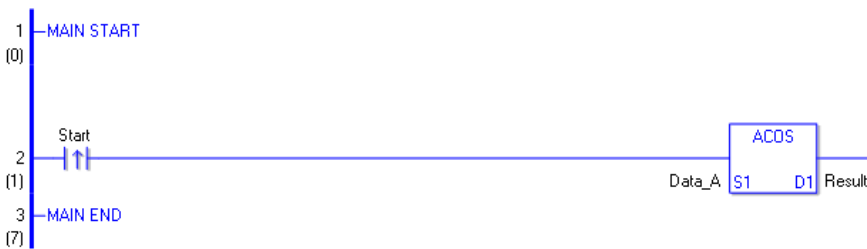
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

ACOS



(1) The ACOS instruction is executed when the positive transition instruction turns ON. The ACOS instruction calculates the arc cosine of DataA 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, only the upward transition is detected, and the ACOSP instruction is executed. 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 transition)		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 table lists the specifiable contents 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

e.g. 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.)



{Data 1 [0] = 2 Steps} + {Result [N] = 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 lists the specifiable contents of operands S1 and D1 for the ATAN and ATANP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

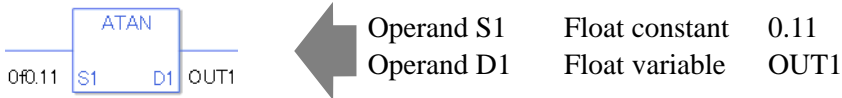
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant (Cannot use for D1.)	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	○	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of the ATAN and ATANP Instructions**

The ATAN and ATANP instructions are arc tangent instructions for trigonometric functions. The ATAN instruction calculates the arc tangent of S1 and stores the result in D1. $TAN^{-1}(S1)$ is stored in D1. Enter a value between -1.0 and 1.0 for S1. The result in D1 in radians is a real value between $-\pi/2$ and $\pi/2$. π is approximately 3.1415926535897 (real). 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.

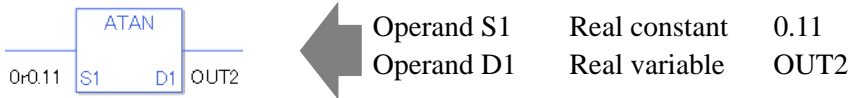
When operand D1 is a float variable

When Of (zero and lower case “f”) is input, the following values become float values.



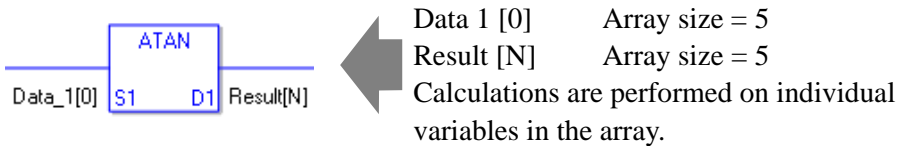
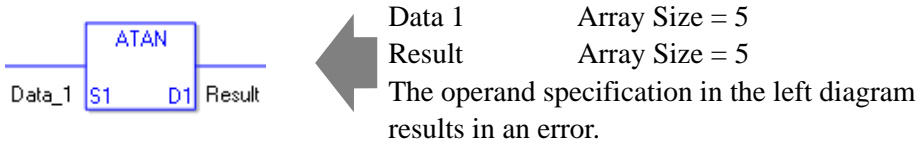
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

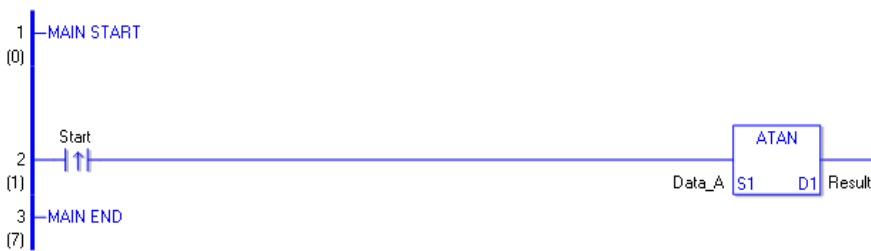
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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 DataA 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, only the upward transition is detected, and the ATANP instruction is executed. 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 transition)		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

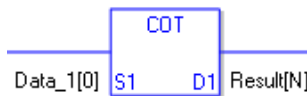
The following table lists the specifiable contents of operands S1 and D1 for the COT and COTP instructions.

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

e.g. 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.)



{Data 1 [0] = 2 Steps} + {Result [N] = 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 lists the specifiable contents of operands S1 and D1 for the COT and COTP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant (Cannot use for D1.)	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	○	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of the COT and COTP Instructions**

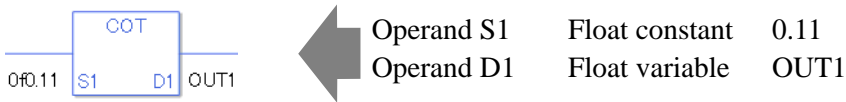
The COT and COTP instructions are cotangent instructions for trigonometric functions. The COT instruction calculates the cotangent of S1 and stores the result in D1. Enter the number of radians in S1. The result in D1 increases as S1 approaches a multiple of π in absolute value. The range that can be expressed, however, is a real value between approximately $\pm 2.225e-308$ and $\pm 1.79e+308$ (real).

π is approximately 3.1415926535897 (real). The COT and COTP instructions are always conducted. When using the COT and COTP 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.

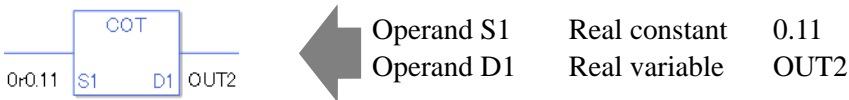
When operand D1 is a float variable

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



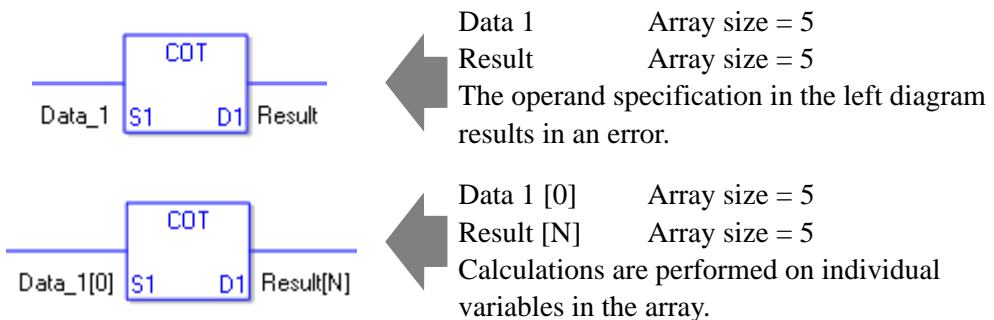
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

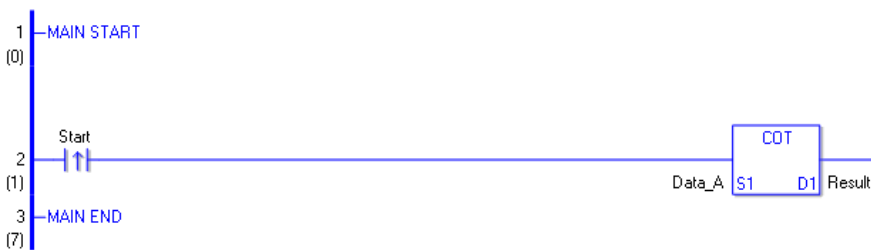
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

COT



(1) The COT instruction is executed when the positive transition instruction turns ON. The COT instruction calculates the cotangent of DataA 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, only the upward transition is detected, and the COTP instruction is executed. 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 transition)		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 table lists the specifiable contents 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

e.g. 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.)



$$\{ \text{Data 1 [0]} = 2 \text{ Steps} \} + \{ \text{Result [N]} = 3 \text{ Steps} \} + \{ 1 \text{ Step} \} = 6 \text{ Steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the specifiable contents of operands S1 and D1 for the EXP and EXPP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant (Cannot use for D1.)	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	○	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **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.

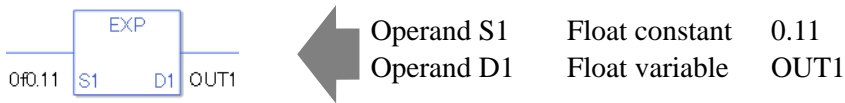
Equation: $D1 = e^{S1}$ e is approximately 2.7182818284590 (real).

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.

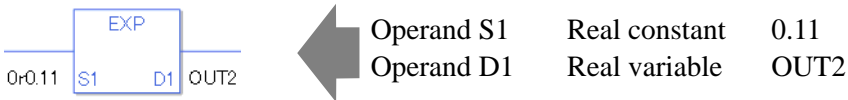
When operand D1 is a float variable

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



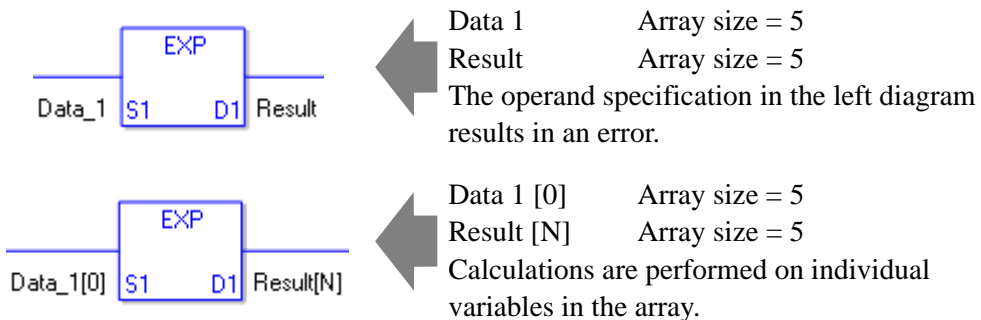
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

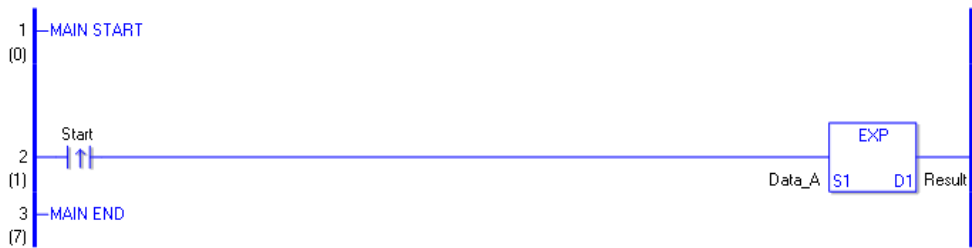
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

EXP



(1) The EXP instruction is executed when the positive transition instruction turns ON. The EXP instruction calculates the exponent of DataA 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, only the upward transition is detected, and the EXPP instruction is executed. 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 transition)		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 table lists the specifiable contents 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

e.g. 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 table lists the specifiable contents of operands S1 and D1 for the LN and LNP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
	Time	.HR/.MIN/.SEC only		—	×
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant (Cannot use for D1.)	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	○	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of the LN and LNP Instructions**

The LN and LNP instructions are exponential 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.

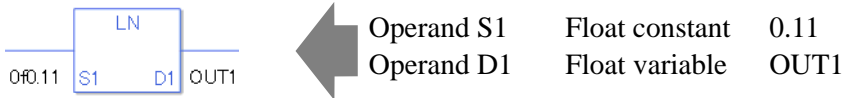
Equation: $D1 = \log_e S1$ e is approximately 2.7182818284590 (real).

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.

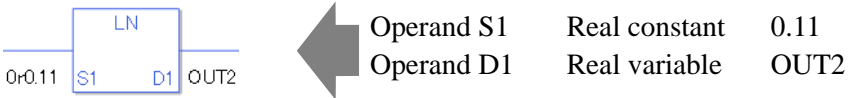
When operand D1 is a float variable

When Of (zero and lower case “f”) is input, the following values become float values.



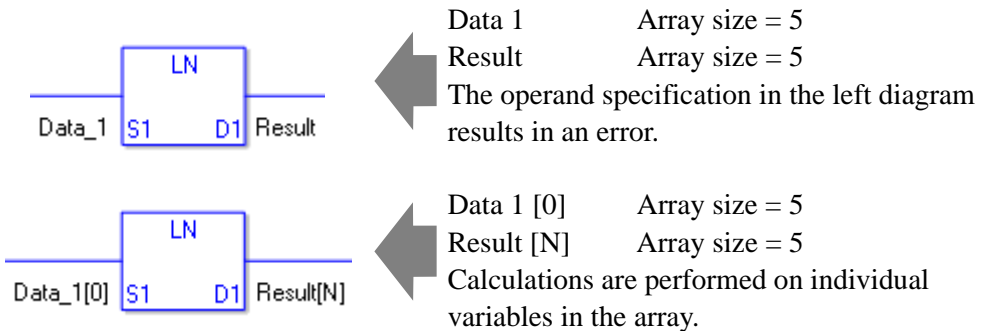
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

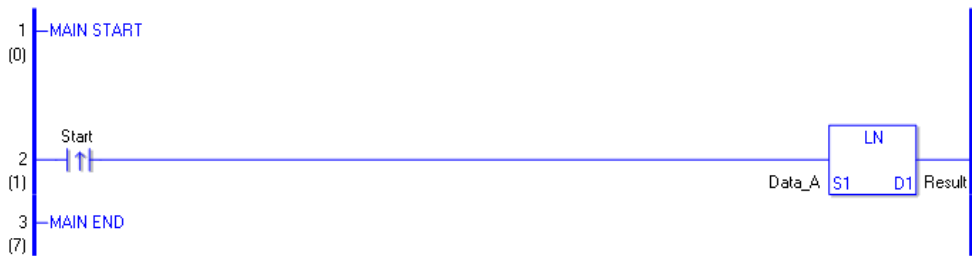
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

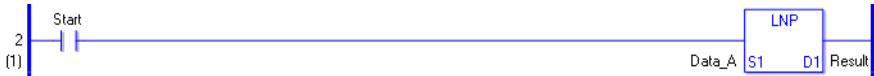
LN



- (1) The LN instruction is executed when the positive transition instruction turns ON. The LN instruction calculates the natural logarithmic function of DataA 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, only the upward transition is detected, and the LNP instruction is executed. 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 transition)		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 table lists the specifiable contents 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

e.g. 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.)



{Data 1 [0] = 2 Steps} + {Result [N] = 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 lists the specifiable contents of operands S1 and D1 for the LG10 and LG10P instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant (Cannot use for D1.)	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	○	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of LG10 and LG10P Instructions**

The LG10 and LG10P instructions are exponential 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 log₁₀ S1 is output as a real value.

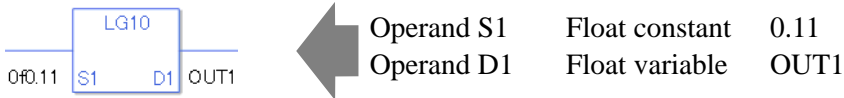
Equation: $D1 = \log_{10} S1$

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.

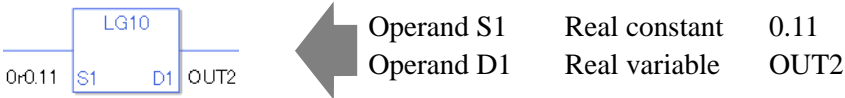
When operand D1 is a float variable

When Of (zero and lower case “f”) is input, the following values become float values.



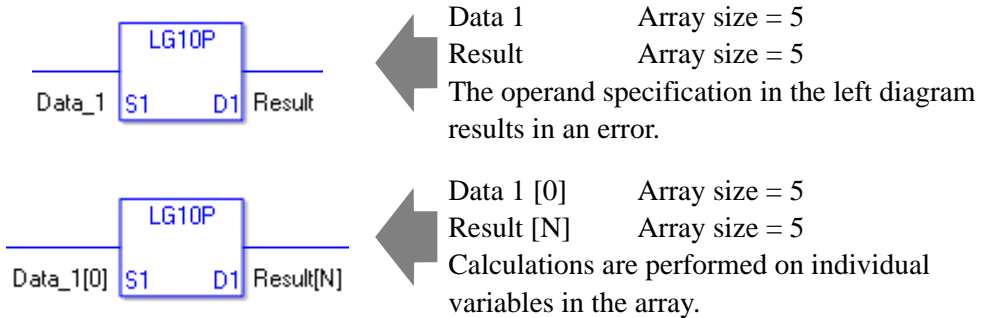
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

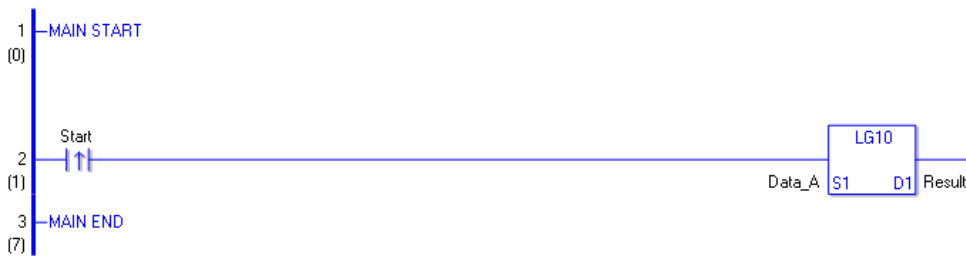
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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 DataA 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

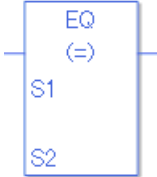


(1) The LG10P and LG10 instructions differ in when they run. In the LG10P instructions, even when using a normally open instruction, only the upward transition is detected, and the LG10P instruction is executed. Therefore, the LG10P instruction is executed only for one scan, even when the normally open instruction bit remains ON.

30.5.15 Compare Instruction (Arithmetic)

■ EQ (=)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
EQ (Equal To - level transition)		Comparison	3 to 9

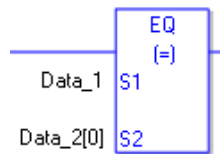
◆ Operand Settings

The following table lists the specifiable contents 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 = Total Number of Steps in One Instruction

e.g. 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 step} + {Data 2 [0] = 2 steps} + {1 step} = 4 steps

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the specifiable contents of operands S1 and S2 for the EQ instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	—	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	—	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
	Time	.HR/.MIN/.SEC only	2	○
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	

Continued

Explanations of Each Instruction

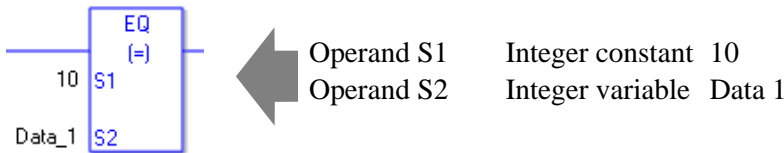
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	1	○
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	2	○
		D_****.B/W [address]	3	○
	F_	—	1	○
	R_	—	1	○
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY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

◆ **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.

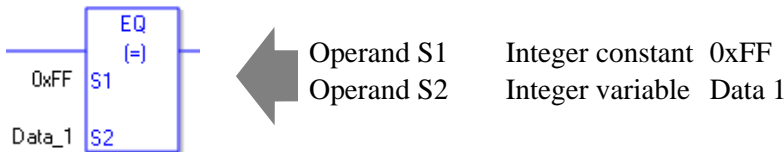
When using the EQ instruction, an error will occur if the variables specified in operands S1 and S2 are not the same type. Specify the same variable type in operands S1 and S2. Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



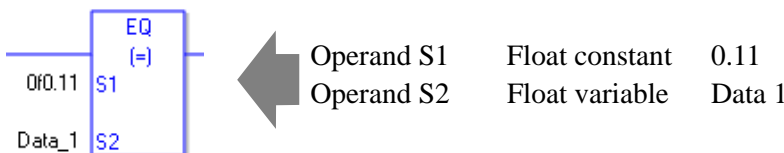
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case “x”) is input, the following values become hexadecimal values.



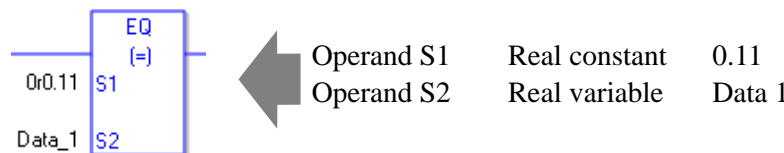
When entering float constants in operands S1 and S2

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



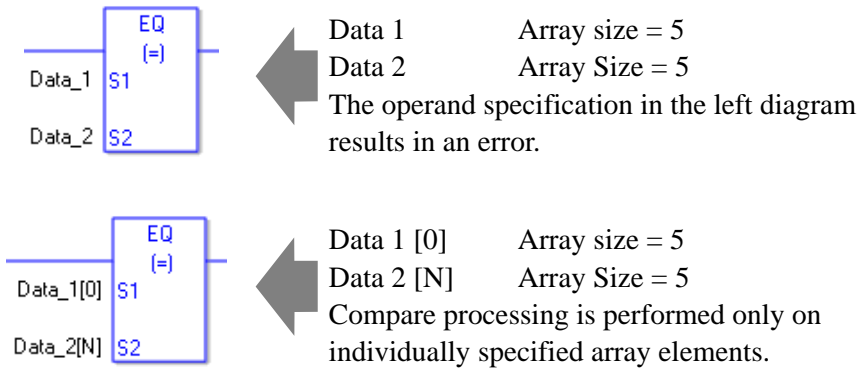
When entering real constants in operands S1 and S2

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).

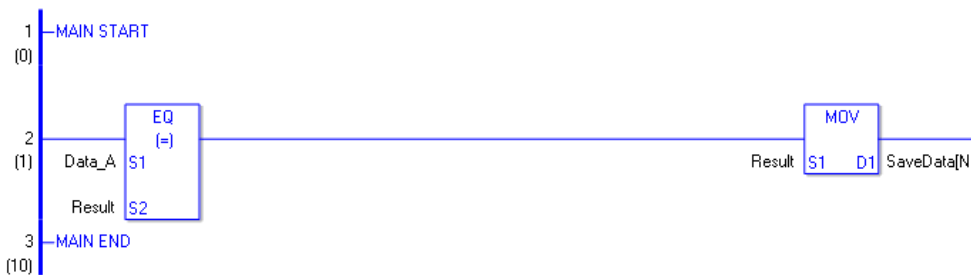
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

EQ

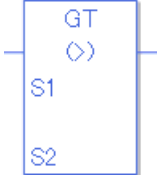
Compares integer variables and outputs the result in D1.



- (1) DataA and the OperationResult 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 transition)		Comparison	3 to 9

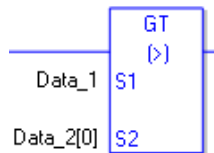
◆ **Operand Settings**

The following table lists the specifiable contents 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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ step\} + \{Data\ 2\ [0] = 2\ steps\} + \{1\ step\} = 4\ steps$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the specifiable contents of operands S1 and S2 for the GT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	1	○	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○	
Symbol	Bit	—	—	×	
	Word	—	1	○	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]		2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		4	○
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		2	○
	Counter	.PV/.CV only		2	○
	Date	.YR/.MO/.DAY only		2	○
	Time	.HR/.MIN/.SEC only		2	○
PID	.KP/.TR/.TD/.PA/.BA/.ST only		2	○	

Continued

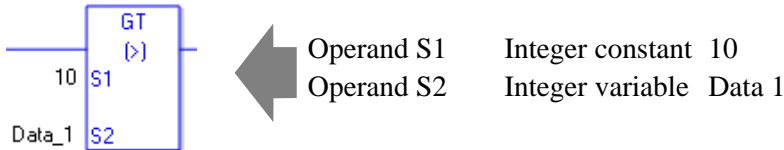
Explanations of Each Instruction

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

◆ **Explanation of the GT Instruction**

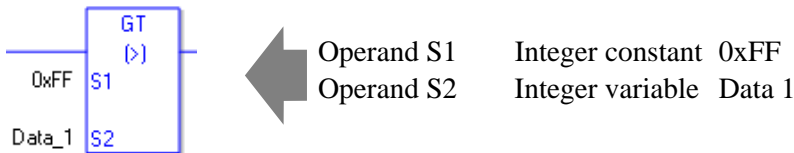
The GT instruction is a compare instruction. The GT instruction compares S1 with S2. If the result of the comparison is $S1 > S2$, the instruction passes power. Be careful when comparing real values. For example, if the operand value is 2.000000000001, it is still greater than 2. When using the GT instruction, 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



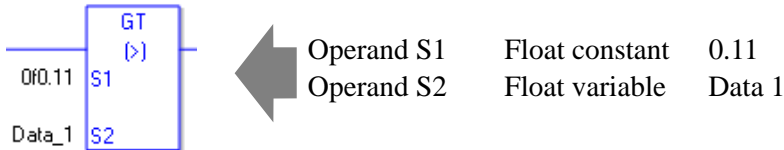
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case “x”) is input, the following values become hexadecimal values.



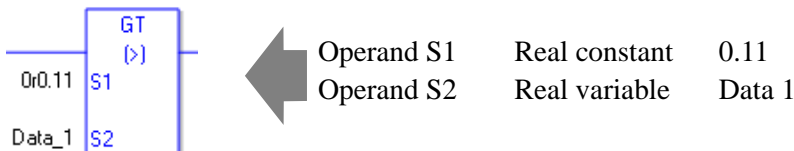
When entering float constants in operands S1 and S2

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



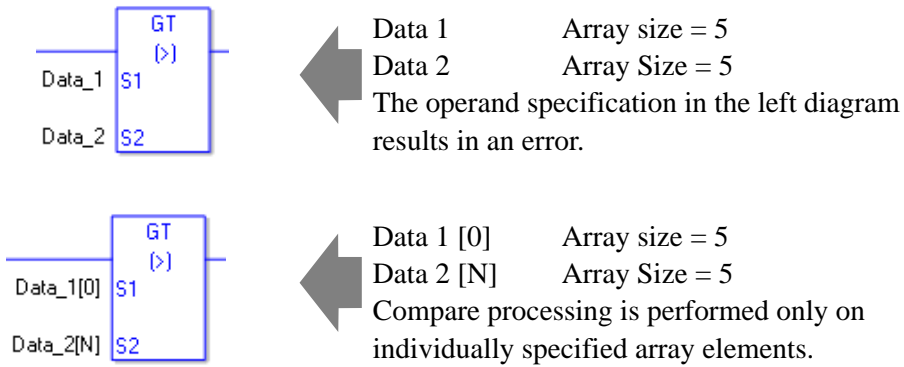
When entering real constants in operands S1 and S2

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).

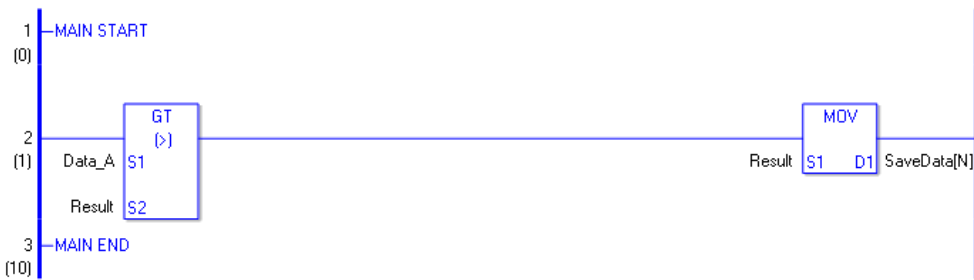
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

GT

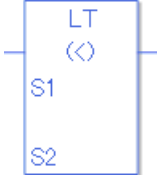
Compares integer variables and outputs the result in D1.



(1) DataA and OperationResult are compared to determine whether DataA is greater than OperationResult. 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 transition)		Comparison	3 to 9

◆ **Operand Settings**

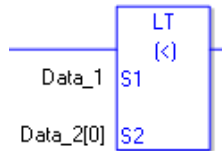
The following table lists the specifiable contents of operands S1 and S2 for the LT instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ step\} + \{Data\ 2\ [0] = 2\ steps\} + \{1\ step\} = 4\ steps$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the specifiable contents of operands S1 and S2 for the LT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	1	○	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○	
Symbol	Bit	—	—	×	
	Word	—	1	○	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]		2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		4	○
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		2	○
	Counter	.PV/.CV only		2	○
	Date	.YR/.MO/.DAY only		2	○
	Time	.HR/.MIN/.SEC only		2	○
PID	.KP/.TR/.TD/.PA/.BA/.ST only		2	○	

Continued

Explanations of Each Instruction

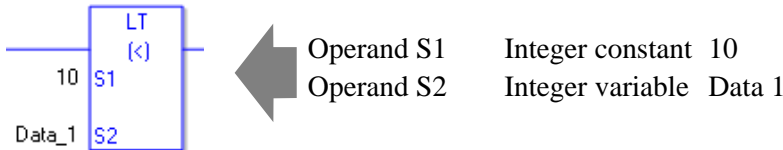
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	1	○
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	2	○
		D_****.B/W [address]	3	○
	F_	—	1	○
	R_	—	1	○
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY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

◆ **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.

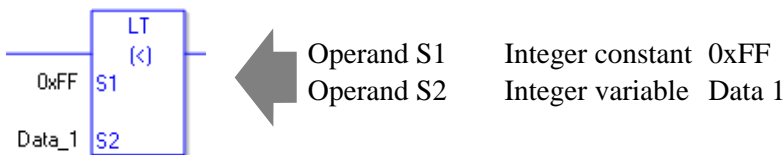
Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



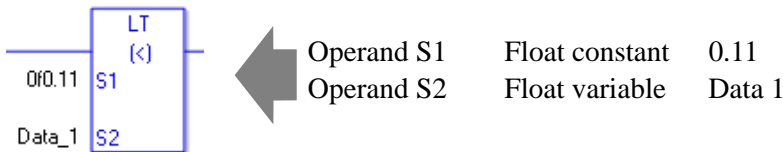
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case “x”) is input, the following values become hexadecimal values.



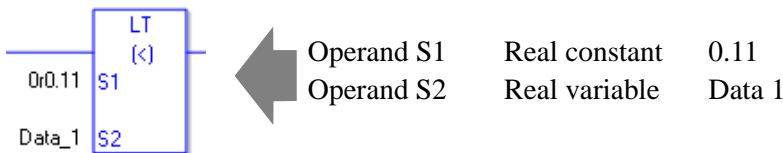
When entering float constants in operands S1 and S2

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



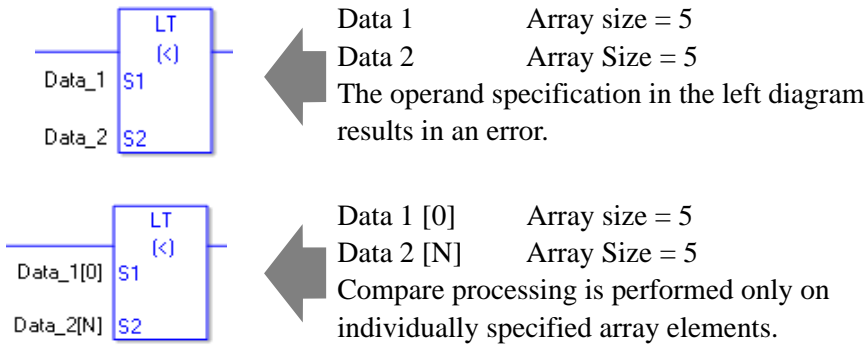
When entering real constants in operands S1 and S2

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).

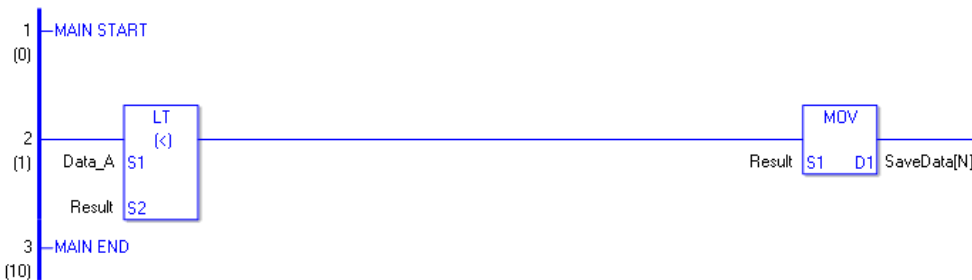
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

LT

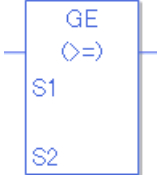
Compares integer variables and outputs the result in D1.



(1) DataA and OperationResult are compared to determine whether DataA is less than OperationResult. 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 transition)		Comparison	3 to 9

◆ **Operand Settings**

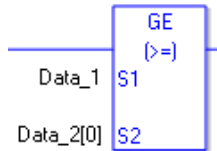
The following table lists the specifiable contents of operands S1 and S2 for the GE instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{ \text{Data 1} = 1 \text{ step} \} + \{ \text{Data 2 [0]} = 2 \text{ steps} \} + \{ 1 \text{ step} \} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the specifiable contents of operands S1 and S2 for the GE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	1	○	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○	
Symbol	Bit	—	—	×	
	Word	—	1	○	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]		2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		4	○
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		2	○
	Counter	.PV/.CV only		2	○
	Date	.YR/ .MO/ .DAY only		2	○
	Time	.HR/ .MIN/ .SEC only		2	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		2	○	

Continued

Explanations of Each Instruction

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

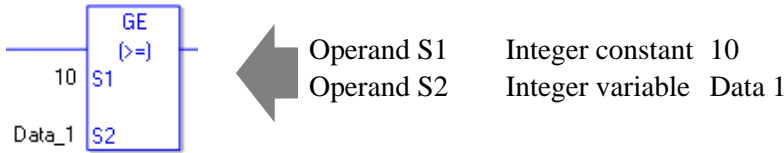
◆ **Explanation of the GE Instruction**

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.

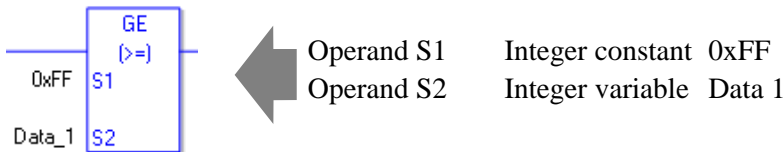
Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



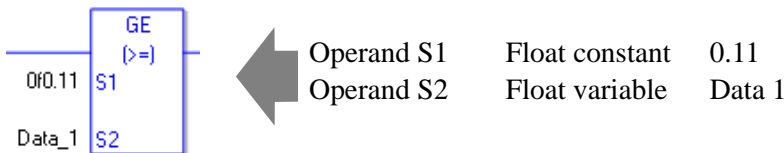
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case “x”) is input, the following values become hexadecimal values.



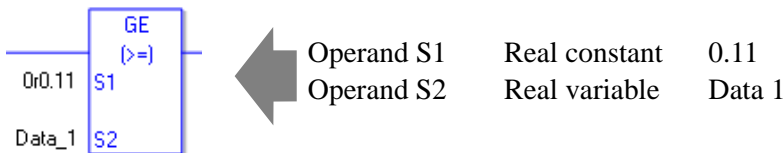
When entering float constants in operands S1 and S2

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



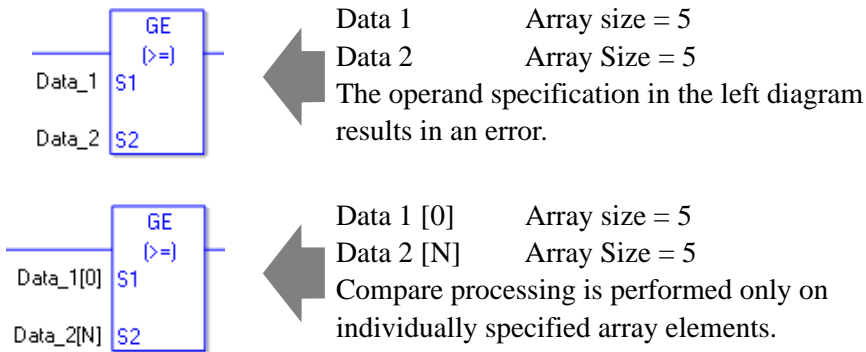
When entering real constants in operands S1 and S2

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).

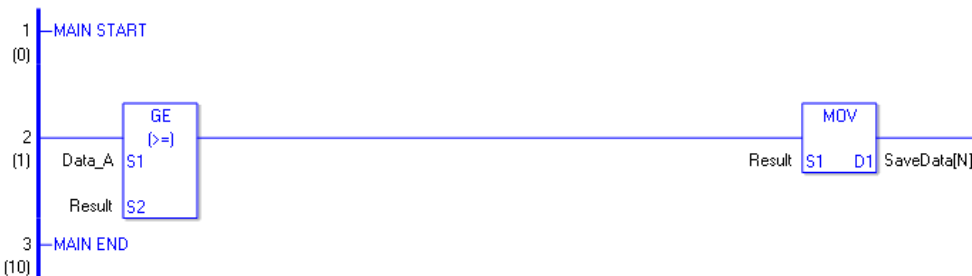
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

GE

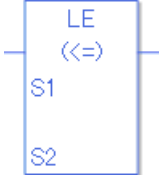
Compares integer variables and outputs the result in D1.



- (1) DataA and OperationResult are compared to determine whether DataA is greater than or equal OperationResult. 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 transition)		Comparison	3 to 9

◆ **Operand Settings**

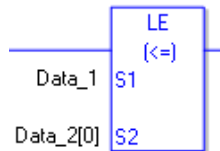
The following table lists the specifiable contents of operands S1 and S2 for the LE instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ step\} + \{Data\ 2\ [0] = 2\ steps\} + \{1\ step\} = 4\ steps$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the specifiable contents of the S1 and S2 operands for the LE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer (including IO)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]	2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	4	○
	Float	Specify float variable	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	Specify real variable	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
	Time	.HR/.MIN/.SEC only	2	○
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	

Continued

Explanations of Each Instruction

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

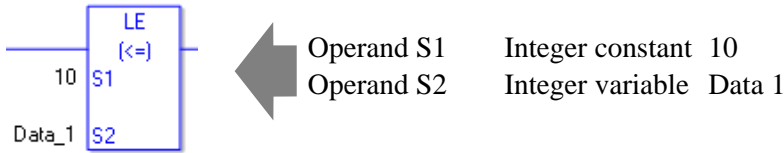
◆ **Explanation of LE Instruction**

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.

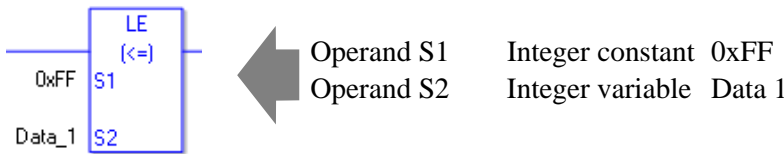
Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



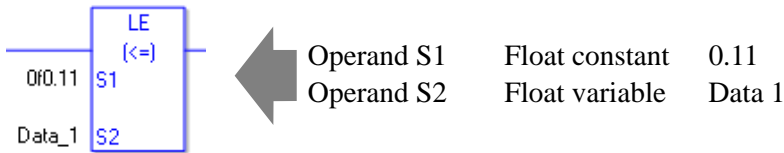
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case “x”) is input, the following values become hexadecimal values.



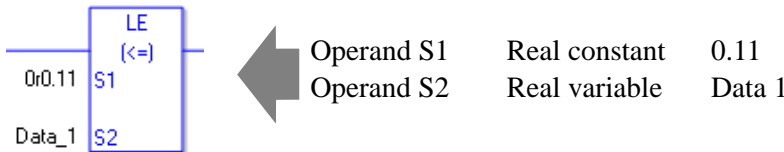
When entering float constants in operands S1 and S2

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



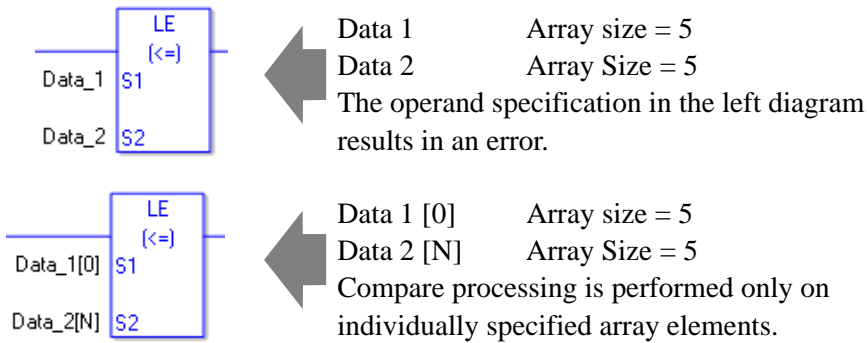
When entering real constants in operands S1 and S2

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).

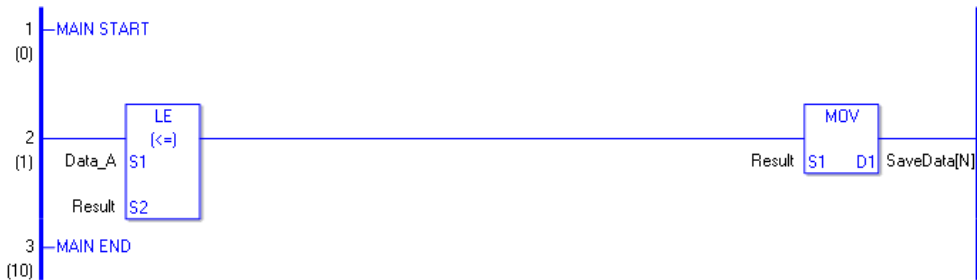
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

LE

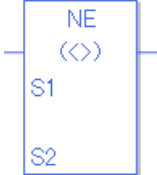
Compares integer variables and outputs the result in D1.



(1) DataA and OperationResult are compared to determine whether DataA is less than or equal to the OperationResult. 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 transition)		Comparison	3 to 9

◆ **Operand Settings**

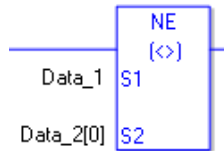
The following table lists the specifiable contents of operands S1 and S2 for the NE instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{ \text{Data 1} = 1 \text{ step} \} + \{ \text{Data 2 [0]} = 2 \text{ steps} \} + \{ 1 \text{ step} \} = 4 \text{ steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the specifiable contents of operands S1 and S2 for the NE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	1	○	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○	
Symbol	Bit	—	—	×	
	Word	—	1	○	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		1	○
		Specify integer variable [constant] or Specify integer variable B/W [constant]		2	○
		Specify integer variable [variable] or Specify integer variable B/W [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		4	○
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		2	○
	Counter	.PV/.CV only		2	○
	Date	.YR/ .MO/ .DAY only		2	○
	Time	.HR/ .MIN/ .SEC only		2	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		2	○	

Continued

Explanations of Each Instruction

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

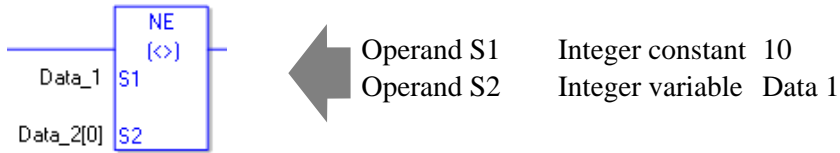
◆ **Explanation of NE Instruction**

The NE instruction is a compare instruction. The NE instruction compares S1 with S2. If the result of the comparison is $S1 \neq S2$, 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.

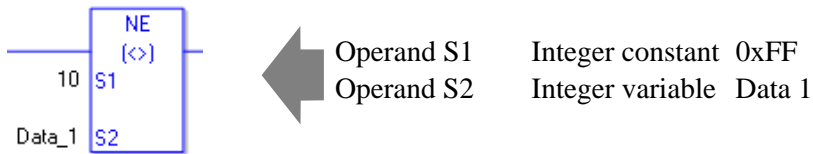
Refer to the following for specifying a constant.

When operand S1 or S2 is an integer constant



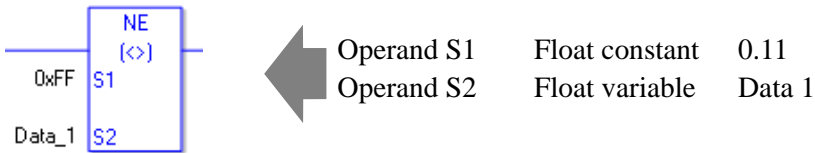
When entering hexadecimal values in operands S1 and S2

When 0x (zero and lower case “x”) is input, the following values become hexadecimal values.



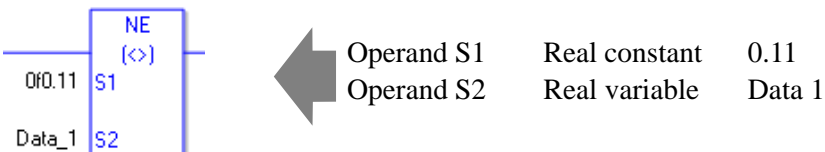
When entering float constants in operands S1 and S2

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



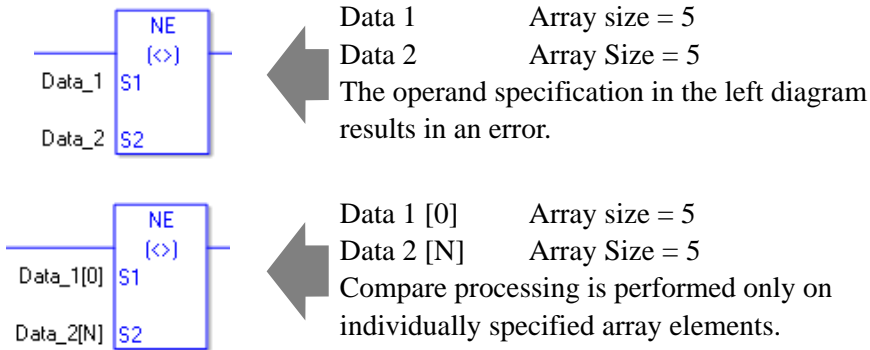
When entering real constants in operands S1 and S2

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).

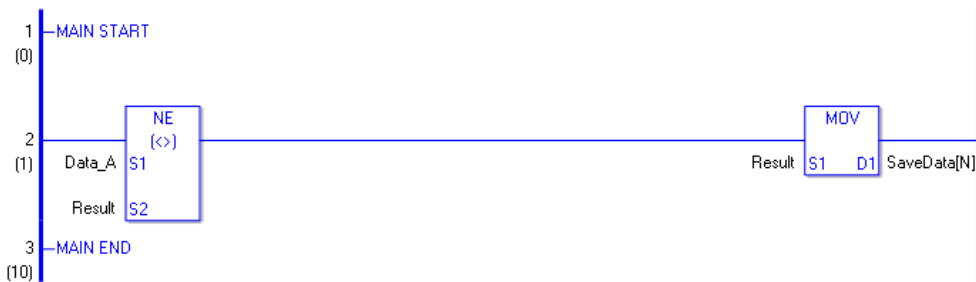
You cannot specify entire arrays for operands S1 or S2. An error will occur even if the specified array variables are the same type.



Program Example

NE

Compares integer variables and outputs the result in D1.

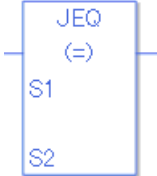


(1) DataA and OperationResult are compared to determine whether DataA is not equal to OperationResult. If the result of the NE instruction is S1 <> S2, the NE instruction passes power. Then the instruction to the right of the NE instruction is executed. In the above diagram, it's the MOV instruction.

30.5.16 Compare (Time)

■ JEQ (Equal)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
JEQ (= - level transition)		Time Compare	3

◆ Operand Settings

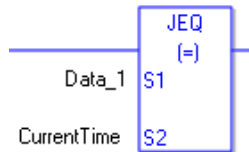
The following shows the configurable conditions for Operands (S1, S2) in the SEQ instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ Step\} + \{Current\ Time = 1\ Step\} + \{1\ Step\} = 3\ Steps$$

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	Variable Settings	Description
Variable Name.HR	Integer Variable	Hours are input in BCD.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the JEQ instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	Specify real variable		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC Structure elements are not specified.		1	○
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×

Continued

Explanations of Each Instruction

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

◆ **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 $S1 = S2$.

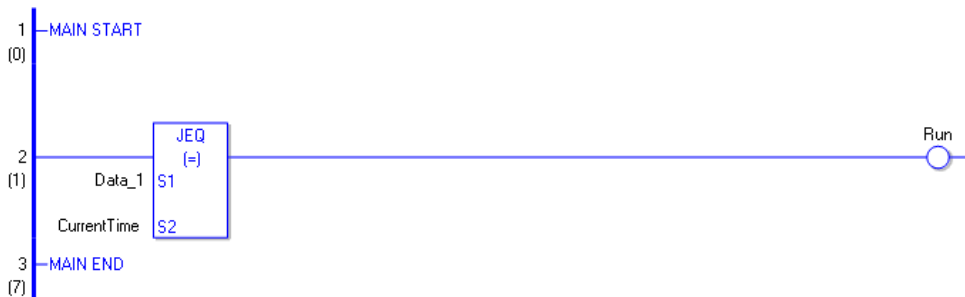
The hour, minute, and second variables are compared simultaneously. To compare a time of 10:20, input 0 for the seconds.

When using JEQ instructions, the only variables you can specify in operands S1 and S2 are time variables.

Program Example

JEQ

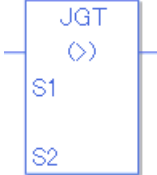
Compares the time variables and determines the result with the coil.



(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 (Greater Than - level transition)		Time Compare	3

◆ **Operand Settings**

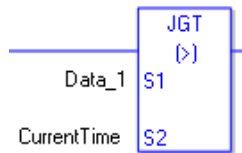
The following shows the configurable conditions for Operands (S1, S2) in the JGT instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ Step\} + \{Current\ Time = 1\ Step\} + \{1\ Step\} = 3\ Steps$$

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	Variable Settings	Description
Variable Name.HR	Integer Variable	Hours are input in BCD.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.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: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC Structure elements are not specified.		1	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

◆ **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 $S1 > S2$.

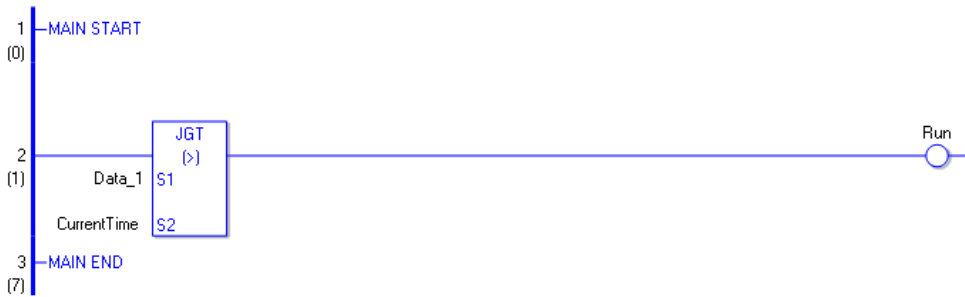
The hour, minute, and second variables are compared simultaneously. To compare a time of 10:20, input 0 for the seconds.

When using the JGT instruction, the only variables you can specify in operands S1 and S2 are time variables.

Program Example

JGT

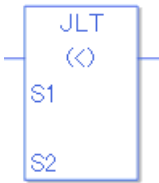
Compares the time variables and determines the result with the coil.



- (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 (Less Than - level transition)		Time Compare	3

◆ **Operand Settings**

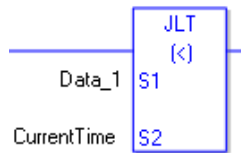
The following shows the configurable conditions for Operands (S1, S2) in the JLT instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ Step\} + \{Current\ Time = 1\ Step\} + \{1\ Step\} = 3\ Steps$$

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	Variable Settings	Description
Variable Name.HR	Integer Variable	Hours are input in BCD.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the JLT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC Structure elements are not specified.		1	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

◆ **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 $S1 < S2$.

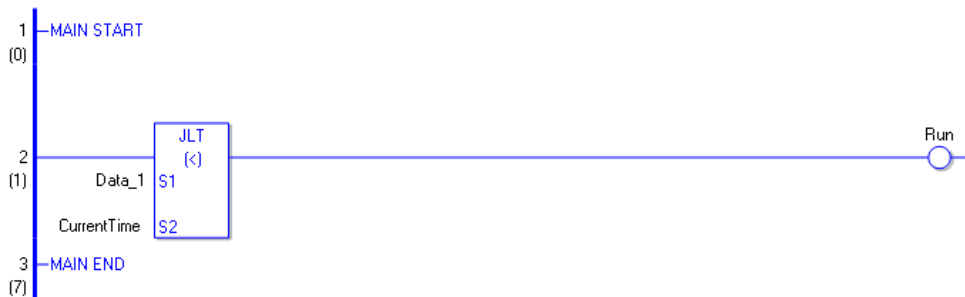
The hour, minute, and second variables are compared simultaneously. To compare a time of 10:20, input 0 for the seconds.

When using the JLT instruction, the only variables you can specify in operands S1 and S2 are time variables.

Program Example

JLT

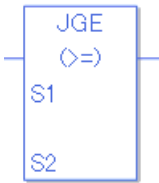
Compares the time variables and determines the result with the coil.



- (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 (Greater Than or Equal To - level transition)		Time Compare	3

◆ **Operand Settings**

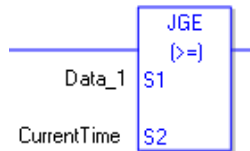
The following shows the configurable conditions for Operands (S1, S2) in the JGE instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ Step\} + \{Current\ Time = 1\ Step\} + \{1\ Step\} = 3\ Steps$$

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	Variable Settings	Description
Variable Name.HR	Integer Variable	Hours are input in BCD.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the JGE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC Structure elements are not specified.		1	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

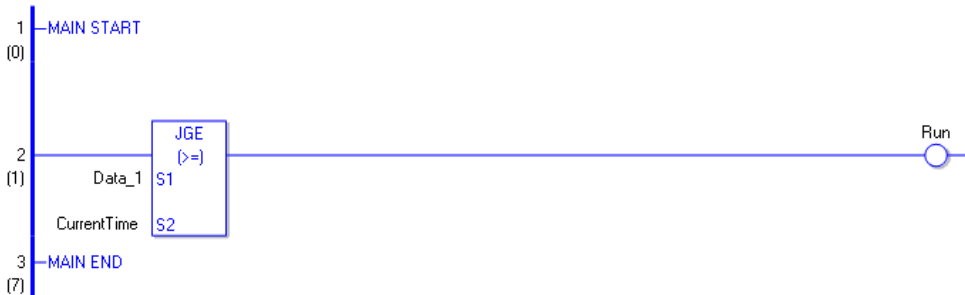
◆ **Explanation of the JGE Instruction**

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

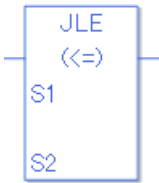
Compares the time variables and determines the result with the coil.



- (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 (Less Than or Equal To - level transition)		Time Compare	3

◆ **Operand Settings**

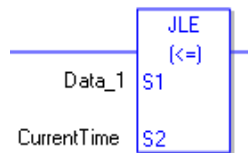
The following shows the configurable conditions for Operands (S1, S2) in the JLE instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ Step\} + \{Current\ Time = 1\ Step\} + \{1\ Step\} = 3\ Steps$$

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	Variable Settings	Description
Variable Name.HR	Integer Variable	Hours are input in BCD.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the JLE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC Structure elements are not specified.		1	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

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

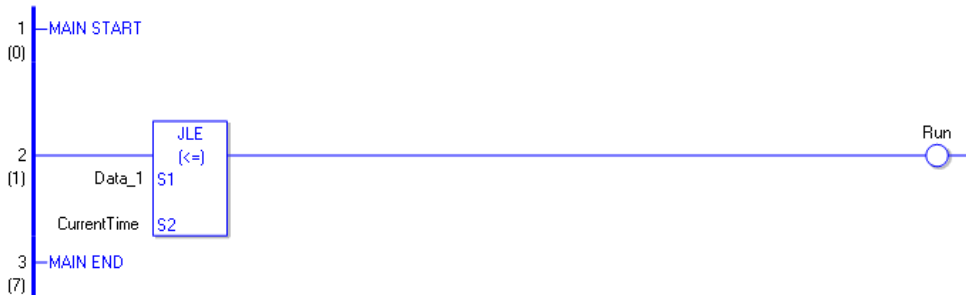
◆ **Explanation of the JLE Instruction**

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

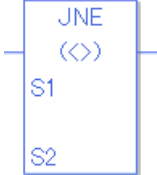
Compares the time variables and determines the result with the coil.



- (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 (Not Equal - level transition)		Time Compare	3

◆ **Operand Settings**

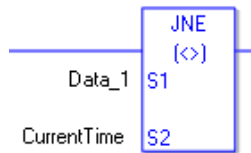
The following shows the configurable conditions for Operands (S1, S2) in the JNE instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ Step\} + \{Current\ Time = 1\ Step\} + \{1\ Step\} = 3\ Steps$$

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	Variable Settings	Description
Variable Name.HR	Integer Variable	Hours are input in BCD.
Variable Name.MIN	Integer Variable	Minutes are input in BCD.
Variable Name.SEC	Integer Variable	Seconds are input in BCD.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the JNE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC Structure elements are not specified.		1	○
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

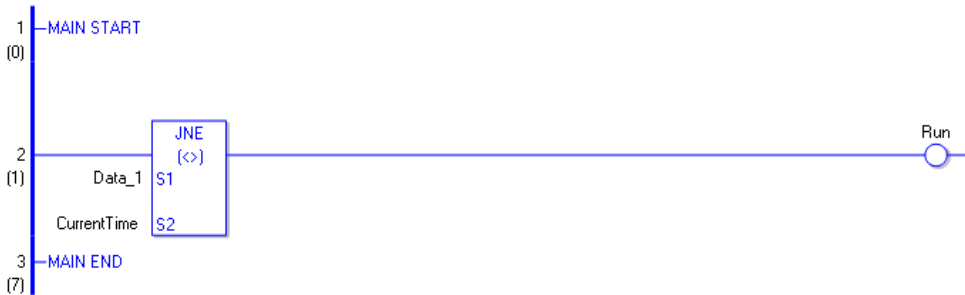
◆ **Explanation of the JNE Instruction**

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.

30.5.17 Compare (Date)

■ NEQ (=)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
NEQ (Equal - level transition)		Date Compare	3

◆ Operand Settings

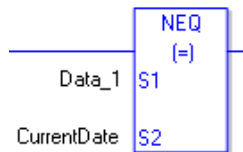
The following shows the configurable conditions for Operands (S1, S2) in the NEQ instruction.

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 = Total Number of Steps in One Instruction

e.g. 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 Step} + {Current Date = 1 Step} + {1 Step} = 3 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	Variable Settings	Description
Variable Name.YR	Integer Variable	The year is input in BCD.
Variable Name.MO	Integer Variable	The month is input in BCD.
Variable Name.DAY	Integer Variable	The day is input in BCD.

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the NEQ instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
			Specify real variable [variable]	—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/ .MO/ .DAY Structure elements are not specified.		1	○
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

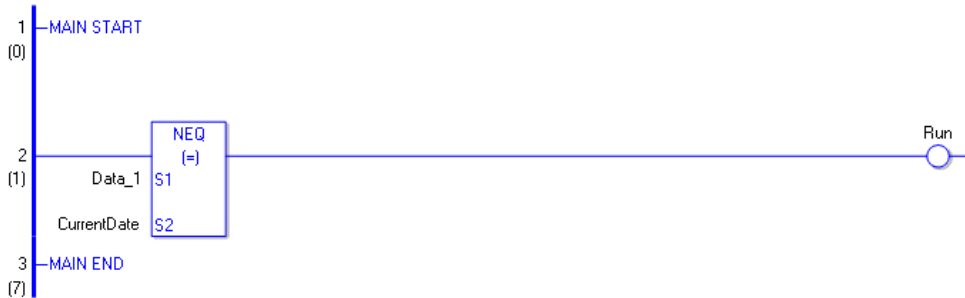
◆ **Explanation of the NEQ Instruction**

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

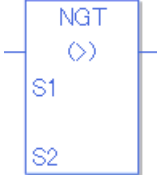
Compares the date variables and determines the result with the coil.



- (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 (Greater Than - level transition)		Date Compare	3

◆ **Operand Settings**

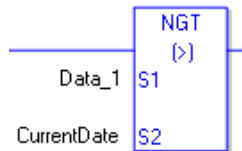
The following shows the configurable conditions for Operands (S1, S2) in the NGT instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ Step\} + \{Current\ Date = 1\ Step\} + \{1\ Step\} = 3\ Steps$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the NGT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
			Specify real variable [variable]	—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/ .MO/ .DAY Structure elements are not specified.		1	○
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

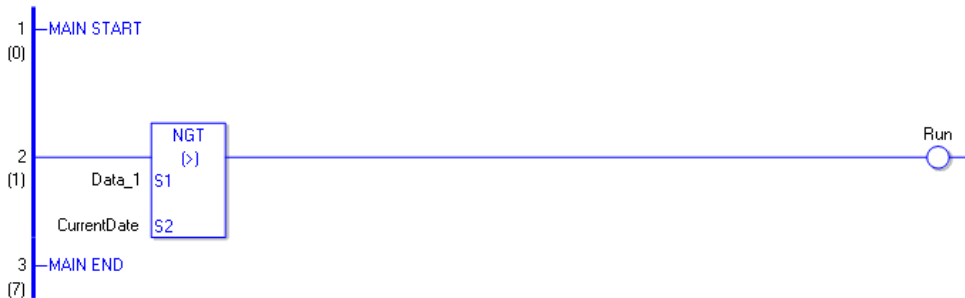
◆ **Explanation of the NGT Instruction**

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

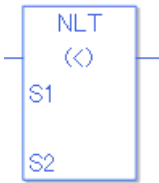
Compares the date variables and determines the result with the coil.



(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 (Less Than - level transition)		Date Compare	3

◆ **Operand Settings**

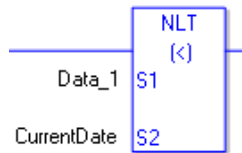
The following shows the configurable conditions for Operands (S1, S2) in the NLT instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ Step\} + \{Current\ Date = 1\ Step\} + \{1\ Step\} = 3\ Steps$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the NLT instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×		
External Device Address	Bit	—	—	×		
	Word	Specify by words only (Example: [PLC1]D0000)	—	×		
Internal Address	Bit	—	—	×		
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×		
Symbol	Bit	—	—	×		
	Word	—	—	×		
Variable Format	Bit	Specify a bit	—	×		
		Specify bit array ([constant])	—	×		
		Specify bit array ([variable])	—	×		
	Integer (including IO)	Arrays and modifiers are not specified		—	×	
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×	
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×	
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×	
	Float	—		—	×	
		Specify float variable [constant]		—	×	
		Specify float variable [variable]		—	×	
	Real	—		—	×	
		Specify real variable [constant]		—	×	
			Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×	
	Counter	.PV/.CV only		—	×	
	Date	.YR/ .MO/ .DAY Structure elements are not specified.		1	○	
	Time	.HR/ .MIN/ .SEC only		—	×	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×		

Continued

Explanations of Each Instruction

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

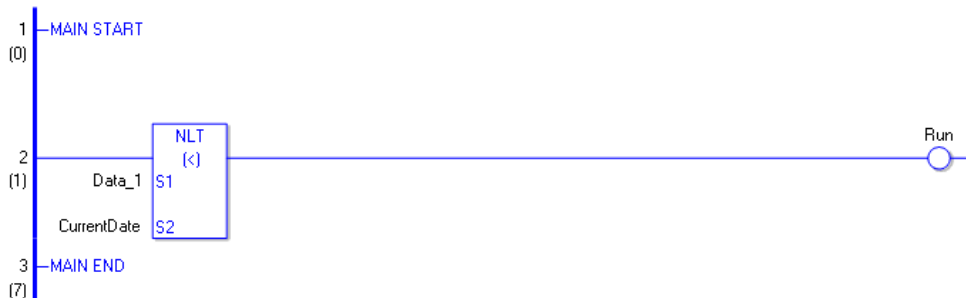
◆ **Explanation of the NLT Instruction**

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

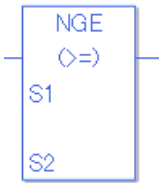
Compares the date variables and determines the result with the coil.



(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 (Greater Than or Equal To - level transition)		Date Compare	3

◆ **Operand Settings**

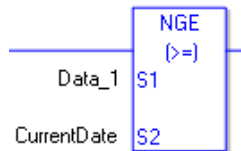
The following shows the configurable conditions for Operands (S1, S2) in the NGE instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{\text{Data 1} = 1 \text{ Step}\} + \{\text{Current Date} = 1 \text{ Step}\} + \{1 \text{ Step}\} = 3 \text{ Steps}$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the NGE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×		
External Device Address	Bit	—	—	×		
	Word	Specify by words only (Example: [PLC1]D0000)	—	×		
Internal Address	Bit	—	—	×		
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×		
Symbol	Bit	—	—	×		
	Word	—	—	×		
Variable Format	Bit	Specify a bit	—	×		
		Specify bit array ([constant])	—	×		
		Specify bit array ([variable])	—	×		
	Integer (including IO)	Arrays and modifiers are not specified		—	×	
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×	
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×	
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×	
	Float	—		—	×	
		Specify float variable [constant]		—	×	
		Specify float variable [variable]		—	×	
	Real	—		—	×	
		Specify real variable [constant]		—	×	
			Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×	
	Counter	.PV/.CV only		—	×	
	Date	.YR/ .MO/ .DAY Structure elements are not specified.		1	○	
	Time	.HR/ .MIN/ .SEC only		—	×	
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

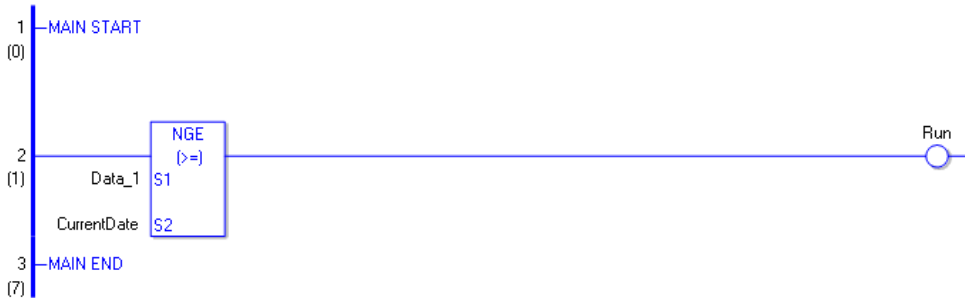
◆ **Explanation of the NGE Instruction**

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

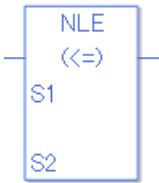
Compares the date variables and determines the result with the coil.



(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 (Less Than or Equal To - level transition)		Date Compare	3

◆ **Operand Settings**

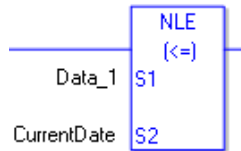
The following shows the configurable conditions for Operands (S1, S2) in the NLE instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{\text{Data 1} = 1 \text{ Step}\} + \{\text{Current Date} = 1 \text{ Step}\} + \{1 \text{ Step}\} = 3 \text{ Steps}$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the NLE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
			Specify real variable [variable]	—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/ .MO/ .DAY Structure elements are not specified.		1	○
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

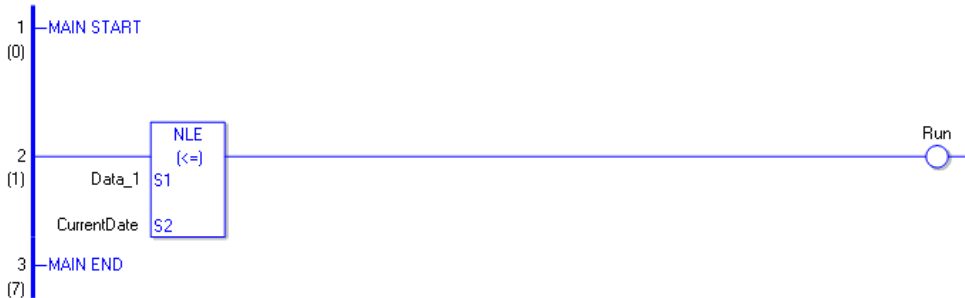
◆ **Explanation of the NLE Instruction**

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

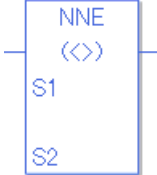
Compares the date variables and determines the result with the coil.



(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 (Not Equal - level transition)		Date Compare	3

◆ **Operand Settings**

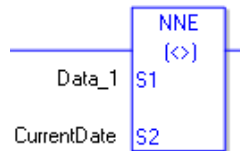
The following shows the configurable conditions for Operands (S1, S2) in the NNE instruction.

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 = Total Number of Steps in One Instruction

e.g. 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.)



$$\{Data\ 1 = 1\ Step\} + \{Current\ Date = 1\ Step\} + \{1\ Step\} = 3\ Steps$$

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operands (S1, S2) in the NNE instruction.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (including IO)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant] or Specify integer variable B/W [constant]		—	×
		Specify integer variable [variable] or Specify integer variable B/W [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
			Specify real variable [variable]	—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/ .MO/ .DAY Structure elements are not specified.		1	○
	Time	.HR/ .MIN/ .SEC only		—	×
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×	

Continued

Explanations of Each Instruction

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

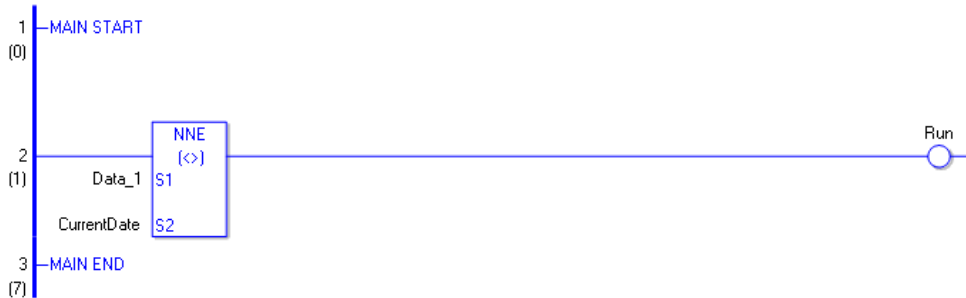
◆ **Explanation of the NNE Instruction**

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.

30.5.18 Convert (Data)

■ BCD/BCDP (BCD Convert)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BCD (BCD Convert - level transition)		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

e.g. 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 [Indirectly Specify] = 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 lists the configurable conditions for Operands (S1, D1) in the BCD/BCDP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) S1 = I/O Possible D1 = I Not Possible	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format *(Notes 2) D1 = Not Possible	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_*(Notes 2)	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV only	2	○	
	N_	.YR/.MO/.DAY only	2	○	
	J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○		
Constant *(Notes 3) D1 = Not Possible	Float	±1.175494351e-38 to ±3.402823466e+38	—	×	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	×	
	Integer *(Notes 3)	0 to 99999999	1	○	

◆ **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.

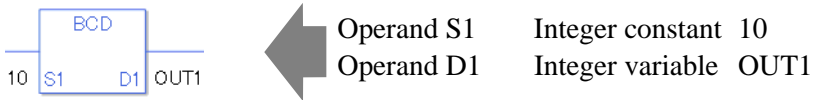
The BCD/BCDP instructions are always conducted. The maximum convertible value of operand S1 is 0x5F5E0FF. If you attempt a conversion above that, no value will be stored in D1.

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.

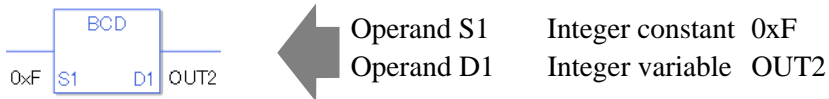
Refer to the following for specifying a constant.

When operand D1 is an integer variable



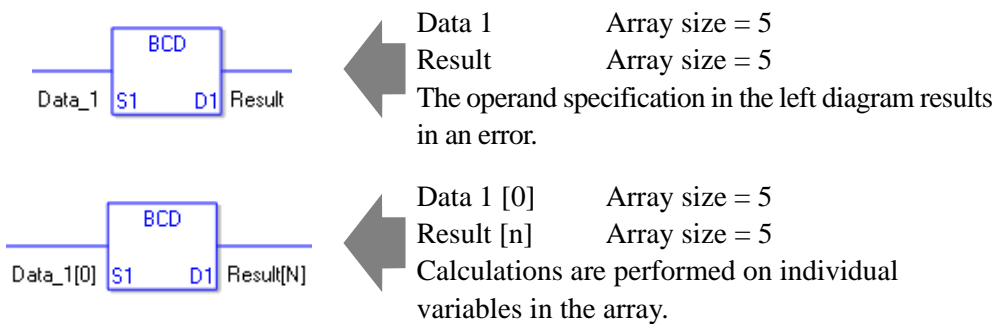
When operand D1 is an integer variable and you want to input hexadecimal values in operand S1.

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

(Notes)

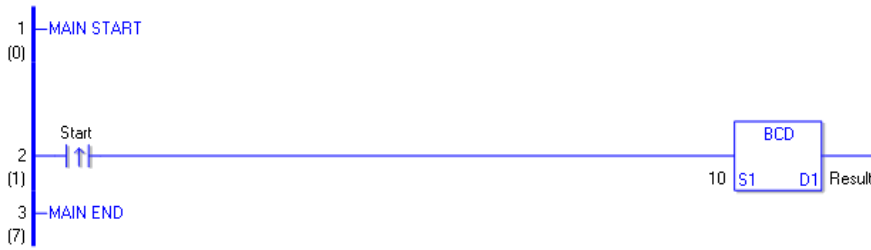
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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 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.

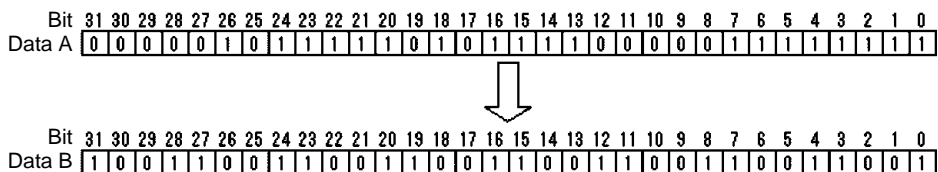
Program Example

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).

e.g. When a value “99999999” is stored in S1 (Data A) and is converted into BCD (binary coded decimal) in D1 (Data B)



■ **BIN/BINP (BIN Convert)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
BIN (BIN Convert - level transition)		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 = of Steps in One Instruction

e.g. 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 [Indirectly Specify] = 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: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) S1 = I/O Possible D1 = I Not Possible	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format *(Notes 2) D1 = Not Possible	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_*(Notes 2)	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV only	2	○	
	N_	.YR/.MO/.DAY only	2	○	
	J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○		
Constant *(Notes 3) D1 = Not Possible	Float	±1.175494351e-38 to ±3.402823466e+38	—	×	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	×	
	Integer *(Notes 3)	0 to 99999999 (BCD value)	1	○	

◆ **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.

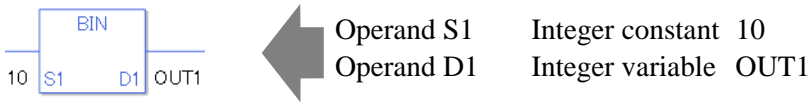
The BIN/BINP instructions are always conducted. The maximum convertible value of operand S1 is 0x5F5E0FF. If you attempt a conversion greater than that, no value will be stored in D1.

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.

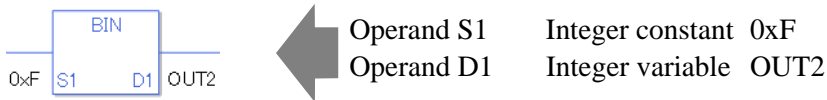
Refer to the following for specifying a constant.

When operand D1 is an integer variable



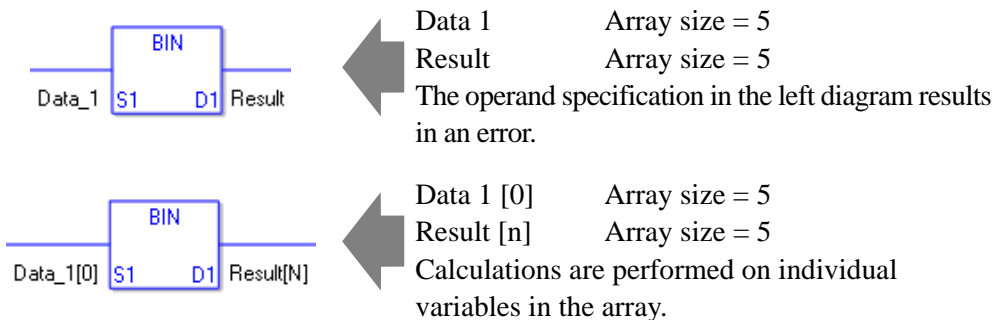
When operand D1 is an integer variable and you want to input hexadecimal values in operand S1.

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

(Notes)

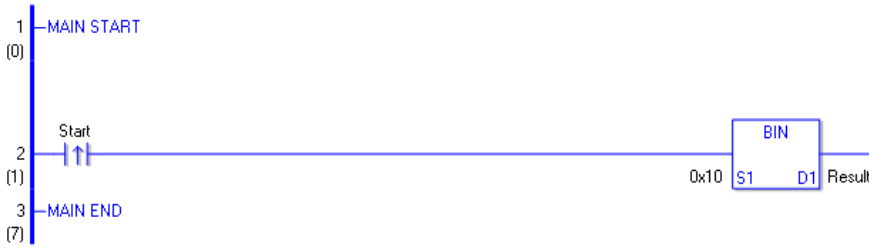
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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.

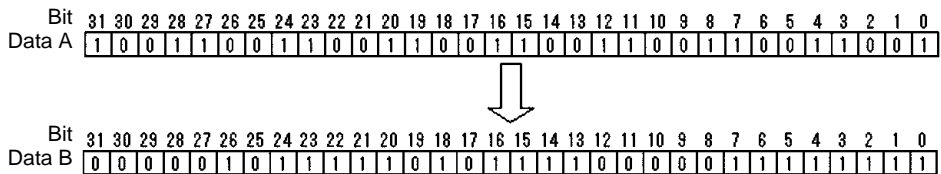
Program Example

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).

e.g. When BCD data “99999999” is set in S1 (Data A) and is converted into BIN in D1 (Data B)



■ ENCO/ENCOP (Encode)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
ENCO (Encode - level transition)		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

e.g. 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.)



$$\{Data\ 1\ [0] = 2\ Steps\} + \{Conversion\ Result\ [Indirectly\ Specify] = 3\ Steps\} + \{1\ Step\} = 6\ Steps$$

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: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○	
Internal Address	Bit	—	—	×	
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○	
Symbol	Bit	—	—	×	
	Word	—	1	○	
Variable Format *(Notes 1) S1 = I/O Possible D1 = I Not Possible	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer *(Notes 1)	Arrays and modifiers are not specified		1	○
		Specify integer variable [constant] array		2	○
		Specify integer variable [variable]		3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		2	○
	Counter	.PV/ .CV only		2	○
	Date	.YR/ .MO/ .DAY only		2	○
Time	.HR/ .MIN/ .SEC only		2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		2	○	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format *(Notes 2) D1 = Not Possible	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_*(Notes 2)	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV only	2	○	
	N_	.YR/.MO/.DAY only	2	○	
	J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○		
Constant *(Notes 3) D1 = Not Possible	Float	±1.175494351e-38 to ±3.402823466e+38	—	×	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	×	
	Integer *(Notes 3)	-2147483648 to 2147483647	1	○	

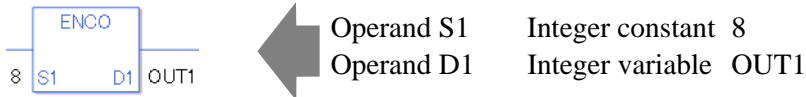
◆ **Explanation of the ENCO/ENCOP Instructions**

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.

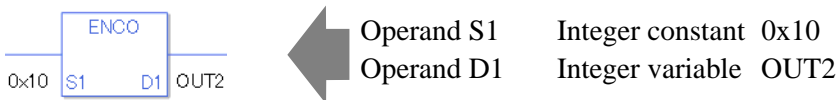
Refer to the following for specifying a constant.

When operand D1 is an integer variable

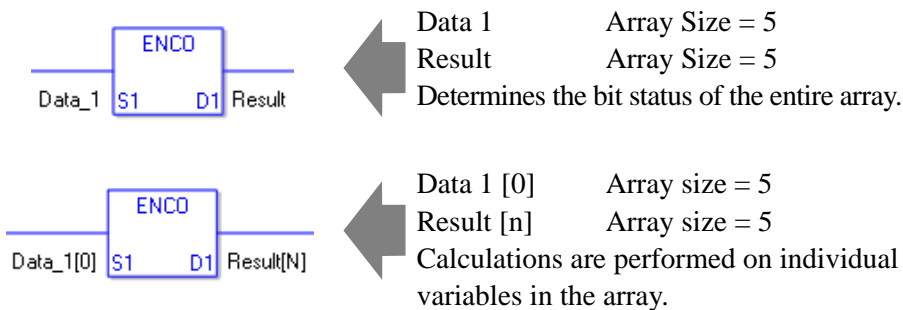


When operand D1 is an integer variable and you want to input hexadecimal values in operand S1.

When 0x (zero and lower case “x”) is input, the following values 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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

(Notes)

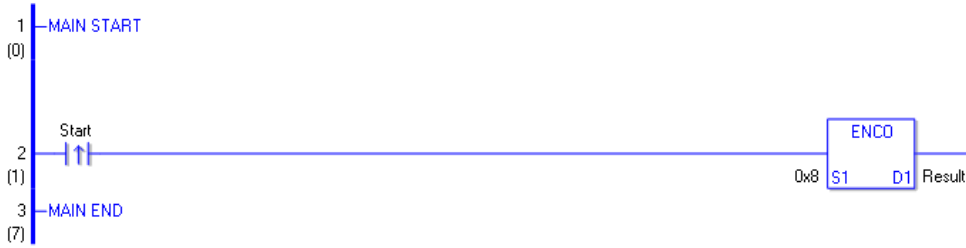
When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

ENCO

Converts a constant and stores the converted value in the result data.



- (1) 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.

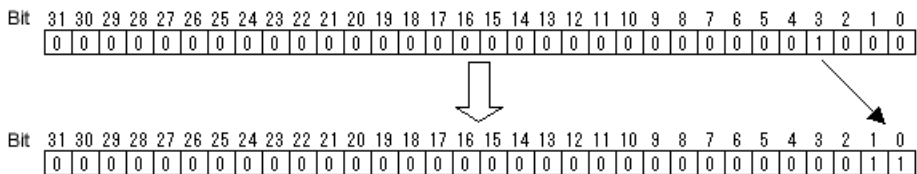
Program Example

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).

e.g. 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 transition)		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

e.g. 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 [Indirectly Specify] = 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: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Only a word is specified. (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) S1 = I/O Possible D1 = I Not Possible	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant] array	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	—	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	—	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/ .CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
Time	.HR/ .MIN/ .SEC only	2	○	
PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○	

Continued

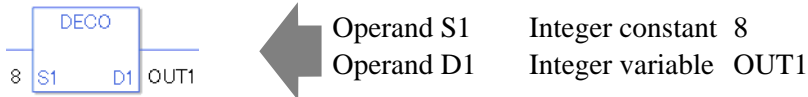
Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format *(Notes 2) D1 = Not Possible	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_*(Notes 2)	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	—	×	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV only	2	○	
	N_	.YR/.MO/.DAY only	2	○	
	J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○		
Constant *(Notes 3) D1 = Not Possible	Float	±1.175494351e-38 to ±3.402823466e+38	—	×	
	Real	±2.2250738585072014e-308 to ±1.7976931348623158e+308	—	×	
	Integer *(Notes 3)	0 to 131071 (Specified array)	1	○	

◆ **Explanation of the DECO/DECOP Instructions**

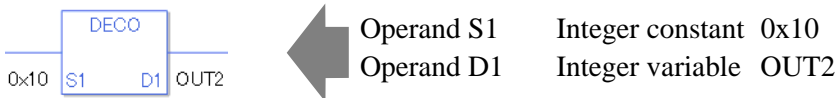
The DECO/DECOP instructions decode values. The value in S1 is decoded and saved in D1. The single bit position in D1 corresponding to the value in S1 is turned ON. When you use an output array, you can decode a bit position up to the maximum ($4096 \times 32 - 1 = 131071$). The DECO/DECOP instructions 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

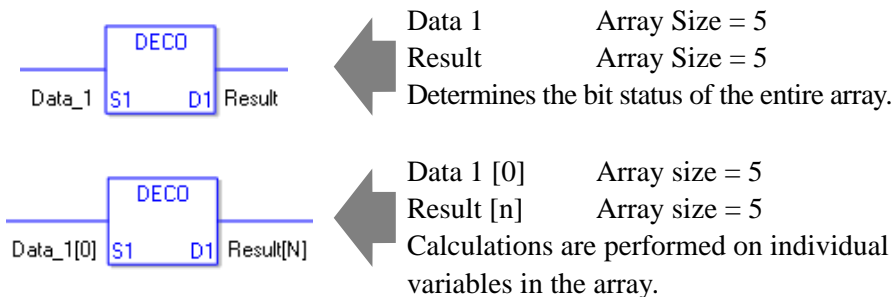


When operand D1 is an integer variable and you want to input hexadecimal values in operand S1.

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 result of the execution is 0, #L_CalcZero turns ON. If the execution results in an error, #L_CalcErrCode stores the error code.



(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

■ RAD/RADP (Convert to Radians)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
RAD (Convert to Radian - level transition)		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

e.g. 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: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant *(Notes 1) D1 = Not Possible	Float *(Notes 1)	±1.175494351e-38 to ±3.402823466e+38	1	○	
	Real *(Notes 1)	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of the RAD/RADP Instructions**

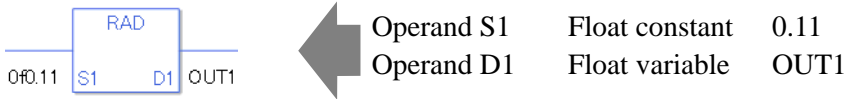
The RAD/RADP instructions convert values to radians. When the RAD instruction is executed, the value S1 of an angle in degrees is converted to radians and stored in D1. π is approximately 3.1415926535897 (real). The RAD/RADP instructions always pass power. When using RAD/RADP instructions, an error will 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.

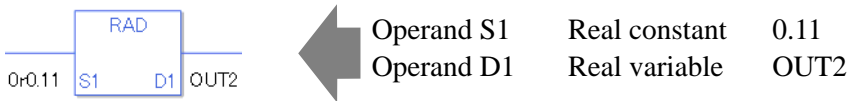
When operand D1 is a float variable

When Of (zero and lower case “f”) is input, the following values are interpreted as float values.



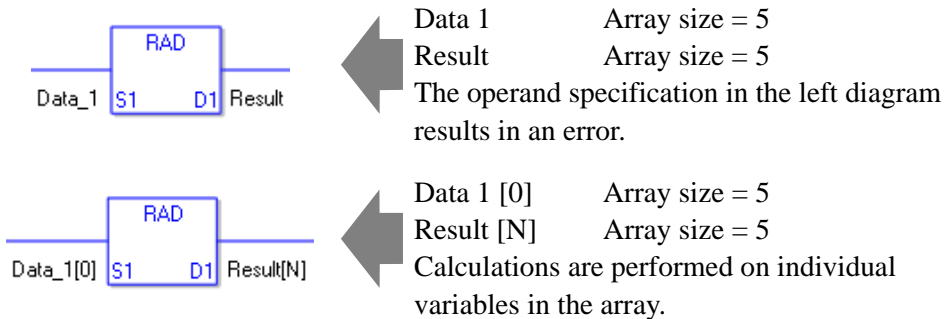
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

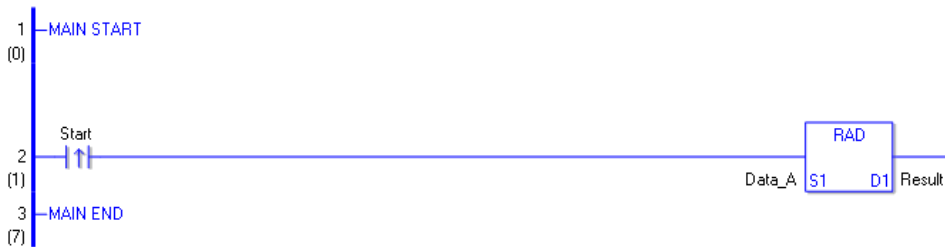
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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 transition)		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

e.g. 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: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Specify float variable		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Specify real variable		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV only	—	×	
	N_	.YR/.MO/.DAY only	—	×	
	J_	.HR/.MIN/.SEC only	—	×	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	—	×		
Constant *(Notes 1) D1 = Not Possible	Float *(Notes 1)	±1.175494351e-38 to ±3.402823466e+38	1	○	
	Real *(Notes 1)	±2.2250738585072014e-308 to ±1.7976931348623158e+308	2	○	
	Integer	-2147483648 to 2147483647	—	×	

◆ **Explanation of the DEG and DEGP Instructions**

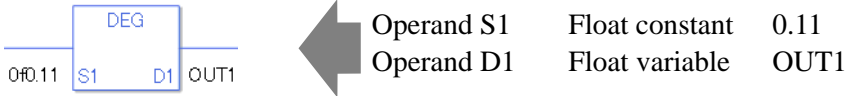
The DEG/DEGP instructions convert values to degrees. The unit of angular measure, radian, is converted to degrees and stored in D1.

π is used as 3.1415926535897. The DEG/DEGP instructions always pass power. When using DEG/DEGP instructions, an error will occur if the variables specified in operands S1 and D1 are not the same.

Refer to the following for specifying a constant.

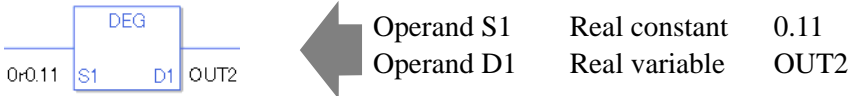
When operand D1 is a float variable

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



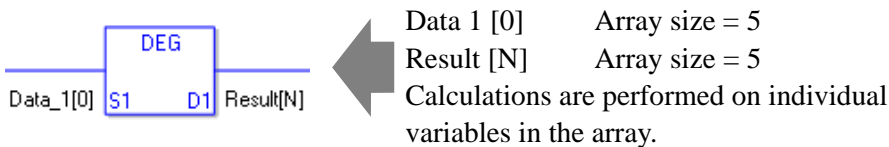
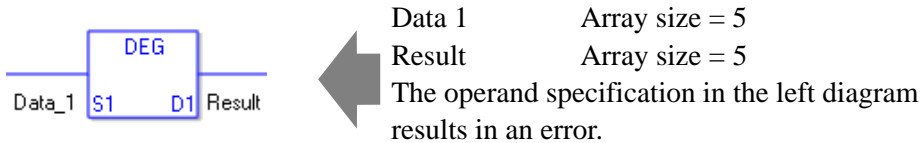
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

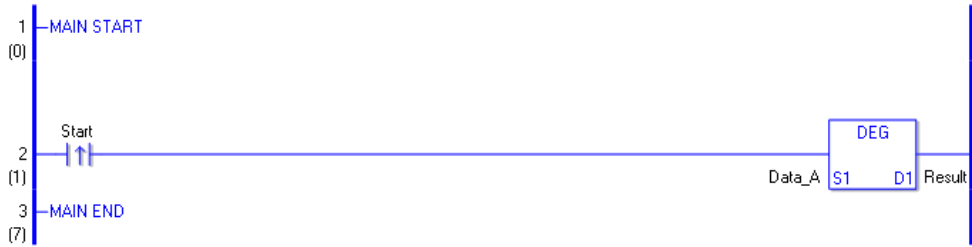
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed DEG instruction will be stored in the system variables.

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 transition)		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

e.g. 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: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) D1 = I Not Possible	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	Arrays and modifiers are not specified	1	○
		Specify float variable [constant]	2	○
		Specify float variable [variable]	3	○
	Real	Arrays and modifiers are not specified	1	○
		Specify real variable [constant]	2	○
		Specify real variable [variable]	3	○
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○

Continued

Explanations of Each Instruction

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format *(Notes 2) D1 = Not Possible	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_*(Notes 2)	—	1	○	
	Q_	—	1	○	
	D_	Modifiers are not specified	—	1	○
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	1	○	
	T_	.PT/.ET only	2	○	
	C_	.PV/.CV only	2	○	
	N_	.YR/.MO/.DAY only	2	○	
	J_	.HR/.MIN/.SEC only	2	○	
U_	.KP/.TR/.TD/.PA/.BA/.ST only	2	○		
Constant *(Notes 3) D1 = Constant Not Possible	Float	$\pm 1.175494351e-38$ to $\pm 3.402823466e+38$	1	○	
	Real	$\pm 2.2250738585072014e-308$ to $\pm 1.7976931348623158e+308$	2	○	
	Integer	-2147483648 to 2147483647	1	○	

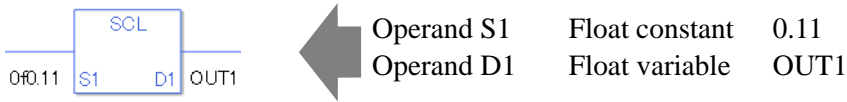
◆ **Explanation of the SCL/ SCLP Instructions**

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.

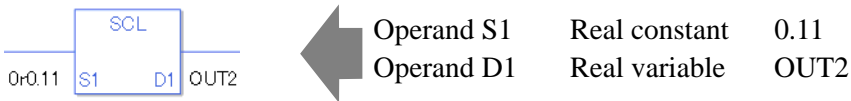
When operand D1 is a float variable

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



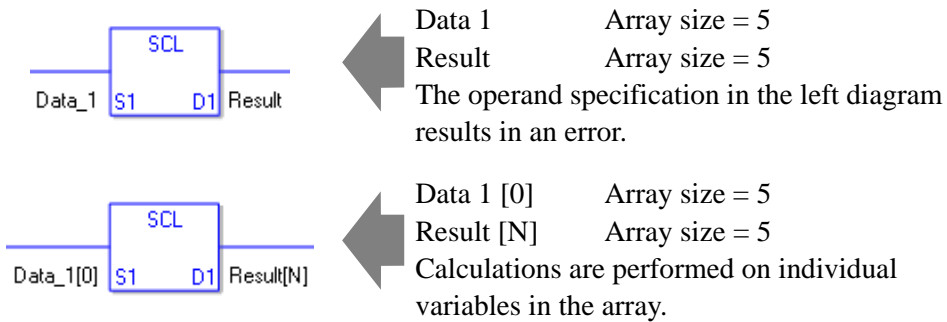
When operand D1 is a real variable

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

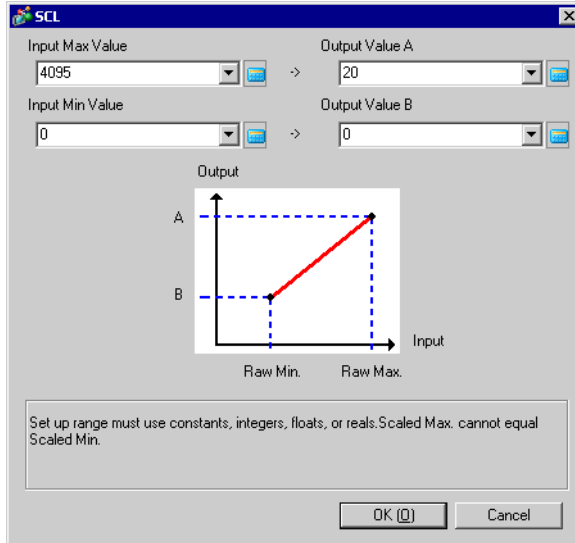
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

◆ **Upper and Lower Limits for Input and Output**

Double-click the SCL instruction to display the below 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.

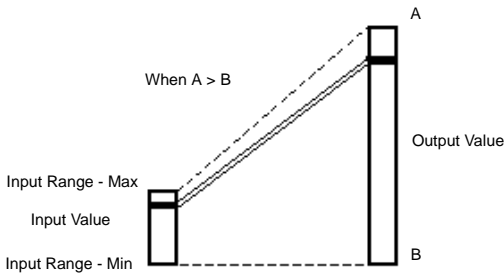
Array Variable Name: Data

Array Size: 5

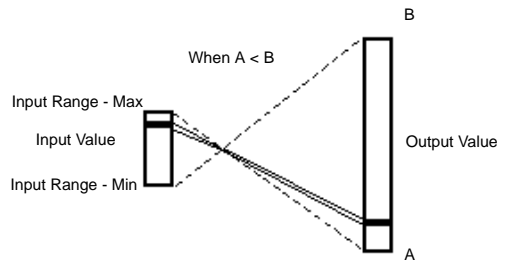
Possible: Data [0] Not Possible: Data [N]

(Notes 2) When inputting real or float variables in operands S1 and D1, add Or and Of to the constants specified for the maximum/minimum input values and output values A and B.

When output value A is greater than output value B



When output value A is greater than output value B

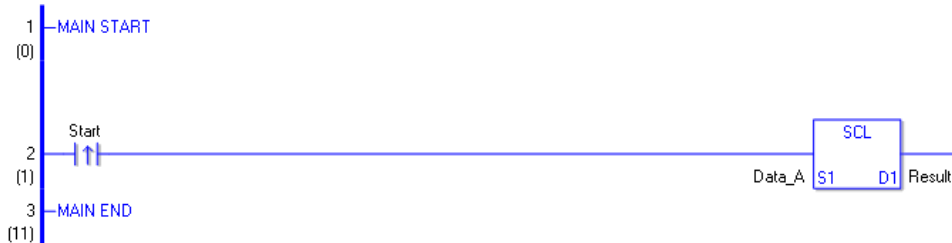


Program Example

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 have different ways of detecting when to execute. In the SCLP instruction, only the upward transition is detected and the SCLP instruction is executed even when using a normally open instruction. Even if the normally open instruction variable remains ON, the SCLP instruction is executed only once (for 1 scan).

30.5.19 Convert Type

■ I2F/I2FP (Integer → Float Conversion)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
I2F (Integer→ Float Conversion - level transition)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
I2FP (Integer→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

e.g. 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 table lists the configurable conditions for Operand (S1) in the I2F/I2FP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) including I/O	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	Arrays and modifiers are not specified	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	Arrays and modifiers are not specified	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the I2F/I2FP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

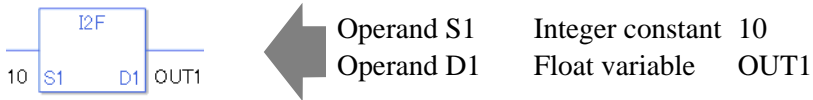
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	—	×	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV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

◆ **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.

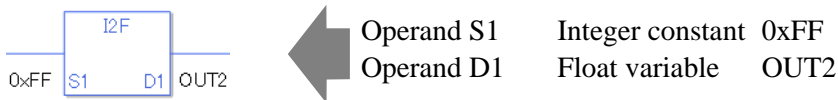
Refer to the following for specifying a constant.

When operand S1 is an integer constant



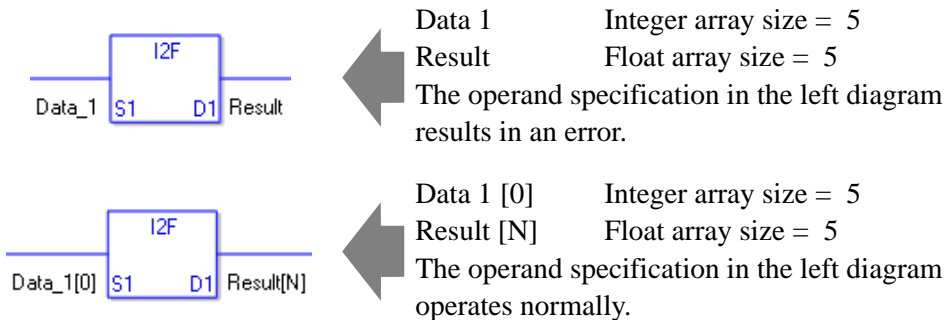
When operand S1 is an integer constant and you want to input a hexadecimal value in operand S1.

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

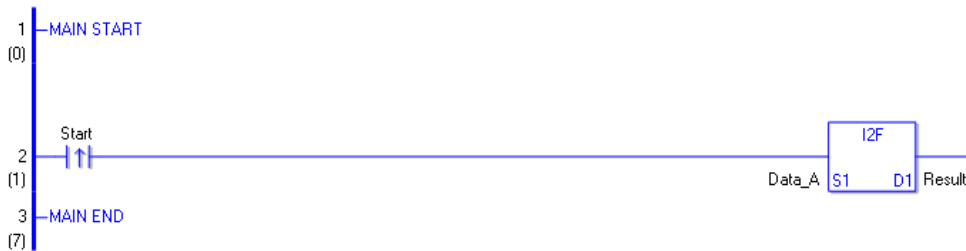
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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 → Real Conversion)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
I2R (Integer→ Real Conversion - level transition)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
I2RP (Integer→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

e.g. 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 Steps} + {Conversion Result [N] = 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 lists the configurable conditions for Operand (S1) in the I2R/I2RP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) including I/O	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	Arrays and modifiers are not specified	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	Arrays and modifiers are not specified	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the I2R/I2RP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

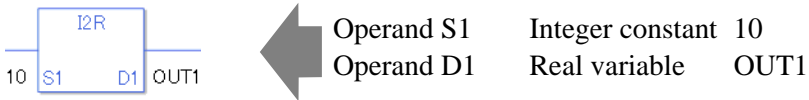
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV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

◆ **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.

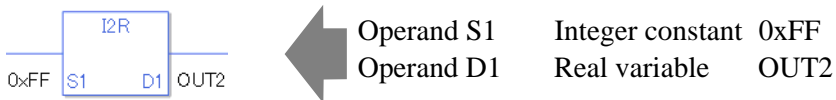
Refer to the following for specifying a constant.

When operand S1 is an integer constant



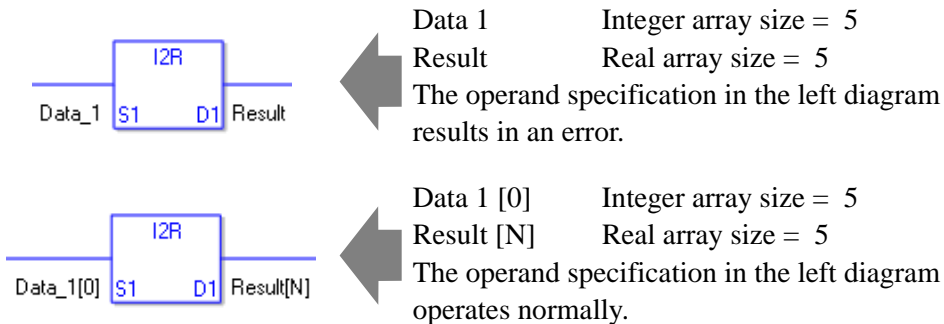
When operand S1 is an integer constant and you want to input a hexadecimal value in operand S1.

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

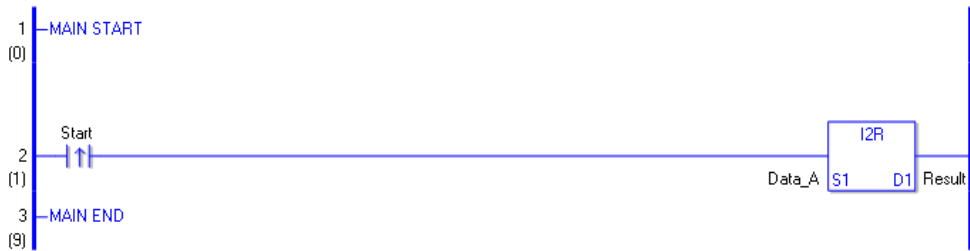
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

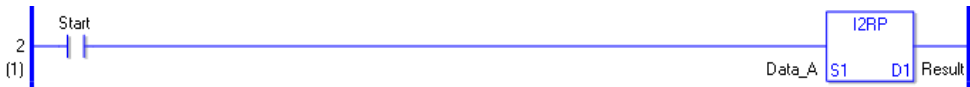
I2R



- (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 → Integer Conversion)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
F2I (Float→Integer Conversion - level transition)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
F2IP (Float→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

e.g. 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 Steps} + {Conversion Result [N] = 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 lists the configurable conditions for Operand (S1) in the F2I/F2IP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the F2I/F2IP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	Arrays and modifiers are not specified	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	Arrays and modifiers are not specified	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○

Continued

Explanations of Each Instruction

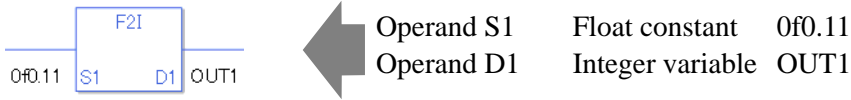
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	—	×
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	—	×
		D_****.B/W [address]	—	×
	F_	—	—	×
	R_	—	—	×
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY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

◆ **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.

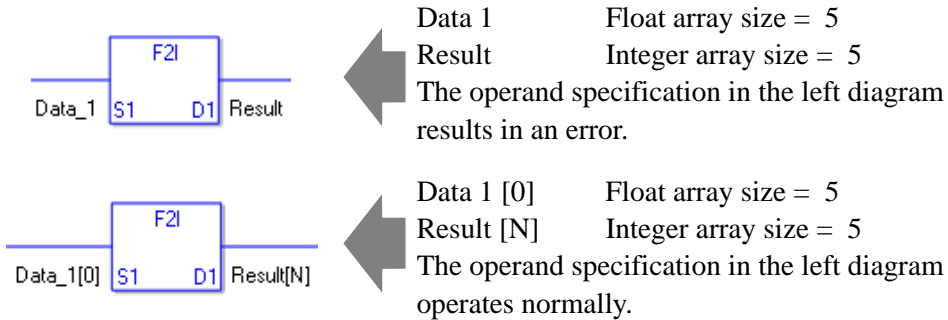
Refer to the following for specifying a constant.

When operand S1 is a float constant



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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

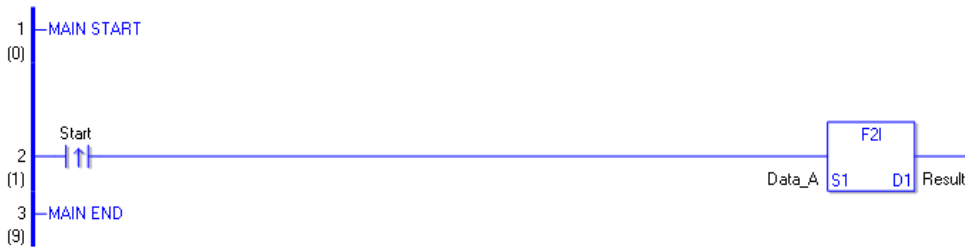
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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 → Real Conversion)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
F2R (Float→Real Conversion - level transition)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
F2RP (Float→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

e.g. 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 table lists the configurable conditions for Operand (S1) in the F2R/F2RP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the F2R/F2RP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer *(Notes 1)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Arrays and modifiers are not specified		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	Arrays and modifiers are not specified		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/ .CV only		—	×
	Date	.YR/ .MO/ .DAY only		—	×
	Time	.HR/ .MIN/ .SEC only		—	×
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only		—	×

Continued

Explanations of Each Instruction

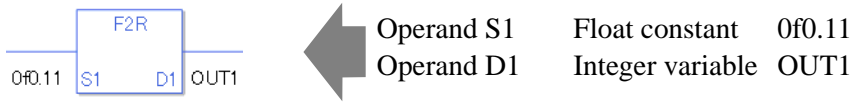
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	—	×	
	R_	—	1	○	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV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

◆ **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.

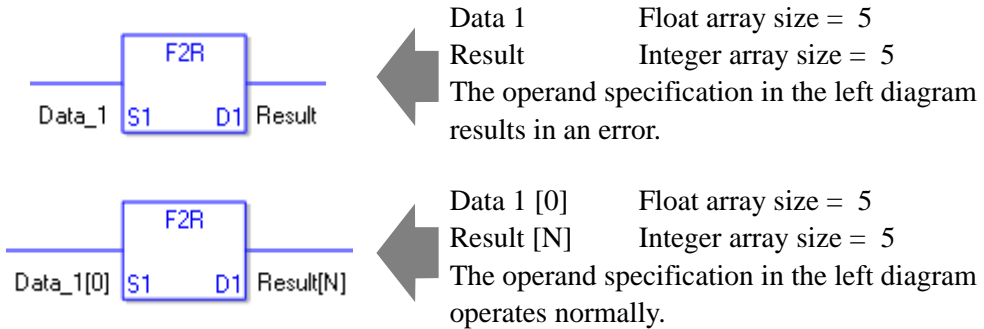
Refer to the following for specifying a constant.

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

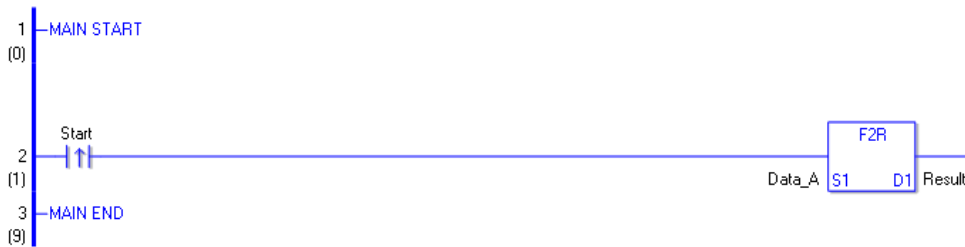
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

Program Example

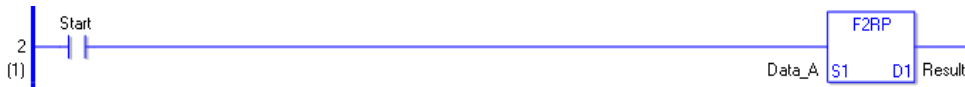
F2R



- (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 → Integer Conversion)**

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
R2I (Real→Integer Conversion - level transition)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
R2IP (Real→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

e.g. 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.)



$$\{ \text{Data 1 [0]} = 2 \text{ Steps} \} + \{ \text{Conversion Result [N]} = 3 \text{ Steps} \} + \{ 1 \text{ Step} \} = 6 \text{ Steps}$$

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in the R2I/R2IP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
	Time	.HR/.MIN/.SEC only		—	×
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the R2I/R2IP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	Arrays and modifiers are not specified	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	Arrays and modifiers are not specified	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
	Time	.HR/.MIN/.SEC only	2	○
	PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○

Continued

Explanations of Each Instruction

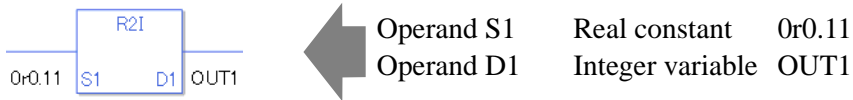
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
Address Format	X_	—	—	×
	Y_	—	—	×
	M_	—	—	×
	I_	—	—	×
	Q_	—	1	○
	D_	Modifiers are not specified	1	○
		D_****.B/W [constant]	—	×
		D_****.B/W [address]	—	×
	F_	—	—	×
	R_	—	—	×
	T_	.PT/.ET only	2	○
	C_	.PV/.CV only	2	○
	N_	.YR/.MO/.DAY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

◆ **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.

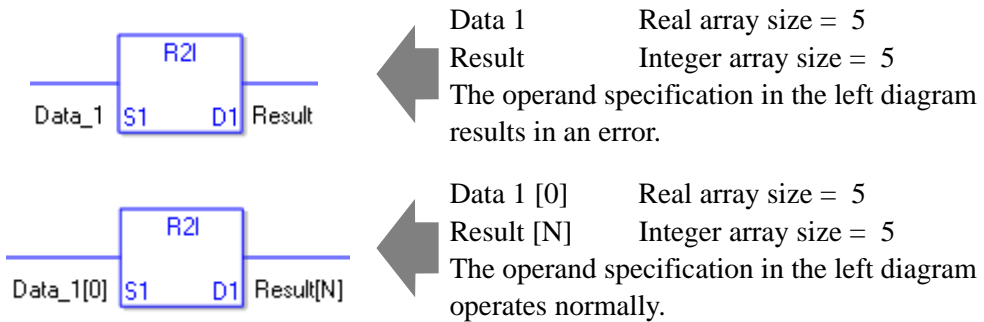
Refer to the following for specifying a constant.

When operand S1 is a real constant



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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

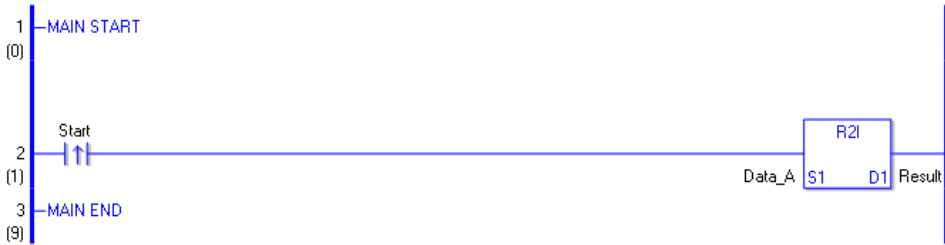
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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 → Float Conversion)

Symbols and Features

Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
R2F (Real→Float Conversion - level transition)		Type Convert	3 to 7
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
R2FP (Real→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

e.g. 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 Steps} + {Conversion Result [N] = 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 lists the configurable conditions for Operand (S1) in the R2F/R2FP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		1	○
		Specify real variable [constant]		2	○
		Specify real variable [variable]		3	○
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	.HR/.MIN/.SEC only		—	×	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the R2F/R2FP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer *(Notes 1)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	Arrays and modifiers are not specified		1	○
		Specify float variable [constant]		2	○
		Specify float variable [variable]		3	○
	Real	Arrays and modifiers are not specified		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
	Time	.HR/.MIN/.SEC only		—	×
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

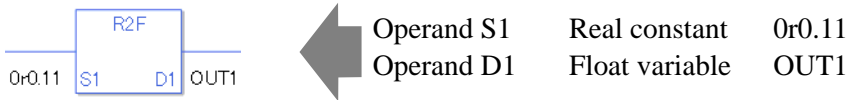
Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
Address Format	X_	—	—	×	
	Y_	—	—	×	
	M_	—	—	×	
	I_	—	—	×	
	Q_	—	—	×	
	D_	Modifiers are not specified	—	—	×
		D_****.B/W [constant]	—	—	×
		D_****.B/W [address]	—	—	×
	F_	—	1	○	
	R_	—	—	×	
	T_	.PT/.ET only	—	×	
	C_	.PV/.CV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

◆ **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.

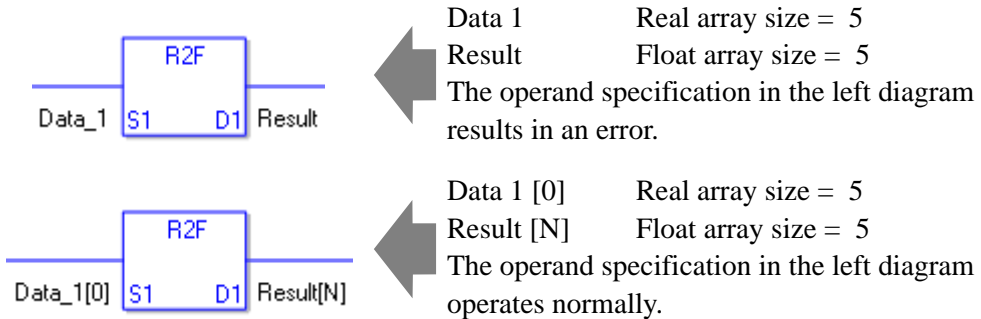
Refer to the following for specifying a constant.

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 result of the execution is 0, #L_CalcZero turns ON.

If the execution results in an error, #L_CalcErrCode stores the error code.

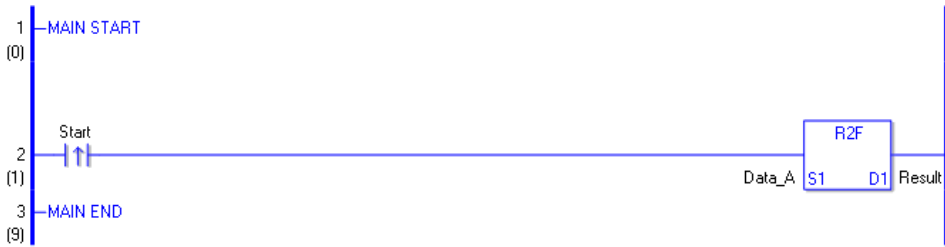
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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→Seconds Conversion - level transition)		Type Convert	3 to 5
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
H2SP (Time→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.

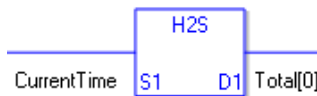
The actual number of steps in the H2S/H2SP instructions depends on the specified operands.

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

e.g. 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.)



{Elapsed Time = 1 Step} + {Total Seconds [0] = 2 Steps} + {1 Step} = 4 Steps

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in the H2S/H2SP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	Other than .HR / .MIN / .SEC		1	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the H2S/H2SP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	Arrays and modifiers are not specified	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	Arrays and modifiers are not specified	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/.MO/.DAY only	2	○
	Time	.HR/.MIN/.SEC only	2	○
PID	.KP/.TR/.TD/.PA/.BA/.ST only	2	○	

Continued

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

◆ **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 result of the execution is 0, #L_CalcZero turns ON.
If the execution results in an error, #L_CalcErrCode stores the error code.

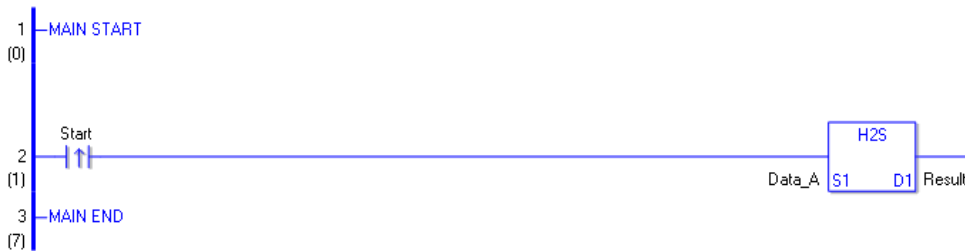
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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→Time Conversion - level transition)		Type Convert	3 to 5
Ladder Instruction Name	Ladder Symbol	Feature	Number of Steps
S2HP (Seconds→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

e.g. 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.)



{Elapsed Time = 1 Step} + {Total Seconds [0] = 2 Steps} + {1 Step} = 4 Steps

One last step is included in the instruction. Be sure to add that one step.

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (S1) in the S2H/ S2HP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×
External Device Address	Bit	—	—	×
	Word	Specify by words only (Example: [PLC1]D0000)	1	○
Internal Address	Bit	—	—	×
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	1	○
Symbol	Bit	—	—	×
	Word	—	1	○
Variable Format *(Notes 1) Output only	Bit	Specify a bit	—	×
		Specify bit array ([constant])	—	×
		Specify bit array ([variable])	—	×
	Integer *(Notes 1)	Arrays and modifiers are not specified	1	○
		Specify integer variable [constant]	2	○
		Specify integer variable [variable]	3	○
		Specify integer variable [constant/variable] .B/W [constant/variable]	—	×
	Float	Arrays and modifiers are not specified	—	×
		Specify float variable [constant]	—	×
		Specify float variable [variable]	—	×
	Real	Arrays and modifiers are not specified	—	×
		Specify real variable [constant]	—	×
		Specify real variable [variable]	—	×
	Timer	.PT/.ET only	2	○
	Counter	.PV/.CV only	2	○
	Date	.YR/ .MO/ .DAY only	2	○
	Time	.HR/ .MIN/ .SEC only	2	○
	PID	.KP/ .TR/ .TD/ .PA/ .BA/ .ST only	2	○

Continued

Explanations of Each Instruction

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

◆ **Operand Settings**

The following table lists the configurable conditions for Operand (D1) in the S2H/ S2HP instructions.

Name	Type	Condition	Number of Steps in the Operand	Possible: ○ Not Possible: ×	
External Device Address	Bit	—	—	×	
	Word	Specify by words only (Example: [PLC1]D0000)	—	×	
Internal Address	Bit	—	—	×	
	Word	Specify by words only (Example: [#INTERNAL]LS0000)	—	×	
Symbol	Bit	—	—	×	
	Word	—	—	×	
Variable Format	Bit	Specify a bit	—	×	
		Specify bit array ([constant])	—	×	
		Specify bit array ([variable])	—	×	
	Integer (not including I/O)	Arrays and modifiers are not specified		—	×
		Specify integer variable [constant]		—	×
		Specify integer variable [variable]		—	×
		Specify integer variable [constant/variable] .B/W [constant/variable]		—	×
	Float	—		—	×
		Specify float variable [constant]		—	×
		Specify float variable [variable]		—	×
	Real	—		—	×
		Specify real variable [constant]		—	×
		Specify real variable [variable]		—	×
	Timer	.PT/.ET only		—	×
	Counter	.PV/.CV only		—	×
	Date	.YR/.MO/.DAY only		—	×
Time	Other than .HR / .MIN / .SEC		1	○	
PID	.KP/.TR/.TD/.PA/.BA/.ST only		—	×	

Continued

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

◆ **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 result of the execution is 0, #L_CalcZero turns ON.
If the execution results in an error, #L_CalcErrCode stores the error code.

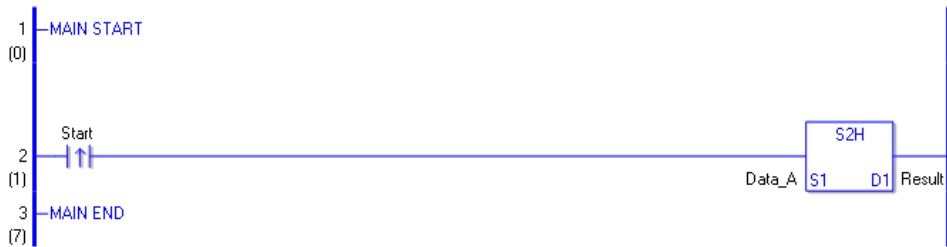
(Notes)

When checking the execution result with system variables, check the result after the instruction has been executed.

Note that when checking the state after multiple instructions have been executed, the result of the last processed instruction will be stored in the system variables.

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).