M-Series Controller

Basic Logic

Instruction Manual

January 2024 V1.0
Version:ATC/MMXS2310

Contents

Preface

Thank you for purchasing the M series controller. This manual focuses on controller logic instructions such as math operation instructions, timer, counter, and data conversion instructions.

■ Intended audience

The intended readers of this manual include the technical personnel of M series controller programming and debugging, who need to have a certain programmable controller-related basic knowledge and programming mindset.

■ Manual revision information

Version	Date	Content
V1.00	2023/8/23	First edition

■ Other information

- The content of this manual is edited based on product information and customer requirements, users who have questions or find errors in the contents of the manual are welcome to call HCFA or send an email to 400@hcfa.cn and follow the version number marked on the front cover to assist in the clarification.
- The contents of this manual, including text, pictures, logos, forms, etc., may not be reproduced or disseminated in any form without the authorization of the company. Otherwise, the company shall pursue the violator's legal responsibility in accordance with the law.

Chapter 1 Data types

The data types and corresponding data ranges supported by the M series controller are listed in the following table.

Classification	Data type	Data width	Data range	Initial value
Boolean	BOOL	1 bit	TRUE or FALSE	FALSE
Bit string	BYTE	1 byte	16#00 ~ 16#FF	0
	WORD	2 bytes	16#0000 ~ 16#FFFF	0
	DWORD	4 bytes	16#00000000 ~ 16#FFFFFFFF	0
	LWORD	8 bytes	16#0000000000000000 ~ 16#FFFFFFFFFFFFFFFF	0
Integer	USINT	1 byte	0 ~ 255	0
	UINT	2 bytes	0 ~ 65535	0
	UDINT	4 bytes	0 ~ 4294967295	0
	ULINT	8 bytes	0 ~ 18446744073709551615	0
	SINT	1 byte	-128 ~ +127	0
	INT	2 bytes	-32768 ~ +32767	0
	DINT	4 bytes	-2147483648 ~ +2147483647	0
	LINT	8 bytes	-9223372036854775808 ~ +9223372036854775807	0
Real number	REAL	4 bytes	-3.402823e+38 ~ -1.175495e-38, 0, 1.175495e-38 ~ 3.402823e+38	0.0
	LREAL	8 bytes	-1.79769313486231e+308 ~ -2.22507385850721e-308, 0, 2.22507385850721e-308 ~ 1.79769313486231e+308,	0.0
Time, date	TIME	8 bytes	Display format: T#XXXXXXdXXhXXmXXsXXX.XXXms Range: T#0ms~213503d23h34m33s709.552ms Unit: Nanosecond (ns)	T#0ms
	DATE	4 bytes	Display format: D# year-month-day Range: D#1970-01-01~D#2106-02-07 Unit: Second (s)	D#1970-01-01
	TOD	4 bytes	Display format: TOD# Hours: minute: second. millisecond Range: TOD#0: 0: 0.000~23: 59: 59.999 Unit: Milliseconds (ms) When the value is 0, the corresponding value is TOD#0: 0: 0.000. When the value is 1000, the corresponding value is TOD#0: 0: 1.000.	TOD#0: 0: 0.000
	DT	4 bytes	Display format: DT#year-month-day-hour-minute-second Range: DT#1970-01-01-00: 00: 00~2106-02-07-06: 28: 15 Unit: Second (s)	DT#1970-01-01-00: 00: 00
String	STRING	0~80 bytes	0~80 characters	‘’

Chapter 2 Function description

2.1 Sequence input and output instructions

2.1.1 R_TRIG (rising edge detection)

When this instruction detects the rising edge of the input, the output signal is TRUE for one task period.

Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
R_TRIG	Rising edge triggering	FB	R_TRIG_Instance R_TRIG Clk Q	R_TRIG_Instance (Clk:= Parameter , Q => Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Clk	Input detection	Input	Input signal	FALSE or TRUE
Q	Output status	Output	Output signal	FALSE or TRUE

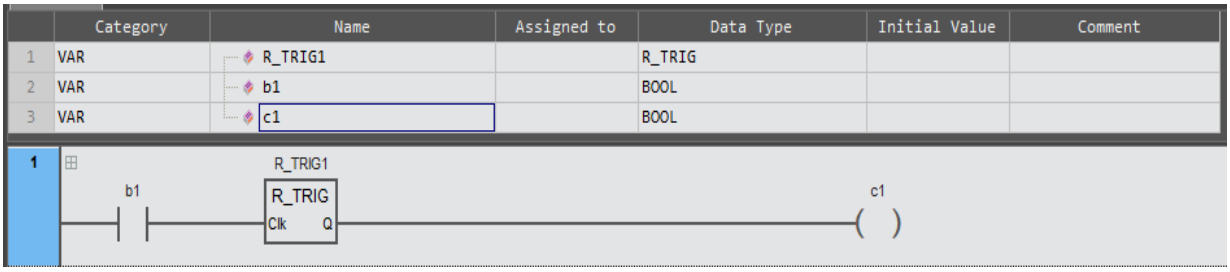
	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Clk	○																			
Q	○																			

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- When this instruction detects the rising edge of the input variable 'Clk' (changes from FALSE to TRUE), the output variable 'Q' becomes TRUE within one task period, FALSE over one task period, and FALSE in other cases.
- The example program is shown below.

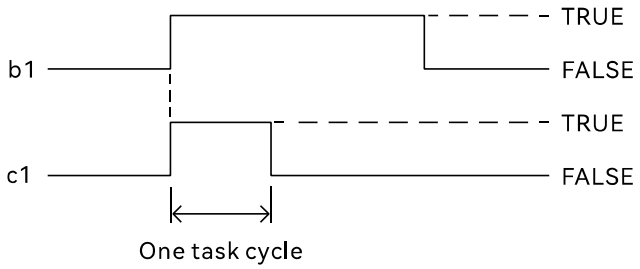
LD:



ST:

```
R_TRIG1(Clk: = b1 , Q=> c1 );
```

- The timing diagram is shown below.



2.1.2 F_TRIG (falling edge detection)

When this instruction detects the falling edge of the input, the output signal is TRUE for one task period.

Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
F_TRIG	Falling edge detection	FB	F_TRIG_Instance F_TRIG Clk Q	F_TRIG_Instance (Clk:= Parameter , Q => Parameter);

■ **Input / Output variable instructions and data types**

Name	Meaning	I/O	Description	Parameter scope
Clk	Input detection	Input	Input signal	FALSE or TRUE
Q	Output status	Output	Output signal	FALSE or TRUE

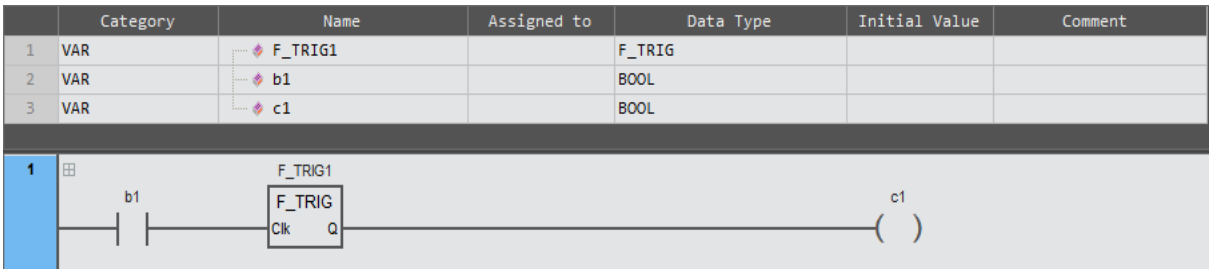
	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Clk	○																			
Q	○																			

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ **Function description**

- When this instruction detects the falling edge of the input variable 'Clk' (changes from TRUE to FALSE), the output variable 'Q' becomes TRUE within one task period, FALSE over one task period, and FALSE in any other cases.
- The example program is shown below.

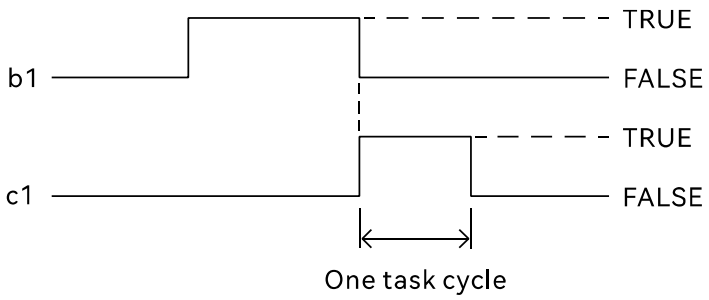
LD:



ST:

```
F_TRIG1(Clk: = b1 , Q=> c1 );
```

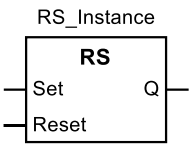
- The timing diagram is shown below.



2.1.3 RS (Reset before Set)

When the Reset input signal is TRUE, the Reset input signal has priority and the output signal is FALSE.

Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
RS	Reset before Set	FB		RS_ Instance (Set:= Parameter, Reset := Parameter, Q=> Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Set	Set input	Input	Set input	FALSE or TRUE
Reset	Reset input		Reset input	FALSE or TRUE
Q	Output	Output	Output	FALSE or TRUE

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Set	○																			
Reset	○																			
Q	○																			

*** Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

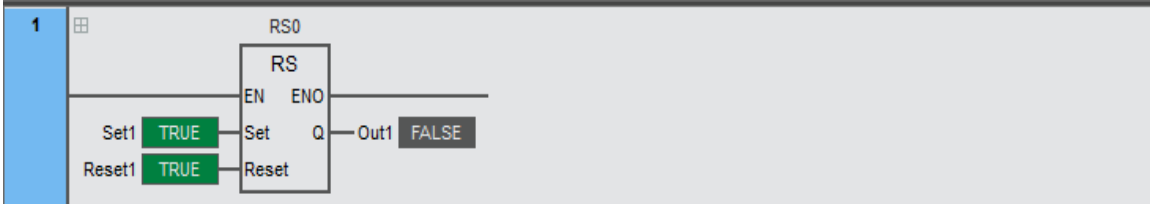
- This instruction is used for the Set or Reset operation, where the Reset operation has priority over the Set operation, i.e., when the Reset operation is executed, the Set operation is invalid.
- If 'Set' is TRUE, the Set operation is executed (set 'Q' to TRUE), and if 'Reset' is TRUE, the Reset operation is executed (set 'Q' to FALSE).
- When both Set and Reset are TRUE, the Reset operation takes priority and the output 'Q' is FALSE; When both Set and Reset are FALSE, the value of output 'Q' remains unchanged.
- The input and output logic relationships are shown in the table below.

'Set'	'Reset'	'Q'
FALSE	TRUE	FALSE
TRUE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	FALSE	Remain unchanged

- The example program and the timing diagram are shown below.

LD:

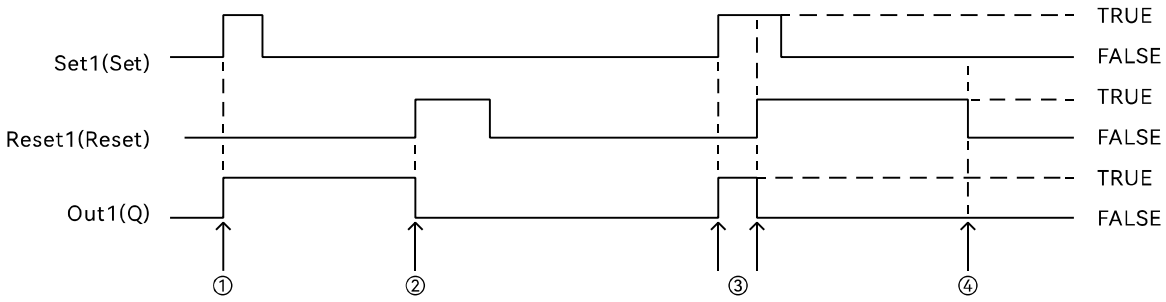
	Category	Name	Assigned to	Data Type	Online Value
1	VAR	RS0		RS	
3	VAR	Set1		BOOL	TRUE
4	VAR	Reset1		BOOL	TRUE
5	VAR	Out1		BOOL	FALSE



ST:

RS0(Set: = Set1 , Reset: = Reset1 , Q=> Out1);

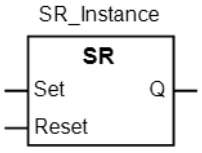
Timing diagram:



① If Set1 is TRUE and Reset1 is FALSE, then Out1 will become TRUE.
② If Set1 is FALSE and Reset1 is TRUE, then Out1 will become FALSE.
③ If Set1 and Reset1 are both TRUE, then Reset1 will have higher priority and Out1 will become FALSE.
④ If Set1 is FALSE and Reset1 becomes FALSE, then Out1 will remain unchanged.

2.1.4 SR (Set before Reset)

When the Set input signal is TRUE, the Set input signal has priority and the output signal is TRUE. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
SR	Set before Reset	FB		SR_Instance (Set:= Parameter , Reset :=Parameter , Q =>Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Set	Set input	Input	Set input	FALSE or TRUE
Reset	Reset input		Reset input	FALSE or TRUE
Q	Output	Output	Output	FALSE or TRUE

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Set	○																			
Reset	○																			
Q	○																			

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

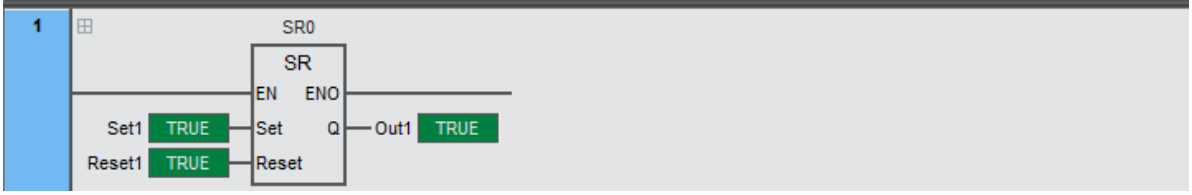
- This instruction is used for the Set or Reset operation, where the Reset operation has priority over the Set operation, i.e., when the Set operation is executed, the Reset operation is invalid.
- If 'Set' is TRUE, the Set operation is executed (set 'Q' to TRUE), and if 'Reset' is TRUE, the Reset operation is executed (set 'Q' to FALSE).
- When both Set and Reset are TRUE, the Set operation takes priority and the output 'Q' is TRUE; When both Set and Reset are FALSE, the value of output 'Q' remains unchanged.
- The input and output logic relationships are shown in the table below.

'Set'	'Reset'	'Q'
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE
FALSE	FALSE	Remain unchanged

- The example program and the timing diagram are shown below.

LD:

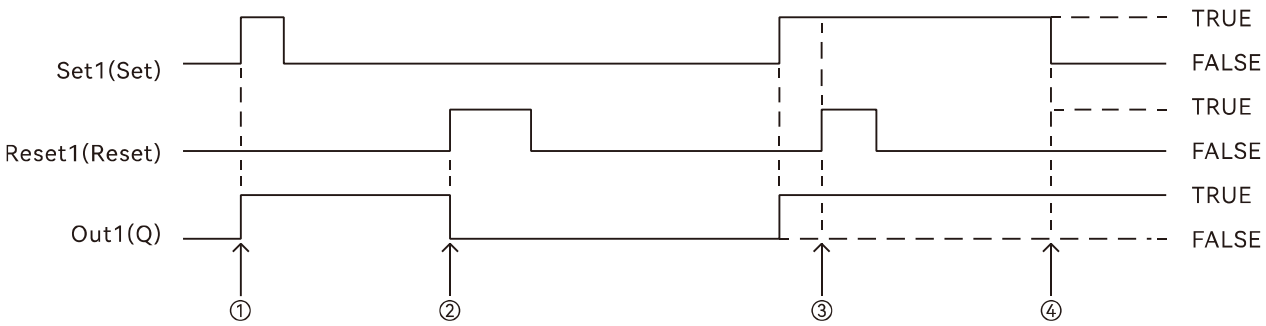
	Category	Name	Assigned to	Data Type	Online Value
1	VAR	SR0		SR	
3	VAR	Set1		BOOL	TRUE
4	VAR	Reset1		BOOL	TRUE
5	VAR	Out1		BOOL	TRUE



ST:

SR0(Set: = Set1 , Reset: = Reset1 , Q=> Out1);

Timing diagram:



① If Set1 is TRUE and Reset1 is FALSE, then Out1 will become TRUE.
② If Set1 is FALSE and Reset1 is TRUE, then Out1 will become FALSE.
③ If Set1 and Reset1 are both TRUE, then the SET1 will have higher priority and Out1 will remain to be TRUE.
④ If Reset1 is FALSE and Set1 becomes FALSE, then Out1 will remain unchanged.

2.1.5 SEMA (Set output delay)

When the Set input is TRUE, the Set input takes priority and the output signal is delayed by one period and becomes TRUE. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
SEMA	Set output delay	FB	SEMA_Instance SEMA Claim Q Release	SEMA _Instance (Claim :=Parameter , Release :=Parameter , Q =>Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Claim	Set input	Input	Set input	FALSE or TRUE
Release	Reset input		Reset input	FALSE or TRUE
Q	Output	Output	Output signal	FALSE or TRUE

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Claim	○																			
Release	○																			
Q	○																			

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

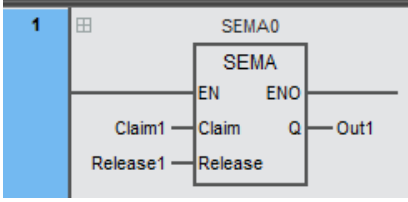
- This instruction is used for the Set or Reset operation, where the Reset operation has priority over the Set operation, i.e., when the Set operation is executed, the Reset operation is invalid.
- If 'Claim' is TRUE, the Set operation is executed (set 'Q' to TRUE), and if 'Release' is TRUE, the Reset operation is executed (set 'Q' to FALSE).
- When both Set and Reset are TRUE, the Set operation takes priority and the output 'Q' is delayed by one period and becomes TRUE; When both Set and Reset are FALSE, the value of output 'Q' remains unchanged.
- The input and output logic relationships are shown in the table below.

'Claim'	'Release'	'Q'
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE
FALSE	FALSE	Remain unchanged

- The example program and the timing diagram are shown below.

LD:

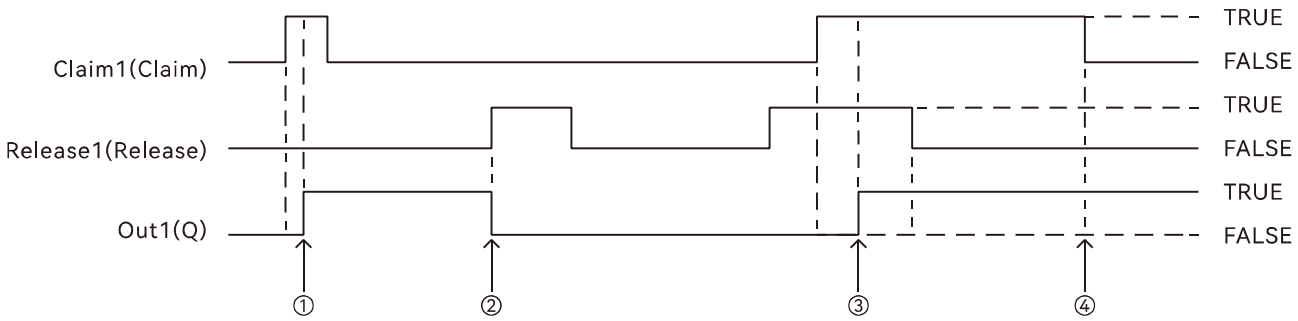
	Category	Name	Assigned to	Data Type	Initial Value
1	VAR	SEMA0		SEMA	
2	VAR	Claim1		BOOL	
3	VAR	Release1		BOOL	
4	VAR	Out1		BOOL	



ST:

SEMA0(Claim: = Claim1 , Release: = Release1 , Q=> out1);

Timing diagram:



① If Release1 is FALSE and Claim1 becomes TRUE, then Out1 will delay for one task cycle and become TRUE.
② If Claim1 is FALSE and Release1 becomes TRUE, then Qut1 will become FALSE.
③ If Release1 is TRUE and Claim1 becomes TRUE, then Out1 will delay for one task cycle and become TRUE.
④ If Release1 is FALSE and Claim1 becomes FALSE, then Out1 will remain unchanged.

2.1.6 JMP (jump)

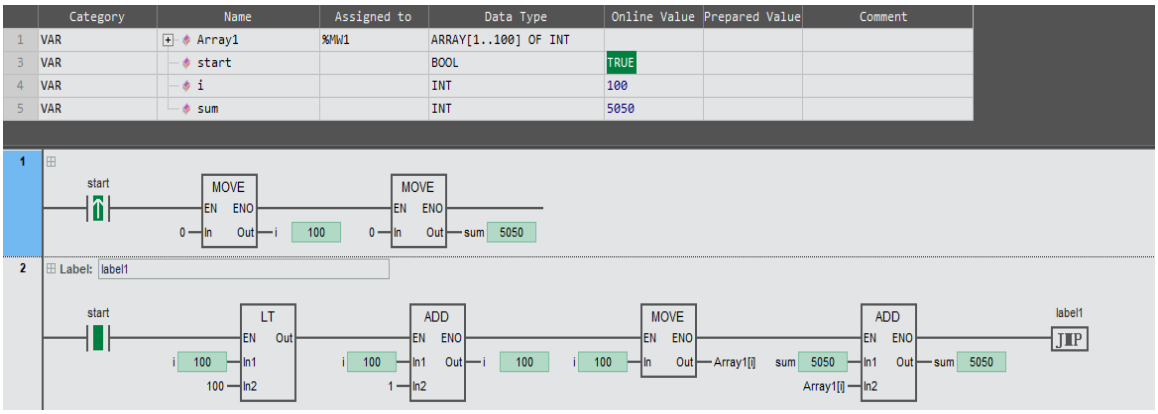
Moves processing to the specified jump destination. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
JMP	Jump	FUN	label JMP	GOTO Label ;

■ Function description

- When the input condition is satisfied, the ' JMP ' instruction takes effect and jumps to the specified destination for execution. Otherwise, the instructions after ' JMP ' will be executed.
- Functional description of the example program: Network 1 assigns the value of i and sum to 0 by the rising edge of the start variable. Network 2 realizes the cycle of 100 times in one period by the cooperation of the ' JMP ' instruction and the LT instruction, and puts the 1~100 correspondingly into the device of %MW1~%MW100, and seeks for the cumulative sum of the 1~100.
- The example program is shown below.

LD:



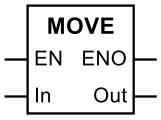
ST:

```
IF start THEN
    label1:
        IF i<100 THEN
            i: =i+1;
            Array1[i]: =i;
            sum: =sum + Array1[i];
            GOTO label1;
        END_IF;
END_IF;
```

2.2 Data movement instructions

2.2.1 MOVE (movement instructions)

This instruction is used to move the value of a single variable or constant within another variable. Library: Standard.

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
MOVE	Movement instructions	FUN		Out: = In ;

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data source	Input	Data source	Depends on variable type
Out	Move destination	Output	Data output	Depends on variable type

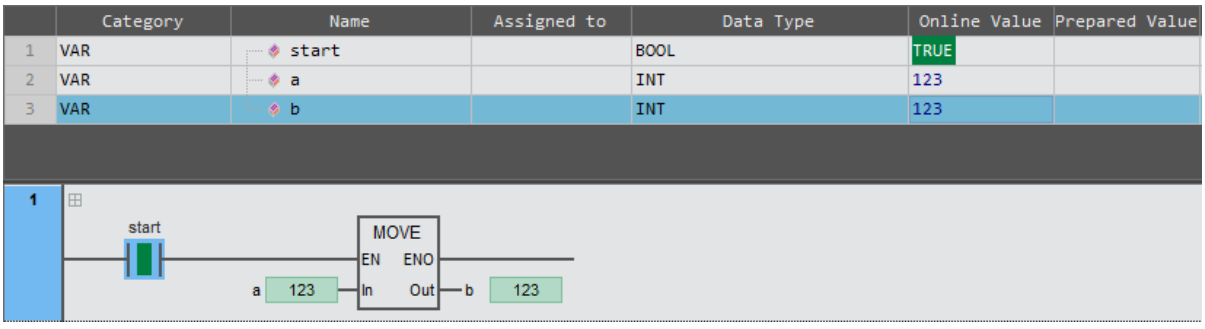
	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Out		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- This instruction is used to move the value of a variable or constant from the data source 'In' to 'Out'.
- The data source 'In' can be an enumeration, an array variable name, an array member, a structure, or a member of a structure.
- The data types of 'In' and 'Out' shall be the same. When 'In' and 'Out' are the names of array variables, the data of all members of the 'In' array can be moved to all members of the 'Out' array, and the data type and the number of members of the array need to be the same.
- The example program1: Move the value of variable an into variable b.

LD:



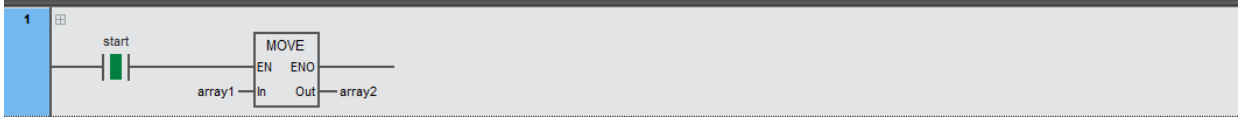
ST:

b: = a;

- The example program2: Moves the values of all members of the array variable array1 to those of the array variable array2.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value	Comment
1	VAR	start		BOOL	TRUE		
2	VAR	array1		ARRAY[1..5] OF INT			
4	VAR	array2		ARRAY[1..5] OF INT			
5	VAR	array2[1]		INT	1		
6	VAR	array2[2]		INT	2		
7	VAR	array2[3]		INT	3		
8	VAR	array2[4]		INT	4		
9	VAR	array2[5]		INT	5		

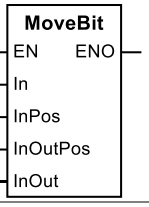


ST:

```
IF EDPOSE(start) THEN
        array2: =array1;
END_IF ;
```

2.2.2 MoveBit (bit movement instructions)

This instruction is used to move the value of one bit in one variable to one bit in another variable. Library: Standard.

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
MoveBit	Bit movement instructions	FUN		MoveBit(In , InPos , InOutPos , InOut);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data source	Input	Data source	Depends on variable type
InPos	Move source bit		Move source bit	0 to the number of bits in In -1
InOutPos	Move destination bit		Move destination bit	0 to the number of bits in InOut -1
InOut	Move destination		Move destination	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○											
InPos							○													
InOutPos							○													
InOut		○	○	○	○	○	○	○	○											

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

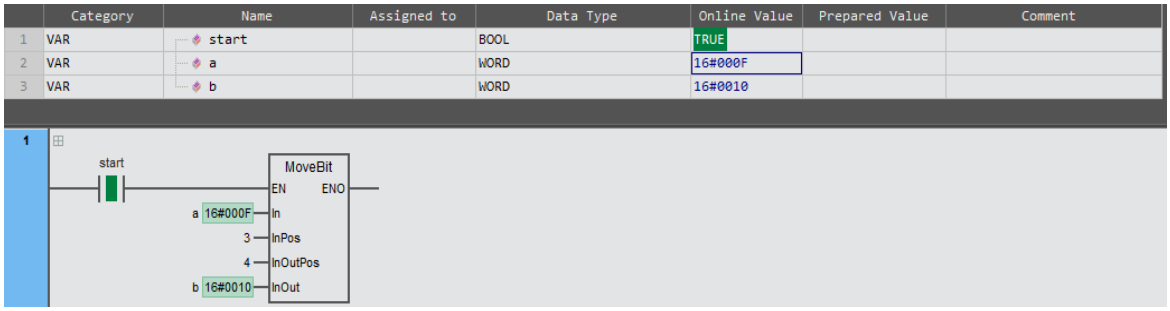
■ Function description

- This instruction is used to move the value of one bit in the specified position of the input variable to one bit in the specified position of the 'InOut' variable, while the values of other bits in the 'InOut' variable remain unchanged.
- 'InPos' indicates the position of the specified bit of 'In' in the data source, and the value of 'InPos' is calculated from bit 0 of 'In'. If the value of 'InPos' is 0, it means that bit 0 of 'In' is specified, and if the value of 'InPos' is 1, it means that bit 1 of 'In' is specified, and so on.
- 'InOutPos' indicates the position of the specified bit of 'InOut' in the move destination, and the value of 'InOutPos' is calculated from bit 0 of 'InOut'. If the value of 'InOutPos' is 0, it means that bit 0 of 'InOut' is specified, and if the value of 'InOutPos' is 1, it means that bit 1 of 'InOut' is specified, and so on.
- Example program description: Bit moves two WORD type variables, the value of variable a is 16#0F, the value of 'InPos' is 3, the value of 'InOutPos' is 4, after the execution of this instruction, the value of variable b is 16#10.

⚠ Precautions

- If the position specified by ' InPos ' exceeds the position range of ' In ' , bit movement will not be executed and the value of ' InOut ' will remain unchanged.
- If the position specified by ' InOutPos ' exceeds the position range of ' InOut ' , bit movement will not be executed and the value of ' InOut ' will remain unchanged.
- The example program and the timing diagram are shown below.

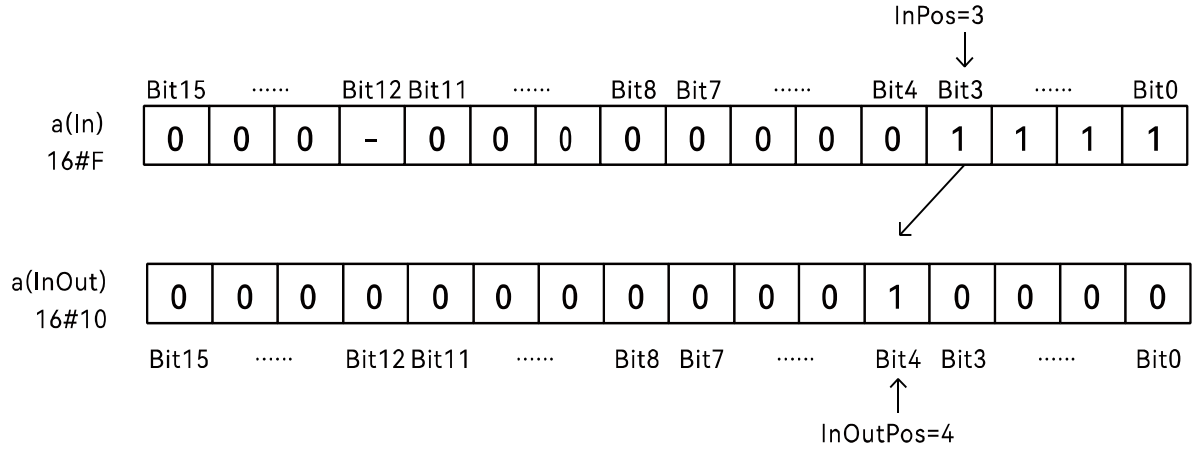
LD:



ST:

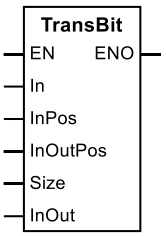
```
MoveBit(a , 3 , 4 , b );
```

Execution results diagram:



2.2.3 TransBit (multi-bit movement instructions)

This instruction is used to move the value of multiple bits at the specified position in an input variable to multiple bits at the specified position in another variable. Library: Standard.

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
TransBit	Multi-Bit movement instructions	FUN		TransBit (In , InPos , InOutPos , Size , InOut);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data source	Input	Data source	Depends on variable type
InPos	Move source bit		Move source bit	0 to the number of bits in In -1
InOutPos	Move destination bit		Move destination bit	0 to the number of bits in InOut -1
Size	Number of bits		Number of moves	0 to the number of bits in In
InOut	Move destination		Move destination	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○											
InPos							○													
InOutPos							○													
Size							○													
InOut		○	○	○	○	○	○	○	○											

*Note: The ' ○ ' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- This instruction is used to move the value of multiple bits in the specified position in the input variable to multiple bits in the specified position in the InOut variable, while the value of other bits in the InOut variable remains unchanged.
- ' InPos ' indicates the position of the specified bit of ' In ' in the move source, and the value of ' InPos ' is calculated from bit 0 of ' In ' . If the value of ' InPos ' is 0, it means that bit 0 of ' In ' is specified, and if the value of ' InPos ' is 1, it means that bit 1 of ' In ' is specified, and so on.
- ' InOutPos ' indicates the position of the specified bit of ' InOut ' in the move destination, and the value of ' InOutPos ' is calculated from bit 0 of ' InOut ' . If the value of ' InOutPos ' is 0, it means that bit 0 of ' InOut ' is specified, and if the value of ' InOutPos ' is 1, it means that bit 1 of ' InOut ' is specified, and so on.
- Size indicates the number of moves.

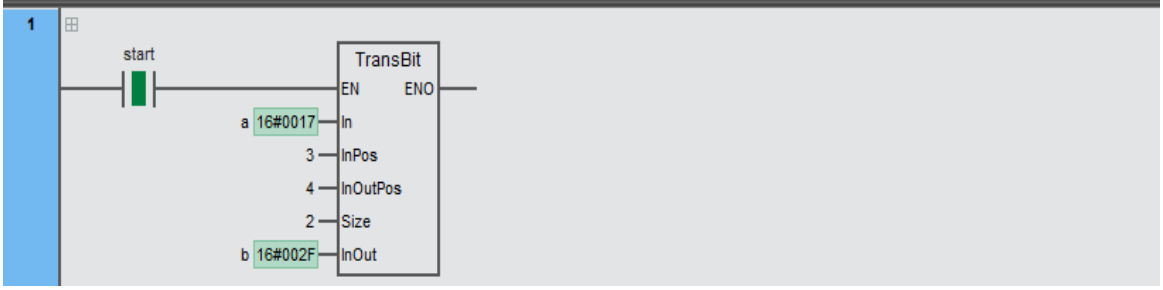
- Example program description: Bit moves two WORD type variables, the value of variable a is 16#0F, the value of 'InPos' is 3, the value of 'InOutPos' is 4, after the execution of this instruction, the value of variable b is 16#10. Multi-bit moves two WORD type variables, 'In' is 16#17, 'InPos' is 3, 'InOutPos' is 4, and size value is 2, after the execution of the instruction, the value of b is 16#2F.

⚠ Precautions

- If the position specified by 'InPos' exceeds the position range of 'In', bit movement will not be executed and the value of 'InOut' will remain unchanged.
- If the position specified by 'InOutPos' exceeds the position range of 'InOut', bit movement will not be executed and the value of 'InOut' will remain unchanged.
- If the value of 'Size' exceeds the range of the 'In' or 'InOut' variable, bit movement will not be executed and the value of 'InOut' will remain unchanged.
- The example program and the timing diagram are shown below.

LD:

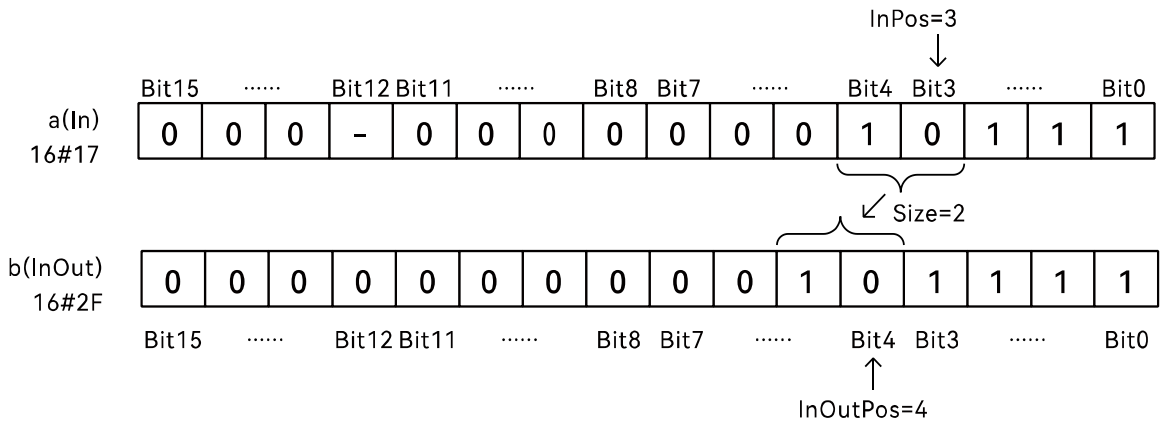
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	start		BOOL	TRUE	
2	VAR	a		WORD	16#0017	
3	VAR	b		WORD	16#002F	



ST:

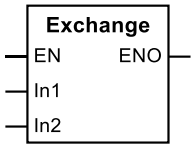
```
TransBit( a , 3 ,  4  , 2 , b );
```

Execution results diagram:



2.2.4 Exchange (data exchange instructions)

This instruction is used to exchange the values of two variables. Library: Standard.

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
Exchange	Data exchange	FUN		Exchange(In1 , In2);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Input1	Input	Data to exchange 1	Depends on variable type
In2	Input2		Data to exchange 2	Depends on variable type

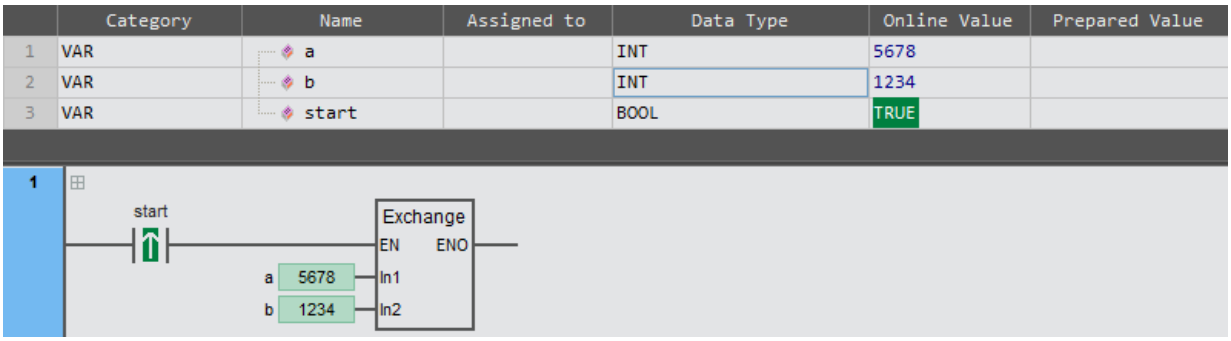
	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
In2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- This instruction is used to exchange the values of two variables.
- The data types of the two variables shall be the same.
- When this instruction is executed, it is recommended that the execution condition of this instruction is a rising edge. If the execution condition of this instruction is always TRUE, the values of the two variables will always be exchanged.
- The example program is shown below.

LD:

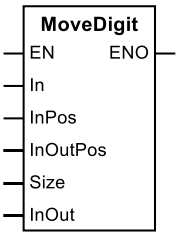


ST:

```
Exchange( a , b );
```

2.2.5 MoveDigit (digit movement instructions)

This instruction is used to move the value of a number of digits at a specified position and number of digits in one variable to a number of digits starting at a specified position in another variable. Library: Standard.

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
MoveDigit	Digit movement instructions	FUN		MoveDigit (In , InPos , InOutPos , Size , InOut);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data source	Input	Data source	Depends on variable type
InPos	Move source bit		Move source bit	0 to the number of bits in In -1
InOutPos	Move destination bit		Move destination bit	0 to the number of bits in Out -1
Size	Number of digits		Number of digits to move (four digits/move)	0 to the number of bits in In
InOut	Move destination		Move destination	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○											
InPos							○													
InOutPos							○													
Size							○													
InOut		○	○	○	○	○	○	○	○											

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- The number of digits in the specified position and number of digits in the input variable is moved to the number of digits starting from the specified position in the 'InOut' variable, and the four digits in the 'In' variable and the 'InOut' variable form a single digit.
- 'InPos' Indicates the position of the input variable digit. Digit 0 is composed of bit 0 ~ bit 3 of the input variable, corresponding to the value of 'InPos' is 0; Digit 1 is composed of bit 4 ~ bit 7 of the input variable, corresponding to the value of 'InPos' is 1, and so on. 'InOutPos' indicates the position of the number of digits of the 'InOut' variable. Digit 0 is composed of bit 0 to bit 3 of the 'InOut' variable, and the corresponding value of 'InOutPos' is 0; Digit 4 is composed of bit 1 to bit 7 of the 'InOut' variable, the corresponding value of 'InOut' is 1, and so on.
- Size indicates the number of digits.

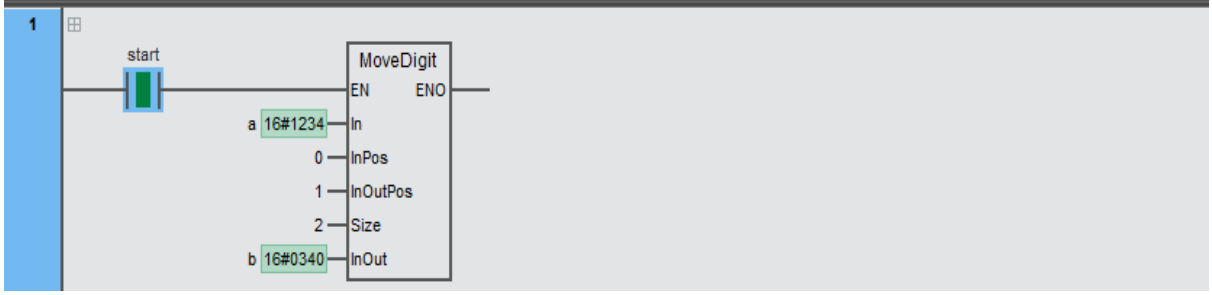
- Example program description: Move the digit of two variables of WORD type. The value of 'In' is 16#1234, the value of 'InPos' is 0, the value of 'InOutPos' is 1, the 'Size' value is 2, and the initial value of b is 0. After this instruction is executed, the value of b is 16#340.

⚠ Precautions

- If the position specified by 'InPos' exceeds the position range of 'In', digit movement will not be executed and the value of 'InOut' will remain unchanged.
- If the position specified by 'InOutPos' exceeds the position range of 'InOut', digit movement will not be executed and the value of 'InOut' will remain unchanged.
- If the value of 'Size' exceeds the range of the 'In' or 'InOut' variable, digit movement will not be executed and the value of 'InOut' will remain unchanged.
- The example program and the timing diagram are shown below.

LD:

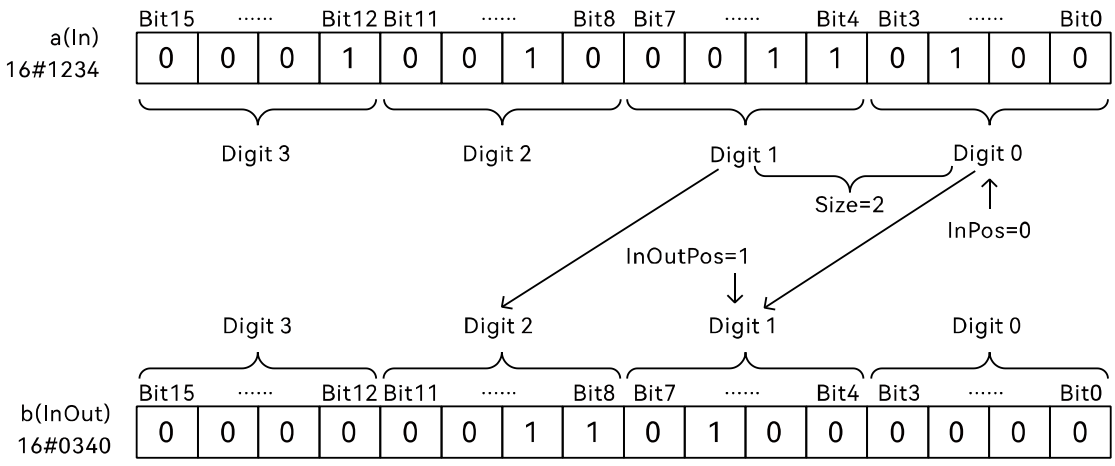
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	start		BOOL	TRUE	
2	VAR	a		WORD	16#1234	
3	VAR	b		WORD	16#0340	



ST:

```
MoveDigit(a , 0 , 1 , 2 , b );
```

Execution results diagram:



2.2.6 Swap (high and low bit data swap instructions)

This instruction is used to swap the high and low bytes of the value in the input variable and put it into the output variable. Library: Standard .

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
Swap	Swap Bytes	FUN	Swap EN ENO In Out	Out: =Swap(In);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data source	Input	Data source	Depends on variable type
Out	Swap result	Output	Swap result	Depends on variable type

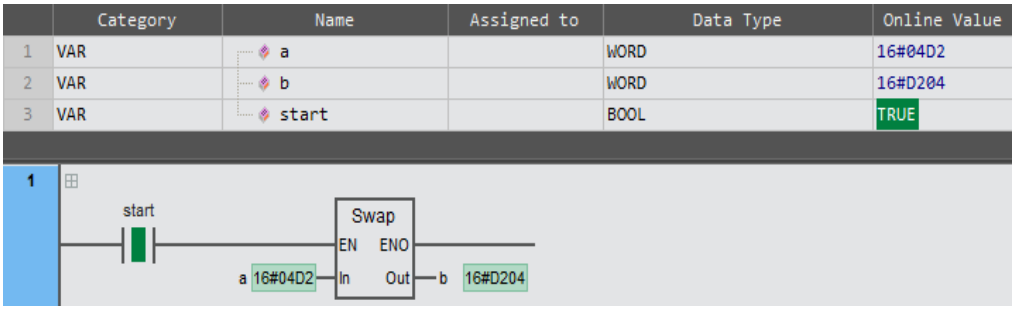
	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In			○						○											
Out			○						○											

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- This instruction is used to input the value in the 'In' variable, swap the high and low bytes, and then put it into the output 'Out' variable.
- The data types of the two variables shall be the same.
- When this instruction is executed, it is recommended that the execution condition of this instruction is a rising edge. If the execution condition of this instruction is always TRUE, the values of the two variables will always be exchanged. If the execution condition of this instruction is always TRUE, the value of the 'In1' variable will be put into the 'Out' variable after swapping the high and low bytes.
- The example program is shown below.

LD:



ST:

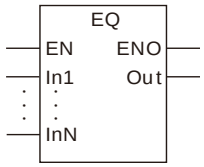
```
b: =Swap( a );
```

2.3 Comparison operations

2.3.1 EQ(=)

The instruction compares two or more variables/constants for equality. The comparison result is ' Out '.

Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
EQ	Equal	FUN		Out: =(In1=In2)&(In2= In3)&(InN-1= InN) ;

■ **Input / Output variable instructions and data types**

Name	Meaning	I/O	Description	Parameter scope
In1	Comparison data	Input	Comparison data	Depends on data type
In2~InN	Comparison data		Comparison data. LD: PRG can increase or decrease the number of operation objects， Values to compare N = 2 ~ 8	Depends on data type
Out	Comparison result	Output	Comparison result	FALSE or TRUE

	Boolean	Bit string				Integer								Real number		Time ,date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
In2~InN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Out	○																			

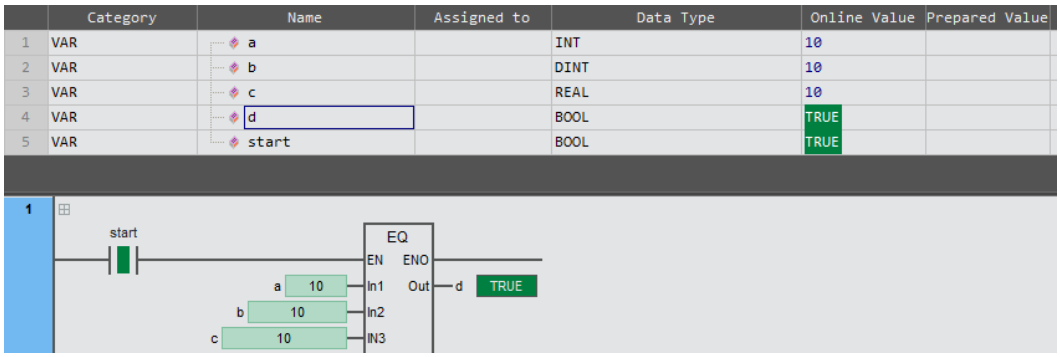
***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- The instruction compares 'In1' ~ 'InN' variables for equality. The comparison result is 'Out'. For example, Out : =(In1=In2)&(In2= In3)&(InN-1= InN).The comparison result 'Out' is 'TRUE' only when all values are equal. Otherwise, the value of 'Out' is 'FALSE'.
- When the input variable types are bit strings, integers and real numbers, the input variables 'In1' ~ 'In2' should be different data types. If the data types of 'In1' ~ 'InN' are different, they will be expanded to a data type that includes the ranges of all of the data types. For example, 'In1' data type is INT, 'In2' date type is DINT, and the DINT data type is used for computational processing.
- If the data type of the input variable is one of BOOL, TIME, DATE, TOD, DT, and STRING, the input variables 'In1' ~ 'InN' must be of this data type. For example, 'In1' data type is DATE, and the input variables 'In2' ~ 'InN' must be DATE, otherwise a building error will occur.

- The example program is shown below

LD:

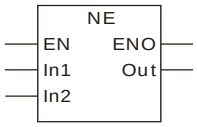


ST:

d: =(a = b) & (b = c);

2.3.2 NE (< >)

The instruction compares two variables/constants for inequality. The comparison result is 'Out'. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
NE	Not equal	FUN		Out: =(In1 <> In2);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Comparison data	Input	Comparison data	Depends on data type
In2	Comparison data		Comparison data	Depends on data type
Out	Comparison result	Output	Comparison result	FALSE or TRUE

	Boolean	Bit string				Integer								Real number		Time ,date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
In2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Out	○																			

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

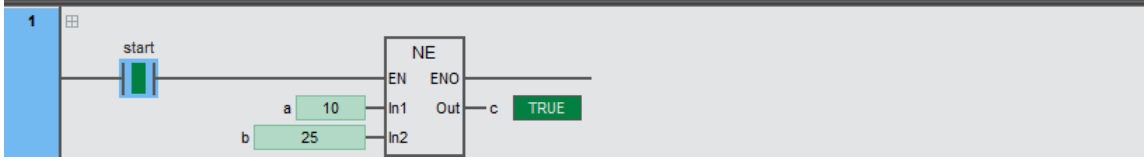
■ Function description

- The instruction compares two variables for inequality. The comparison result is 'Out', that is, Out: =In1<>In2. If they are not equal, the comparison result 'Out' is 'TRUE'. If they are equal, it is 'FALSE'.
- When the input variable types are bit strings, integers and real numbers, the input variables 'In1' ~ 'In2' should be different data types. If the data types of 'In1' ~ 'In2' are different, they will be expanded to a data type that includes the ranges of all of the data types. For example, 'In1' date type is INT, 'In2' date type is DINT, and the DINT data type is used for computational processing.
- If the data type of the input variable is one of BOOL, TIME, DATE, TOD, DT, and STRING, the input variables 'In1' ~ 'In2' must be of these data types. For example, 'In1' date type is TIME, and the input variables 'In2' ~ 'InN' must be TIME, otherwise a building error will occur.

- The example program is shown below

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		INT	10	
2	VAR	b		DINT	25	
3	VAR	c		BOOL	TRUE	
4	VAR	start		BOOL	TRUE	

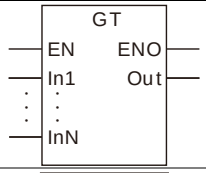
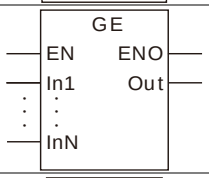
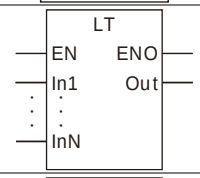
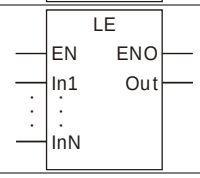


ST:

```
c: =a <> b;
```

2.3.3 GT (>), GE (=>), LT (<), and LE (<=)

These instructions compare two or more variables/constants. The comparison result is 'Out'. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
GT	Greater than	FUN		Out: =(In1 > In2)&(In2 > In3)&(InN-1 > InN) ;
GE	Greater than or equal to	FUN		Out: =(In1=>In2)&(In2 => In3)&(InN-1 => InN) ;
LT	Less than	FUN		Out: =(In1 < In2)&(In2 < In3)&(InN-1 < InN);
LE	Less than or Equal to	FUN		Out: =(In1<=In2)&(In2<= In3)&(InN-1<= InN);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Comparison data	Input	Comparison data	Depends on data type
In2~InN	Comparison data		Comparison data LD: PRG can increase or decrease the number of operation objects，Values to compare N = 2 ~ 8	Depends on data type
Out	Comparison result	Output	Comparison result	Depends on data type

	Boolean	Bit string				Integer								Real number		Time ,date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
In2~InN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Out	○																			

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

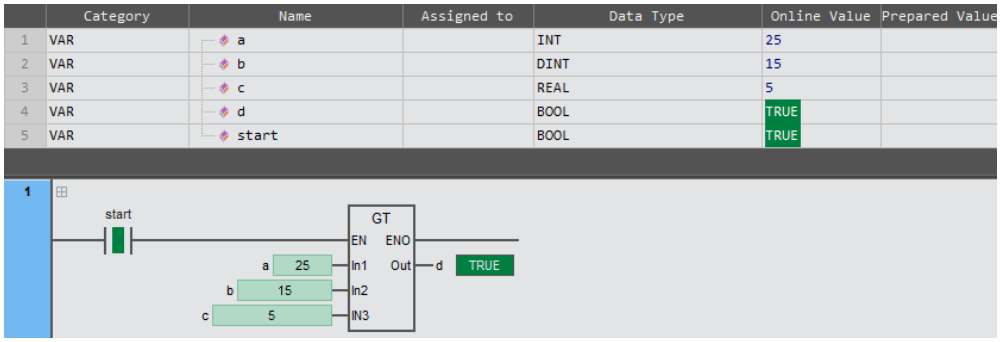
■ Function description

- These instructions compare two or more variables or constants. The comparison result is ' Out '. The output value 'Out' is shown below for each instruction

Instruction	' Q '
GT (>)	If In1 > In2 > ... > InN, Out is TRUE. Otherwise, it is FALSE
GE (=>)	If In1 => In2 => ... => InN, Out is TRUE. Otherwise, it is FALSE
LT (<)	If In1 < In2 < ... < InN, Out is TRUE. Otherwise, it is FALSE
LE (<=)	If In1 <= In2 <= ... <= InN, Out is TRUE. Otherwise, it is FALSE

- When the input variable types are bit strings, integers and real numbers, the input variables 'In1' ~ 'In2' should be different data types. If the data types of 'In1' ~ 'In2' are different, they will be expanded to a data type that includes the ranges of all of the data types. For example, 'In1' date type is INT, 'In2' date type is DINT, and the DINT data type is used for computational processing.
- If the data type of the input variable is one of BOOL, TIME, DATE, TOD, DT, and STRING, the input variables 'In1' ~ 'In2' must be of this data type. For example, 'In1' date type is TIME, and the input variables 'In2' ~ 'InN' must be TIME, otherwise a building error will occur.
- The GT example program is shown below.

LD:

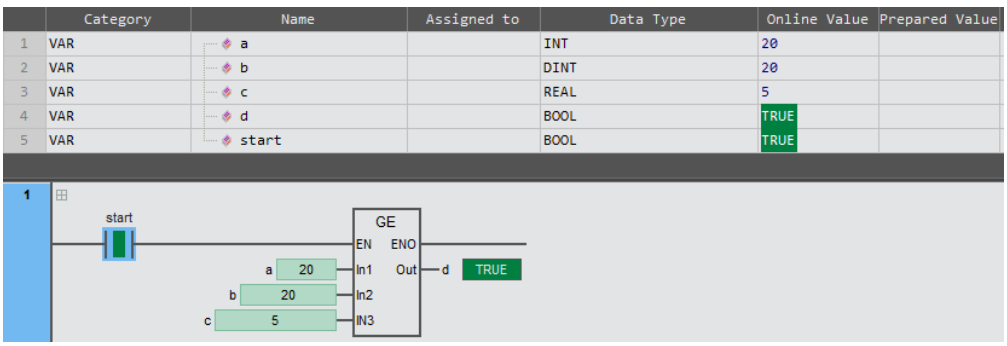


ST:

```
d: =( a > b )&( b > c );
```

- The GE example program is shown below.

LD:



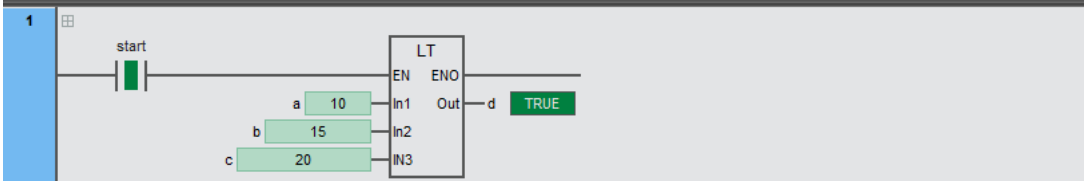
ST:

```
d: =( a => b )&( b> = c);
```

- The LT example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		INT	10	
2	VAR	b		DINT	15	
3	VAR	c		REAL	20	
4	VAR	d		BOOL	TRUE	
5	VAR	start		BOOL	TRUE	



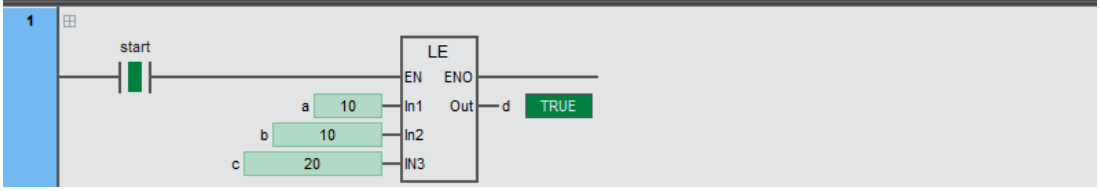
ST:

d: =(a < b)&(b < c);

- The LE example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		INT	10	
2	VAR	b		DINT	10	
3	VAR	c		REAL	20	
4	VAR	d		BOOL	TRUE	
5	VAR	start		BOOL	TRUE	



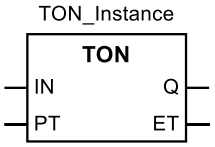
ST:

d: =(a <= b)&(b <=c);

2.4 Timer

2.4.1 TON (On-Delay timer)

The TON instruction outputs ' TRUE ' when the set time elapses after the timer starts. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
TON	On-Delay timer	FB		TON_Instance (IN:= Parameter , PT:= Parameter, Q=> Parameter, ET=> Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
IN	Timer input	Input	TRUE : starts delay counter FALSE: resets delay counter	FALSE or TRUE
PT	Set time		The time span from the time timer starts to the time Q changes to TRUE	Refer to the *note
Q	Timer output	Output	TRUE: if IN is TRUE and delay time PT elapsed FALSE: if IN is FALSE	FALSE or TRUE
ET	Elapsed time		Elapsed time since rising edge at IN	Refer to the *note

***Note:** T#0ms ~ T#213503d23h34m33s709.552ms.

	Boolean	Bit strings				Integers								Real numbers		Times, dates				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	○																			
PT																○				
Out	○																			
ET																○				

***Note:** The ' ○ ' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

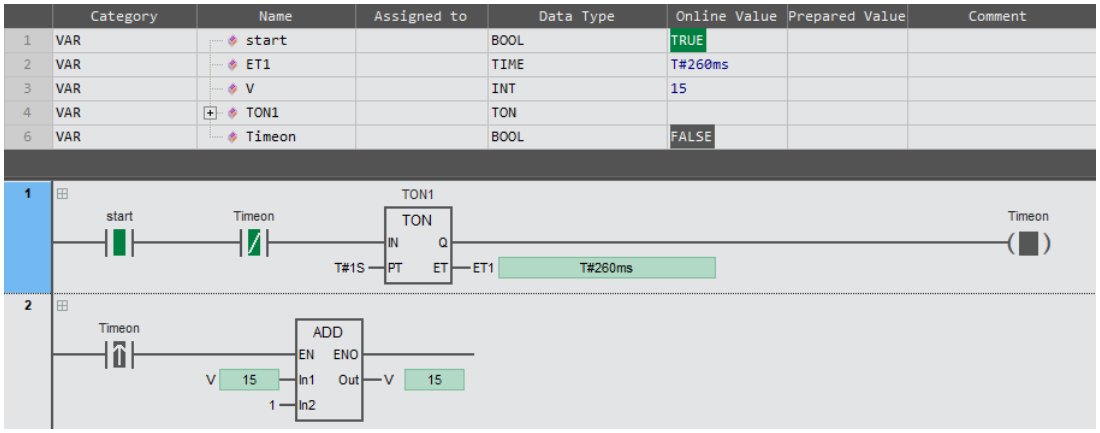
■ Function description

- The TON instruction outputs ' TRUE ' when the set time elapses after the timer starts. The minimum unit of timer setting time is nanoseconds (ns).
- The timer starts when timer input ' IN ' changes to ' TRUE '. Elapsed time ' ET ' is incremented as time elapses. When ' ET ' reaches the set time ' PT ', timer output ' Q ' changes to ' TRUE '. ' ET ' will not be incremented after that. The timer is reset when ' IN ' changes to ' FALSE '. ' ET ' changes to 0, and ' Q ' changes to ' FALSE '.
- If ' IN ' changes to ' FALSE ' after the timer is started, the timer is reset even before ' ET ' reaches ' PT '. ' ET ' changes to 0, and ' Q ' changes to ' FALSE '.

- The example program is shown below.

LD:

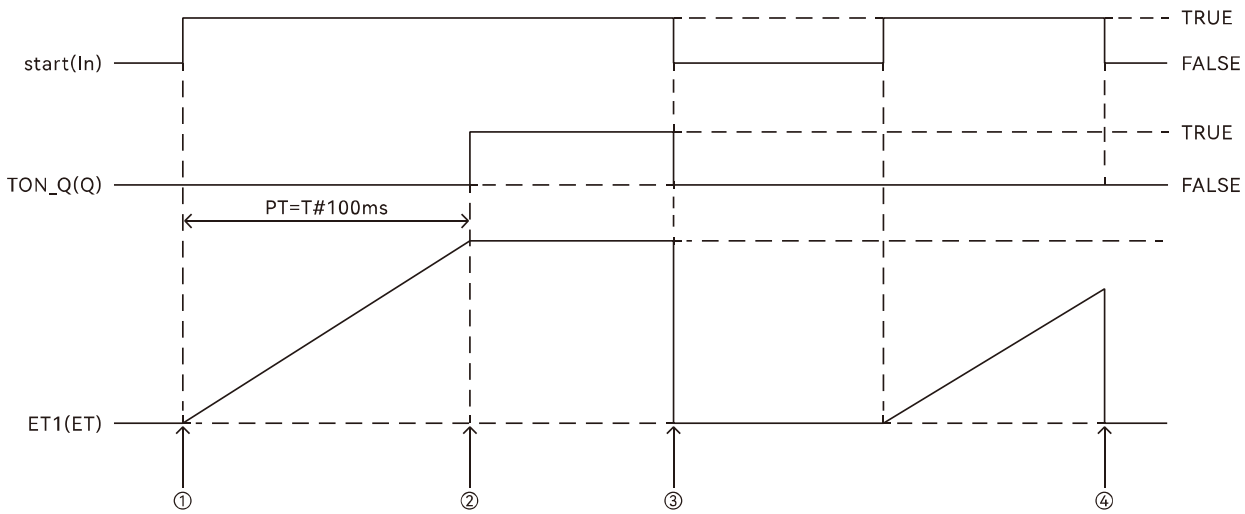
Triggers once in 1 second, and the variable is increased by 1. When Start is TRUE, the value of ' V ' is incremented by 1 at intervals of 1 second.



ST:

```
TON1 (IN: =start AND (NOT Timeon) ,PT: = T#1s ,Q=>Timeon ,ET=>ET1 );
IF   EDGEPOS(Timeon)   THEN;
      V: =V+1;
END_IF;
```

- The timing diagram is shown below.



① If the timer "In" becomes TRUE, then it will start timing.
② If the timer reaches the set time, then "Q" will become TRUE, and "ET" will remain unchanged.
③ If the timer reaches the set time and "In" becomes FALSE, then "Q" will become FALSE, and "ET" will be reset to 0.
④ If the timer does not reach the set time and "In" becomes FALSE, then "Q" will become FALSE.

2.4.2 TOF (Off-Delay timer)

The TOF instruction outputs FALSE when the set time elapses after the timer starts. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
TOF	Off-Delay timer	FB	TOF_Instance TOF IN Q PT ET	TOF_Instance (IN:= Parameter , PT:= Parameter, Q=> Parameter, ET=> Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
IN	Timer input	Input	TRUE : resets delay counter FALSE: starts delay counter	FALSE or TRUE
PT	Set time		Time from when timer starts until Q changes to FALSE	Refer to the *note
Q	Timer output	Output	TRUE: if IN is TRUE FALSE : if IN is FALSE and delay time PT elapsed	FALSE or TRUE
ET	Elapsed time		Elapsed time since falling edge at IN	Refer to the *note

***Note**: T#0ms ~ T#213503d23h34m33s709.552ms

	Boolean	Bit strings				Integers								Real numbers		Times ,dates				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	○																			
PT																○				
Q	○																			
ET																○				

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

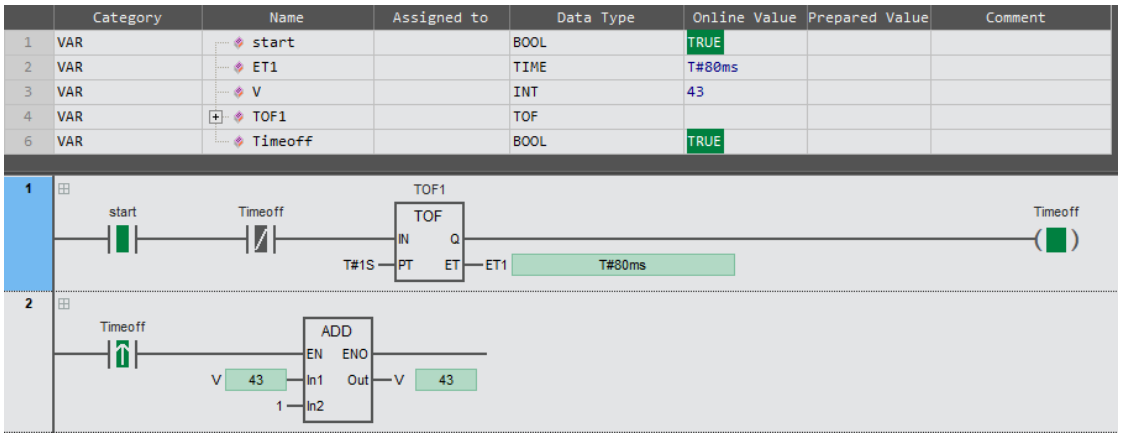
■ Function description

- The TOF instruction outputs 'FALSE' when the set time elapses after the timer starts. The minimum unit of timer setting time is nanoseconds (ns).
- The timer is reset when 'IN 'changes to 'TRUE '. 'ET ' changes to 0, and 'Q 'changes to 'TRUE'.
- The timer starts when timer input 'IN' changes to 'FALSE'. 'Q ' changes to 'TRUE '. Elapsed time 'ET' is incremented as time elapses. When 'ET' reaches the set time 'PT', timer output 'Q' changes to 'FALSE'. 'ET' is not incremented after that. The timer is reset when 'IN' changes to 'TRUE' . 'ET' changes to 0 , and 'Q' changes to 'TRUE'.
- If 'IN' changes to 'TRUE' after the timer is started, the timer is reset even before 'ET' reaches 'PT'. 'ET' changes to 0, and 'Q' still is 'TRUE'.

- The example program is shown below

LD:

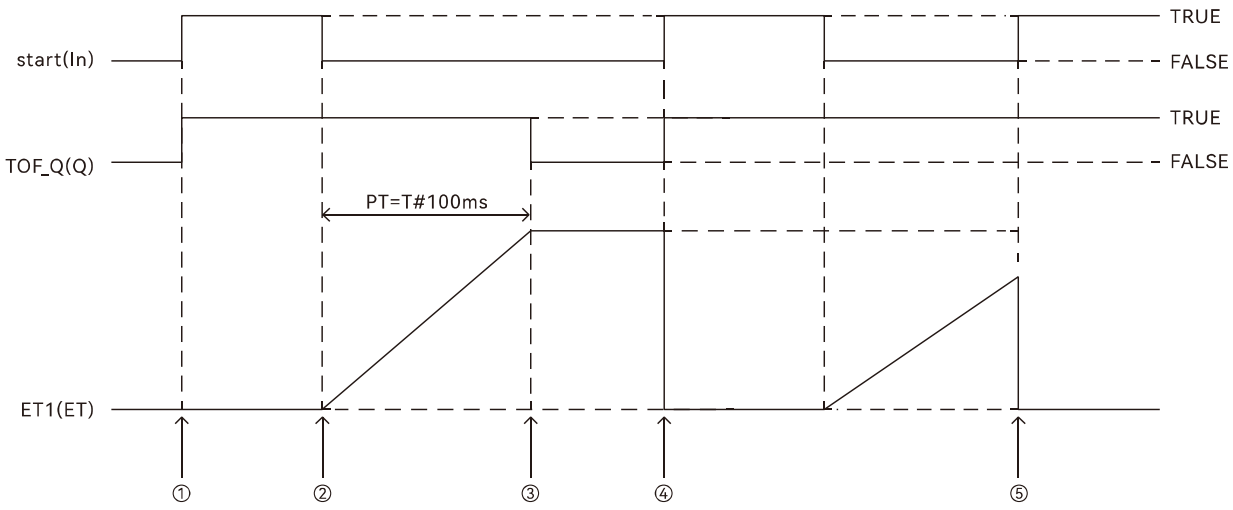
When ' Start ' is 'TRUE ', the value of 'V ' will be incremented by 1 at intervals of 1 second.



ST:

```
TOF1 (IN: =start AND (NOT Timeoff) ,PT: = T#1s ,Q=>Timeoff ,ET=>ET1 );
IF EDGEPOS(Timeoff) THEN;
    V: =V+1;
END_IF;
```

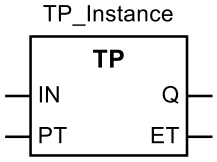
- The timing diagram is shown below



① If the timer "In" becomes TRUE, then "Q" will become TRUE, and "ET" will be reset to 0.
② If the timer "In" becomes FALSE, then "Q" will become TRUE, and the timer will start timing.
③ If the timer reaches the set time and "Q" becomes FALSE , "ET" will be reset to 0.
④ If the timer reaches the set time and the timer "In" becomes TRUE, then "Q" will become TRUE, and "ET" will be reset to 0.
⑤ If the timer does not reach the set time and "In" becomes TRUE, then "Q" will become TRUE, and "ET" will be reset to 0.

2.4.3 TP（timer pulse）

The TP instruction outputs TRUE for a set period of time after the timer starts. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
TP	Timer pulse	FB		TP_Instance (IN:= Parameter , PT:= Parameter, Q=> Parameter, ET=> Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
IN	Timer input	Input	TRUE : starts delay counter FALSE: resets delay counter	FALSE or TRUE
PT	Set time		Time during which ' Q ' remains at TRUE	Refer to the *note
Q	Timer output	Output	TRUE: Timer output ON FALSE: Timer output OFF	FALSE or TRUE
ET	Elapsed time		Elapsed time since rising edge at IN	Refer to the *note

***Note:** T#0ms ~ T#213503d23h34m33s709.552ms

	Boolean	Bit strings				Integers								Real numbers		Times ,dates				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	○																			
PT																○				
Out	○																			
ET																○				

***Note:** The ' ○ ' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

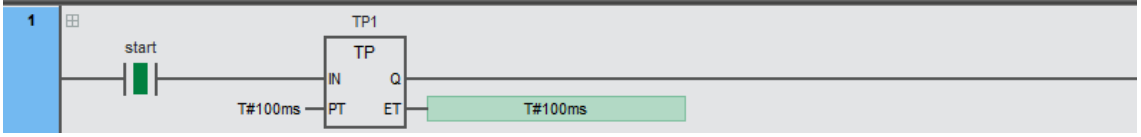
■ Function description

- The TP instruction outputs TRUE for a set time after the timer starts. The minimum unit of timer setting time is nanoseconds (ns).
- The timer starts when timer input ' IN ' changes to ' TRUE '. ' Q ' changes to ' TRUE '. Elapsed time ' ET ' is incremented as time elapses. When ' ET ' reaches the set time ' PT ', timer output ' Q ' changes to ' FALSE '. ' ET ' is not incremented after that. The timer is reset when ' IN ' changes to ' FALSE '. ' ET ' changes to 0, and ' Q ' still is ' FALSE '.
- If ' IN ' changes to ' FALSE ' after the timer is started, the timer will be continued until the ' ET ' reaches ' PT '. If ' IN ' changes to ' FALSE ' after the ' ET ' reaches ' PT ', ' ET ' will change to 0, and ' Q ' change to ' FALSE '.

- The example program is shown below

LD:

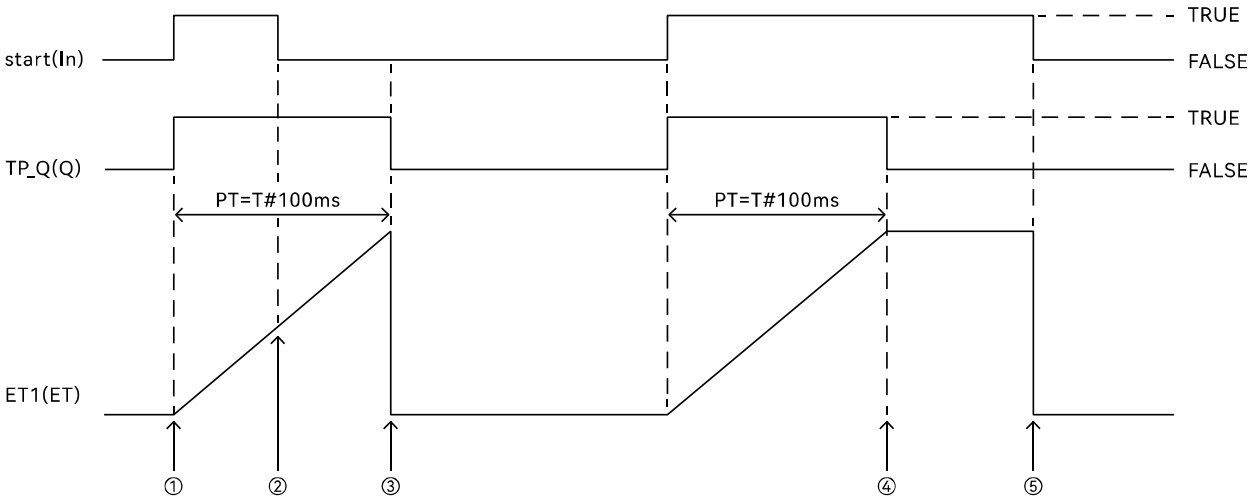
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	TP1		TP		
3	VAR	start		BOOL	TRUE	
4	VAR	TP_Q		BOOL	FALSE	
5	VAR	ET1		TIME	T#0ms	



ST:

```
TP1(EN: = TRUE , IN: = start , PT: = T#100ms , Q=> TP_Q , ET=> ET1 );
```

- The timing diagram is shown below



① If the timer "In" becomes TRUE, then "Q" will become TRUE and the timer will start timing.
② If the timer does not reach the set time and "In" becomes FALSE, then the timer will continue timing.
③ If the timer reaches the set time and "In" becomes FALSE, then "Q" will become FALSE, and "ET" will be reset to 0.
④ If the timer reaches the set time and "In" will remain to be TRUE, then "Q" will become FALSE, and "ET" will remain unchanged.
⑤ If the timer reaches the set time and "In" becomes FALSE, then "Q" will remain to be FALSE, and "ET" will be reset to 0.

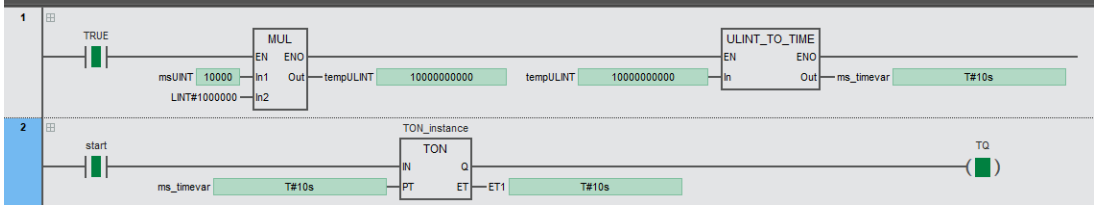
2.4.4 Example program for modifying timer via touchscreen

- The data type of input parameter ' PT ' is TIME, and the unit is ns (nanoseconds). If the touch screen does not support the date type of TIME. If users want to set the time in ms (milliseconds) to communicate with the controller through the touch screen, they can operate following the example below. When converting, bind the variable ' msUINT ' (UINT) to the controller device. In the software bind the variable ' msUINT ' to %MW100 (' Assign to ' the device in the variable table).
- 1ms=10^6ns.

  Firstly, the value of the variable ' msUINT ' is multiplied by 1000000 (ms → ns), and the result is put into ' tempULINT ' (the variable date type is ULINT). Then the ULINT type variable is converted into a TIME type variable through the ULINT_TO_TIME because the input parameter ' PT ' of the timer command is the TIME data type, so realize the conversion from ms to ns.
- When the timing time is 10S, the user only needs to set ' msUINT ' to 10000.
- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value	Comment
1	VAR	msUINT	%MW100	UINT	10000		
2	VAR	tempULINT		ULINT	10000000000		
3	VAR	ms_timevar		TIME	T#10s		
4	VAR	start		BOOL	TRUE		
5	VAR	TON_instance		TON			
7	VAR	ET1		TIME	T#10s		
8	VAR	TQ		BOOL	TRUE		



ST:

```
tempULINT: =msUINT*LINT#1000000;
mstimevar: =ULINT_ TO_ TIME(tempULINT);
TON_Instance(In: =start, PT: = mstimevar,ET=>ET1, Q=>TQ)
```

2.5 Counter

2.5.1 CTU（count up）

A counter that performs an add 1 operation each time the rising edge of an input signal is detected. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
CTU	Count up	FB	CTU_Instance CTU CU Q Reset CV PV	CTU_Instance (CU:= Parameter , Reset:= Parameter , PV:= Parameter , Q=> Parameter , CV=> Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
CU	Count input signal	Input	When CU changes from FALSE to TRUE, the CV value increases by one	FALSE or TRUE
Reset	Reset signal		TRUE: Reset the CV value to 0	FALSE or TRUE
PV	Set value		The setting value of the counter	0 ~ 4294967295
Q	Count completion flag	Output	The current count value has reached the set value flag TRUE: When the CV value is equal to the PV value FALSE: When the CV value is not equal to the PV value	FALSE or TRUE
CV	Current count value		The current value of the counter	0 ~ 4294967295

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	○																			
Reset	○																			
PV								○												
Q	○																			
CV								○												

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

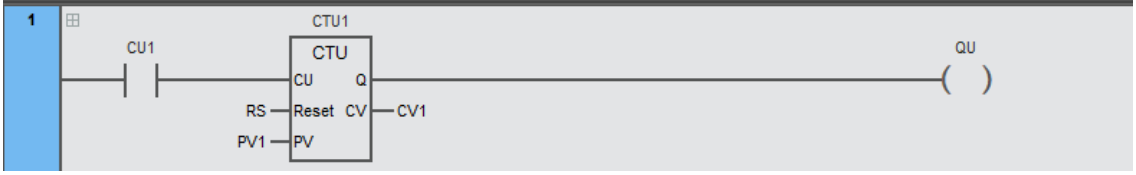
■ Function description

- A counter that performs an add 1 operation each time the rising edge of an input signal is detected.
- When the counter input signal 'CU' changes from FALSE to TRUE, the 'CV' value adds one. When the 'CV' value is equal to the 'PV' set value, the 'CV' value stops increasing, and 'Q' changes from FALSE to TRUE.
- When the counter reset signal 'Reset' is TRUE, the count completion flag 'Q' becomes FALSE, and the current count value 'CV' is reset to 0. When the counter reset signal 'Reset' is TRUE, the input signal 'CU' value is invalid, that is, the 'CV' value cannot be counted through the input signal 'CU'.

- The example program is shown below.

LD:

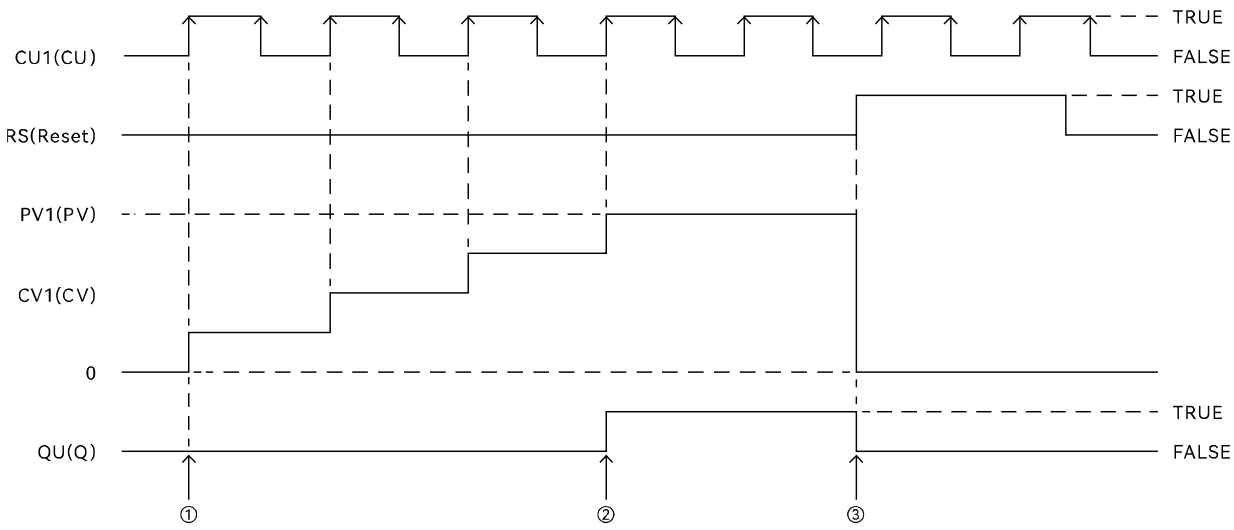
	Category	Name	Assigned to	Data Type	Initial Value
1	VAR	CTU1		CTU	
2	VAR	CU1		BOOL	
3	VAR	RS		BOOL	
4	VAR	PV1		UDINT	4
5	VAR	CV1		UDINT	
6	VAR	QU		BOOL	



ST:

```
CTU1(CU: = CU1 , Reset: = RS , PV: = PV1 , Q=> QU , CV=> CV1 );
```

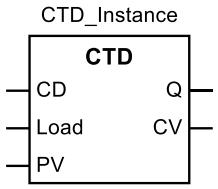
- The timing diagram is shown below.



① If "CU" becomes TRUE, "CV" will increase by 1.
② If "CV" equals "PV", then "Q" will become TRUE, and "CV" will remain unchanged.
③ If "Reset" becomes TRUE, then "Q" will become FALSE, and "CV" will become 0.

2.5.2 CTD（count down）

A counter that performs a minus 1 operation each time the rising edge of an input signal is detected. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
CTD	Count down	FB		CTD_Instance (CD:= Parameter , Load:= Parameter , PV:= Parameter , Q=> Parameter , CV=> Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
CD	Count input signal	Input	When CD changes from FALSE to TRUE, the CV value is subtracted by one	FALSE or TRUE
Load	Load signal		TRUE: Move PV value to CV	FALSE or TRUE
PV	Set value		The setting value of the counter	0 ~ 4294967295
Q	Count completion flag	Output	The current count value decreases to 0 flag TRUE: When the CV value is equal to 0 FALSE: When the CV value is not equal to 0	FALSE or TRUE
CV	Current count value		The current count value of the counter	0 ~ 4294967295

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CD	○																			
Load	○																			
PV								○												
Q	○																			
CV								○												

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

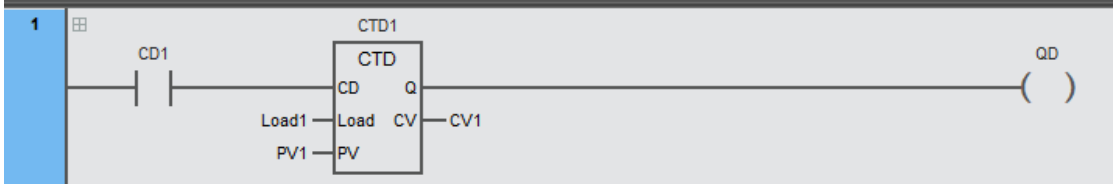
■ Function description

- A counter that performs a minus 1 operation each time the rising edge of an input signal is detected. Library: Standard.
- When the counter loading signal 'Load' is TRUE, the 'PV' value is moved to 'CV', and the count completion flag 'Q' is FALSE. At this point, the input signal 'CD' value is invalid, which indicates that the 'CV' value cannot be counted through the input signal 'CD'. When counting, the loading signal 'Load' needs to be changed to FALSE.
- When the counter loading signal 'Load' changes from TRUE to false, and the counter input signal 'CD' changes from false to TRUE, the 'CV' value decreases by 1. When the 'CV' value is equal to 0, the 'CV' value stops decreasing, and at the same time, 'Q' changes from false to TRUE.

- The example program is shown below.

LD:

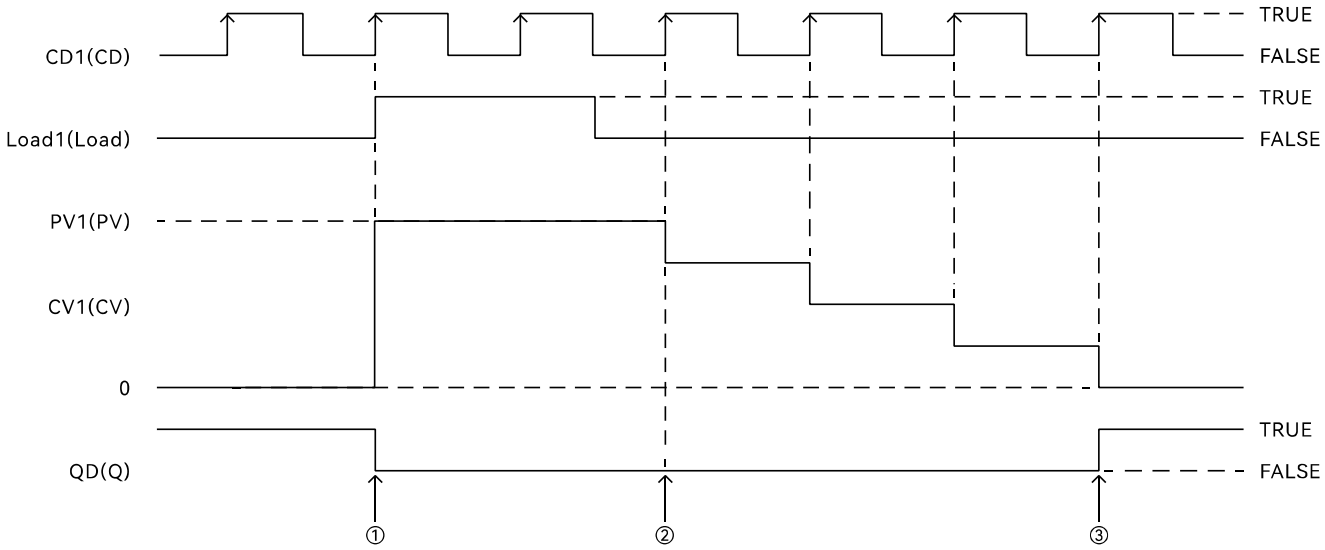
	Category	Name	Assigned to	Data Type	Initial Value
1	VAR	CTD1		CTD	
2	VAR	CD1		BOOL	
3	VAR	Load1		BOOL	
4	VAR	PV1		UDINT	4
5	VAR	CV1		UDINT	
6	VAR	QD		BOOL	



ST:

```
CTD1(CD: = CD1 , Load: = Load1 , PV: = PV1 , Q=> QD , CV=> CV1 );
```

- The timing diagram is shown below.



① If “Load” becomes TRUE, the “Q“ will become FALSE, and ”PV” will be shifted into “CV”.
② If “CD” becomes TRUE, then “CV” will decrease by 1.
③ If “CV” equals 0, then “Q” will become TRUE.

2.5.3 CTUD（count up and down）

A counter that performs an add or subtract operation based on the rising edge of the input signal for addition counting and the rising edge of the input signal for subtraction counting. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
CTUD	Count up and down	FB	CTUD_Instance CTUD CU QU CD QD Reset CV Load PV	CTUD_Instance (CU:= Parameter, CD:= Parameter, Reset:= Parameter , Load:= Parameter , PV:= Parameter , Q=> Parameter , CV=> Parameter).

■ Input/output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
CU	Add count input signal	Input	When CU changes from FALSE to TRUE, the CV value increases by one	FALSE or TRUE
CD	Subtraction input signal		When CD changes from FALSE to TRUE, the CV value is subtracted by one	FALSE or TRUE
Reset	Reset signal		TRUE: Reset the CV value to 0	FALSE or TRUE
Load	Load signal		TRUE: Move PV value to CV	FALSE or TRUE
PV	Set value		The setting value of the counter	0 ~ 4294967295
QU	Add count completion flag	Output	The current count value has reached the set value flag TRUE: When the CV value is equal to the PV value FALSE: When the CV value is not equal to the PV value	FALSE or TRUE
QD	Subtraction count completion flag		The current count value decreases to 0 flag TRUE: When the CV value is equal to 0 FALSE: When the CV value is not equal to 0	FALSE or TRUE
CV	Current count value		The current count value of the counter	0 ~ 4294967295

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	○																			
CD	○																			
Reset	○																			
Load	○																			
PV								○												
QU	○																			
QD	○																			
CV								○												

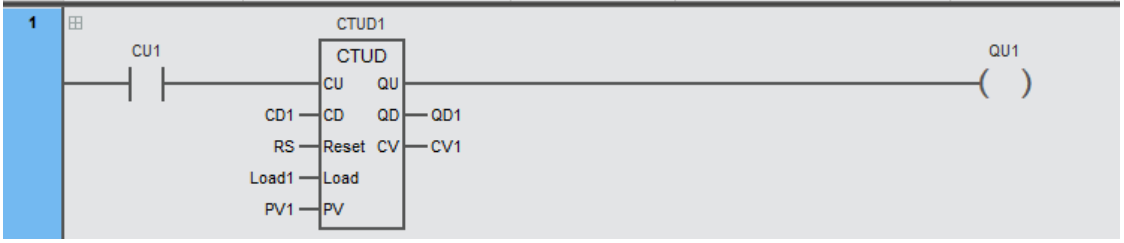
***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- A counter that performs an add or subtract operation based on the rising edge of the input signal for addition counting and the rising edge of the input signal for subtraction counting.
- When ' Reset ' and ' Load ' are FALSE, when the count input signal ' CU ' changes from FALSE to TRUE, the ' CV ' value adds one. When the ' CV ' value equals the ' PV ' set value, the ' CV ' value stops increasing, and at this time, ' QU ' changes from FALSE to TRUE.
- When ' Reset ' and ' Load ' are FALSE, when the subtraction count input signal ' CD ' changes from FALSE to TRUE, the ' CV ' value subtracts one. When the ' CV ' value equals 0, the ' CV ' value stops decreasing, and at this point, the ' QD ' changes from FALSE to TRUE.
- When the counter reset signal ' Reset ' is TRUE, the ' CV ' value becomes 0, the increase count completion flag ' QU ' is FALSE, and the decrease count completion flag ' QD ' is TRUE.
- When the counter loading signal ' Load ' is TRUE, the ' PV ' value is moved to ' CV ', the count completion flag ' QD ' becomes FALSE and ' QU ' becomes TRUE. At this point, the input signal ' CD ' value is invalid, which indicates that the ' CV ' value cannot be counted through the input signal ' CD '. When counting, the loading signal ' Load ' needs to be changed to FALSE.
- When both ' Reset ' and ' Load ' are TRUE, the ' CV ' value becomes 0, the add count completion flag ' QU ' is false, and the subtraction count completion flag ' QD ' is TRUE.
- The example program is shown below.

LD:

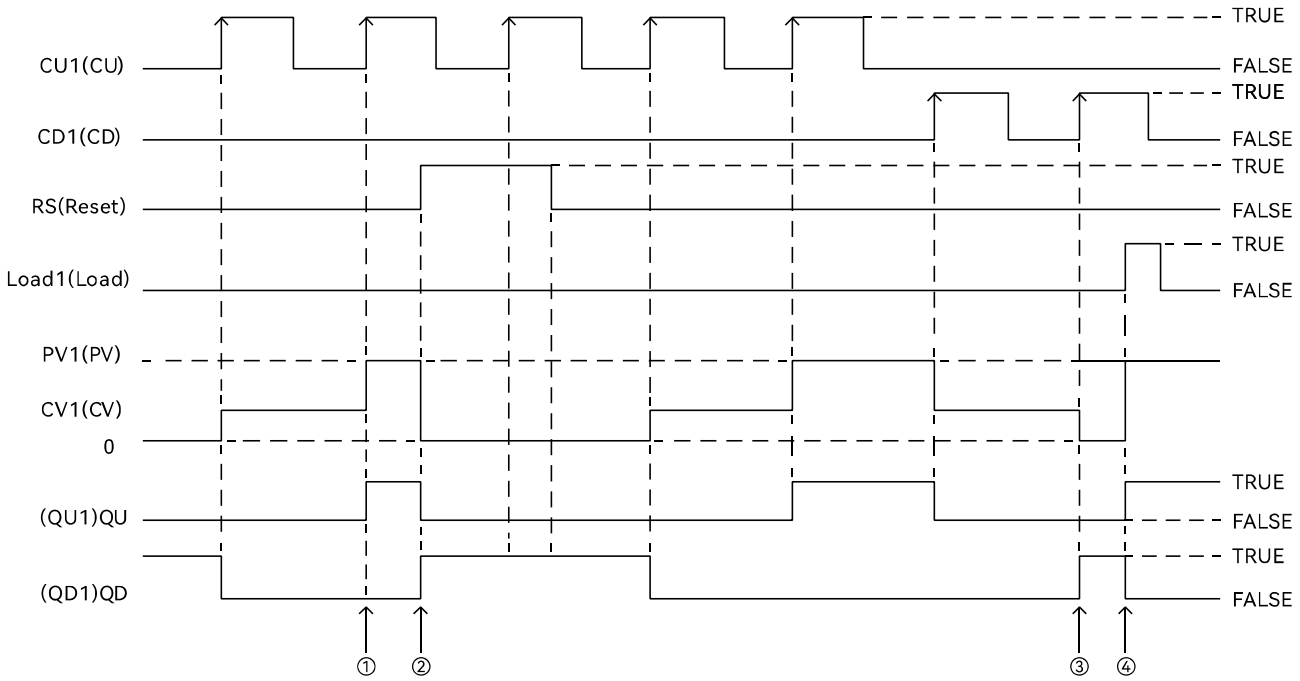
	Category	Name	Assigned to	Data Type	Initial Value
1	VAR	CTUD1		CTUD	
2	VAR	CU1		BOOL	
3	VAR	CD1		BOOL	
4	VAR	RS		BOOL	
5	VAR	Load1		BOOL	
6	VAR	PV1		UDINT	2
7	VAR	CV1		UDINT	
8	VAR	QU1		BOOL	
9	VAR	QD1		BOOL	



ST:

```
CTUD1(
 CU: = CU1 ,
 CD: = CD1 ,
 Reset: = RS ,
 Load: = Load1 ,
 PV: = PV1 ,
 QU=> QU1 ,
 QD=> QD1 ,
 CV=>CV1 );
```

- The timing diagram is shown below.

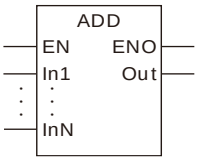


① If “CV” equals “PV”, then “QU” will become TRUE, and “CV” will stop increasing.
② If “Reset” becomes TRUE, then “QU” will become FALSE, “QD” will become TRUE, and “CV” will become 0.
③ If “CV” becomes 0, then “QD” will become TRUE, and “CV” will stop decreasing.
④ If “Load” becomes TRUE, then “QU” will become TRUE, “QD” will become FALSE, and ”PV” will be shifted into “CV”.

2.6 Math functions

2.6.1 ADD（addition）

Perform addition operations on data of bit string, integer, real number, time, and date. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
ADD	Addition	FUN		Out: =In1 + In2 + ⋯ + InN;

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Augend	Input	Augend	Depends on variable type
In2 to InN	Addend		When the program is written, the number of additions can be increased or decreased by the programming software, and the range of N is from 2 to 8	Depends on variable type
Out	Sum	Output	Sum of In1~InN	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○		○	○	
In2 to InN		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○				
Out		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○		○	○	

***Note:** The ' ○ ' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- Add input data of bit string, integer, real number, time, date, and output the result as ' Out ', that is, Out=In1+In2+...+InN.
- When the input variable types are bit string, integer, or real number, it is allowed that the input variables ' In1 ' to ' InN ' have different data types. When the data types from ' In1 ' to ' InN ' are different, perform the operation with the data types containing ' In1 ' to ' InN '. If the data type of ' In1 ' is SINT and the data type of ' In2 ' is INT, then the operation is processed using INT.
- The width of the ' Out ' data type must be greater than or equal to the maximum width of types from ' In1 ' to ' InN ', otherwise software compilation errors will occur. If the data types for ' In1 ' and ' In2 ' are INT and SINT, respectively, then the data types for ' Out ' should be INT, DINT, etc.
- When performing addition operations on data of time and date types, only the following three combinations are supported:

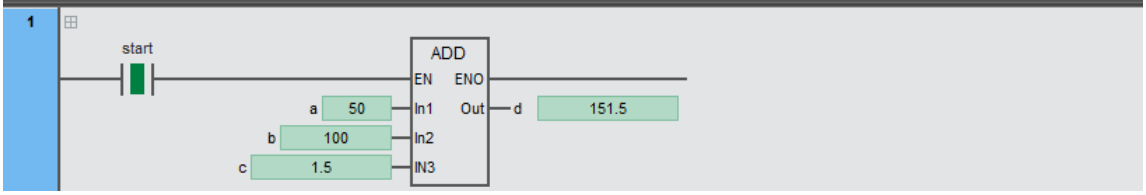
  1. In1, In2, and Out data types are all TIME.
  2. The data type for ' In1 ' is TOD (TIMEOF-DAY), the data type for ' In2 ' is TIME, and the data type for ' Out ' is TOD.
  3. The data type for ' In1 ' is DT (DAY-AND-TIME), the data type for ' In2 ' is TIME, and the data type for ' Out ' is DT.

⚠Note

- The sum of ' In1 ' to ' InN ' may exceed the valid range of the ' Out ' data type. If the data types of ' In1 ' and ' In2 ' are both INT and have numerical values of 32767 and 1, respectively, if the data type of the ' Out ' variable is INT, the value of ' Out ' is -32768; If the ' Out ' data type is DINT, the value is 32768.
- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	start		BOOL	TRUE	
2	VAR	a		INT	50	
3	VAR	b		DINT	100	
4	VAR	c		REAL	1.5	
5	VAR	d		REAL	151.5	

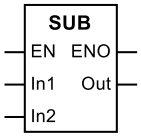


ST:

```
d: = a + b + c ;
```

2.6.2 SUB（subtraction）

Subtract data of bit string, integer, real number, time, date. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
SUB	Subtraction	FUN		Out: = In1- In2;

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Minuend	Input	Minuend	Depends on variable type
In2	Subtrahend		Subtrahend	Depends on variable type
Out	Difference	Output	The difference between In1 and In2	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
In2		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
Out		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○		○	○	

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- Subtract the subtracted number 'In1' from the subtracted number 'In2' and output the result to 'Out', that is, Out=In1- In2.
- When the input variable types are string, integer, and real number, it is allowed that the input variables 'In1' to 'In2' have different data types. When the data types from 'In1' to 'In2' are different, perform the operation with the data types containing 'In1' to 'In2'. If the data type of 'In1' is SINT and the data type of 'In2' is INT, then the operation is processed using INT.
- For time and date types, only the following types of subtraction are supported:

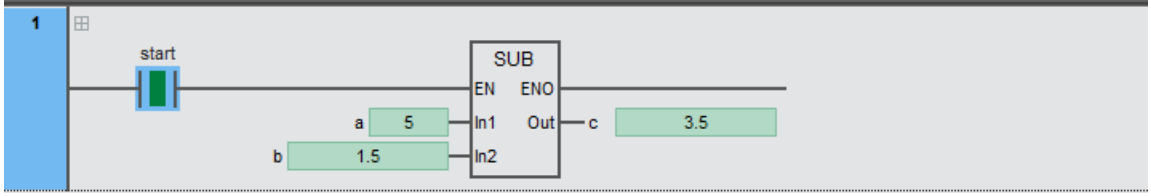
1. The data types 'In1', 'In2', and 'Out' are all TIME.
2. The data type for 'In1' is TOD, the data type for 'In2' is TIME, and the data type for 'Out' is TOD.
3. The data types for 'In1' and 'In2' are TOD, and the data type for 'Out' is TIME.
4. The data types for 'In1' and 'In2' are DATE, and the data type for 'Out' is TIME.
5. The data types for 'In1' and 'In2' are DT, and the data type for 'Out' is TIME.
6. The data types for 'In1' and 'In2' are DT, and the data type for 'Out' is TIME.

⚠Note

- The difference between ' In1 ' and ' In2 ' may exceed the valid range of the ' Out ' data type. If the data types of ' In1 ' and ' In2 ' are both INT and have values of -32768 and 1, respectively, if the data type of ' Out ' is INT, the value of ' Out ' is 32767; If the ' Out ' data type is DINT, then the value of ' Out ' is -32769.
- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value
1	VAR	a		INT	5
2	VAR	b		REAL	1.5
3	VAR	c		REAL	3.5
4	VAR	start		BOOL	TRUE

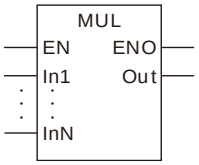


ST:

c: = a – b ;

2.6.3 MUL（multiplication）

Multiply data of bit string, integer, and real number. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
MUL	Multiplication	FUN		Out: =In1 * In2 *…* InN;

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Multiplicand	Input	Multiplicand	Depends on variable type
In2 to InN	Multiplier		Multipliers can be increased or decreased through programming software, values to compare N = 2 to 8	Depends on variable type
Out	Product	Output	The product of In1~InN	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1 to InN		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out		○	○	○	○	○	○	○	○	○	○	○	○	○	○					

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- Multiply input data of bit string, integer, and real number, and output the result to 'Out', that is, Out=In1 * In2 *... * InN.
- Multiply input data of bit string, integer, and real number, and output the result to 'Out', that is, Out=In1 * In2 *... * InN.
- The width of the 'Out' data type must be greater than or equal to the maximum width of types 'In1' to 'In2', otherwise software compilation errors will occur. If the data types for 'In1' and 'In2' are INT and SINT, respectively, then the data types for 'Out' should be INT, DINT, etc.

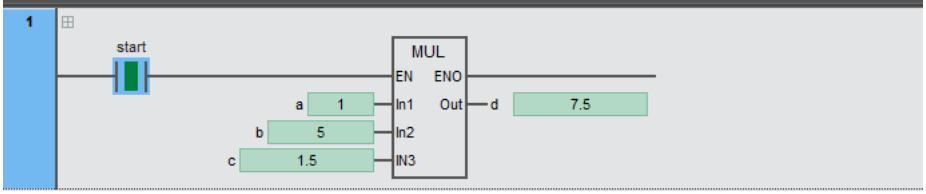
⚠Note

- The product of 'In1' to 'InN' may exceed the valid range of the 'Out' data type. If the data types of 'In1' and 'In2' are both INT and have values of 20000 and 2, respectively, if the data type of 'Out' is INT, the value of 'Out' is -25536; If the 'Out' data type is DINT, then the 'Out' value is 40000.

- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value
1	VAR	a		INT	1
2	VAR	b		DINT	5
3	VAR	c		REAL	1.5
4	VAR	d		REAL	7.5
5	VAR	start		BOOL	TRUE

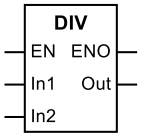


ST:

```
d: = a * b * c;
```

2.6.4 DIV（division）

Perform division operations on data of bit string, integer, and real number. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DIV	Division	FUN		Out: =In1 / In2;

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Dividend	Input	Dividend	Depends on variable type
In2	Divisor		Divisor	Depends on variable type
Out	Quotient	Output	Quotient of In1 and In2	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
In2		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out		○	○	○	○	○	○	○	○	○	○	○	○	○	○					

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

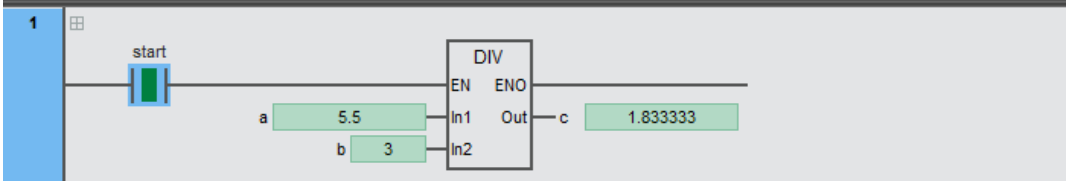
- The dividend 'In1' is divided by the divisor 'In2', and the result is output to 'Out'.Out=In1/In2.
- When the input variable types are bit string, integer, and real number, it is allowed that the input variables 'In1' to 'In2' have different data types. When the data types from 'In1' to 'In2' are different, the operation should include the data types from 'In1' to 'In2'. For example, if the data type of 'In1' is SINT and the data type of 'In2' is INT, the operation will be processed using INT.
- The width of the 'Out' data type must be greater than or equal to the maximum width of types 'In1' to 'In2', otherwise software compilation errors will occur. For example, if the data types for 'In1' and 'In2' are INT and SINT, respectively, then the data types for 'Out' should be INT, DINT, etc.
- When performing addition operations on data of time and date types, only the following three combinations are supported:
  1. The data types 'In1', 'In2', and 'Out' are all TIME.
  2. The data type for 'In1' is TOD (TIMEOF-DAY), the data type for 'In2' is TIME, and the data type for 'Out' is TOD.
  3. The data type for 'In1' is DT (DAY-AND-TIME), the data type for 'In2' is TIME, and the data type for 'Out' is DT.

⚠Note

- The quotient of ' In1 ' and ' In2 ' may exceed the valid range of the ' Out ' data type. If the data type of In1 and In2 is INT, the values are -32768 and -1, respectively. If the data type of Out is INT, the value of Out is -32768; If the Out data type is DINT, the value of Out is 32768.
- When the divisor is 0, the value of ' Out ' is 0.
- The example program is shown below .

LD:

	Category	Name	Assigned to	Data Type	Online Value
1	VAR	a		REAL	5.5
2	VAR	b		INT	3
3	VAR	c		REAL	1.833333
4	VAR	start		BOOL	TRUE

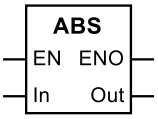


ST:

```
c: =a/b;
```

2.6.5 ABS（absolute value calculation）

Calculate the absolute value of bit string, integer, and real number. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
ABS	Absolute value calculation	FUN		Out: = ABS(In);

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Raw value	Input	Raw value	Depends on variable type
Out	Absolute value	Output	Absolute value	Depends on variable type

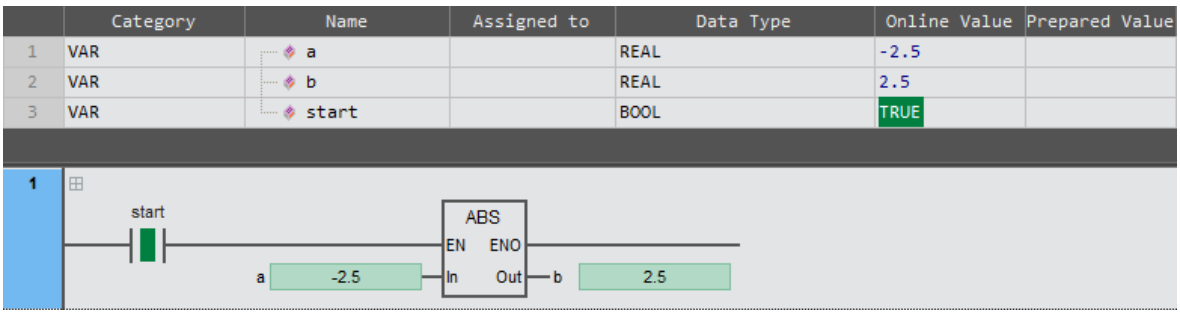
	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out		○	○	○	○	○	○	○	○	○	○	○	○	○	○					

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- Calculate the absolute value of 'In' and output the result to 'Out'.
- When the 'In' and 'Out' data types are different, the width of the 'Out' data type must be greater than or equal to the width of the 'In' data type, otherwise software compilation errors will occur.
- The example program is shown below.

LD:

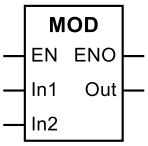


ST:

```
b: =ABS( a );
```

2.6.6 MOD(integer remainder of division)

Calculate the remainder of a division operation on bit string and integer. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
MOD	Integer remainder of division	FUN		Out: = In1 MOD In2;

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Dividend	Input	Dividend	Depends on variable type
In2	Divisor		Divisor	Depends on variable type
Out	Remainder	Output	The remainder of dividing In1 by In2	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1		○	○	○	○	○	○	○	○	○	○	○	○							
In2		○	○	○	○	○	○	○	○	○	○	○	○							
Out		○	○	○	○	○	○	○	○	○	○	○	○							

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

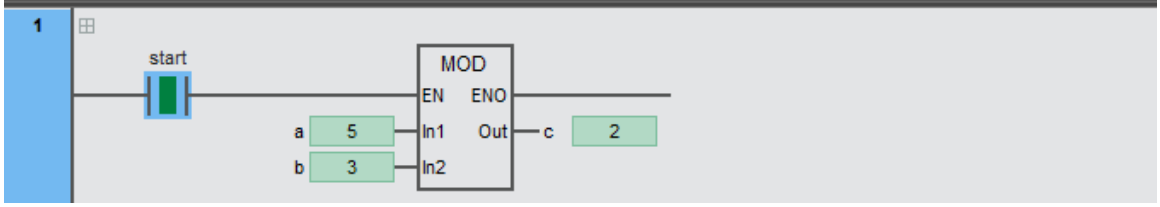
- Output the remainder of dividing the dividend 'In1' by the divisor 'In2' to 'Out'.
- When the input variable types are bit string, integer and real number, different data types of input variables 'In1' to 'In2' are allowed. When the data types from 'In1' to 'In2' are different, perform the operation with the data types containing 'In1' to 'In2'. If the data type of 'In1' is SINT and the data type of 'In2' is INT, then the operation is processed using INT.
- The width of the 'Out' data type must be greater than or equal to the maximum width of types 'In1' to 'In2', otherwise software compilation errors will occur. For example, if the data types for 'In1' and 'In2' are INT and SINT, respectively, then the data types for 'Out' should be INT, DINT, etc.

⚠Note

- When the value of ' In2 ' is 0, the value of ' Out ' is 0.
- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value
1	VAR	a		INT	5
2	VAR	b		INT	3
3	VAR	c		INT	2
4	VAR	start		BOOL	TRUE

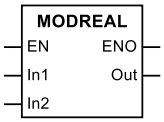


ST:

```
c: = a MOD b;
```

2.6.7 MODREAL（real number remainder of division）

Calculate the remainder after dividing a real number. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
MODREAL	Real number remainder of division	FUN		Out: =MODREAL(In1 , In2);

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Dividend	Input	Dividend	Depends on variable type
In2	Divisor		Divisor	Depends on variable type
Out	Remainder	Output	The remainder of dividing In1 by In2	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1														○	○					
In2														○	○					
Out														○	○					

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

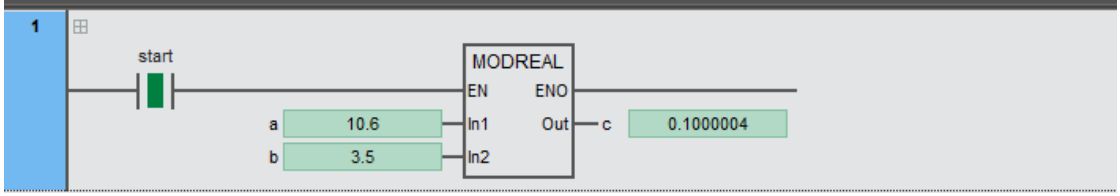
- Output the remainder of dividing the dividend 'In1' by the divisor 'In2' to 'Out'.
- Allow input variables 'In1' to 'In2' with different data types. When the data types from 'In1' to 'In2' are different, perform the operation with the data types containing 'In1' to 'In2'. For example, if the data type of 'In1' is REAL and the data type of 'In2' is LREAL, LREAL will be used for operation processing.
- The width of the 'Out' data type must be greater than or equal to the maximum width of types 'In1' to 'In2', otherwise software compilation errors will occur.

⚠Note

- When the value of ' In2 ' is 0, the value of ' Out ' is 0.
- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value
1	VAR	a		REAL	10.6
2	VAR	b		REAL	3.5
3	VAR	c		REAL	0.1000004
4	VAR	start		BOOL	TRUE

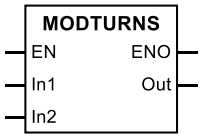


ST:

```
c: =MODREAL( a , b );
```

2.6.8 MODTURNS（modulo rotation number calculation）

Take an integer as the quotient of a division operation on a real number. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
MODTURNS	Modulo rotation number calculation	FUN		Out: =MODTURNS(In1 , In2)；

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Input value	Input	Input value	Depends on variable type
In2	Mode range		Mode range	Depends on variable type
Out	Cycles	Output	Cycles	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1														○	○					
In2														○	○					
Out												○								

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- Divide 'In1' and 'In2' and round their quotients to output 'Out'.
- 'Out' follows rounding rules: rounding downwards. When the quotient is an integer, directly remove the decimal part and round it up; When the quotient is negative, after removing decimals, it needs to be rounded up to -1.

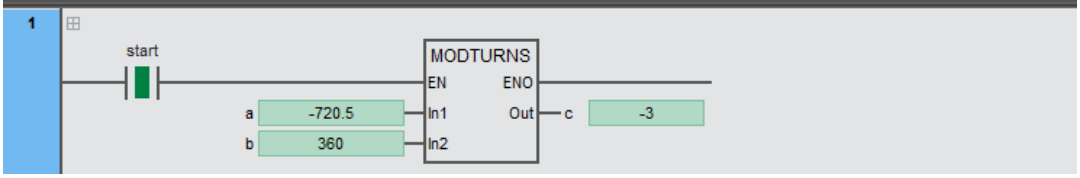
'In1'	'In2'	'Out'
720	360	2
-720	360	-2
721.8	360	2
-721.8	360	-3

⚠Note

- The input variable and output variable data types of this instruction are different.
- When the value of ' In2 ' is 0, the value of ' Out ' is 0.
- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value
1	VAR	a		REAL	-720.5
2	VAR	b		REAL	360
3	VAR	c		DINT	-3
4	VAR	start		BOOL	TRUE

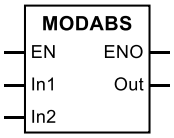


ST:

```
c : =MODTURNS( a , b );
```

2.6.9 MODABS（modulo set position calculation）

The phase is obtained by operating on two real variables or constants. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
MODABS	Modulo set position calculation	FUN		Out: = MODABS(In1 , In2)；

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Input value	Input	Input value	Depends on variable type
In2	Modulo range		Modulo range	Depends on variable type
Out	Phase	Output	The phase obtained by dividing In1 by In2	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1														○	○					
In2														○	○					
Out														○	○					

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- This instruction is used to calculate the phase obtained by performing an operation on two real variables or constants, and the result is output to 'Out'. The 'Out' is a positive number, and the phase of the shaft or cam can be calculated using this command. The examples of values for 'In1', 'In2', and 'Out' are shown in the table below.

'In1'	'In2'	'Out'
540	360	180
-540	360	180
560	360	200
-560	360	160

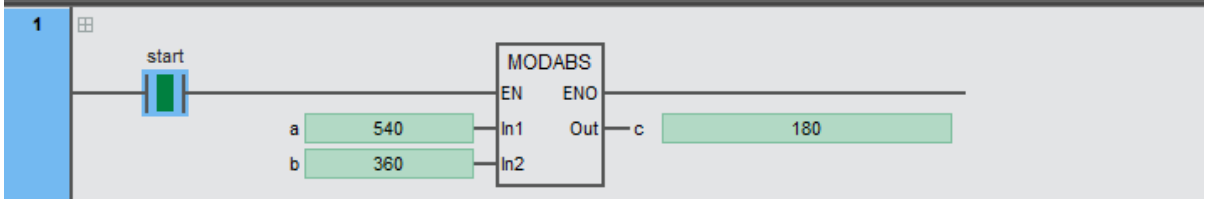
- Allow input variables 'In1' and 'In2' to be variables of different data types. When 'In1' and 'In2' are variables of different data types, perform the operation with the data type that includes all the value ranges of 'In1' and 'In2'. For example, if the data type of 'In1' is REAL and the data type of 'In2' is LREAL, then the LREAL data type is used for processing.
- The width of the 'Out' data type must be greater than or equal to the width of the 'In1' and 'In2' types, otherwise, software compilation will report an error. For example, if the data types for 'In1' and 'In2' are REAL and LREAL, respectively, then the data type for 'Out' needs to be LREAL, etc; If the data type of the 'Out' linked variable is REAL, an error will be reported during software compilation.

⚠Note

- When the value of ' In2 ' is 0, the value of ' Out ' is 0.
- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value
1	VAR	a		REAL	540
2	VAR	b		REAL	360
3	VAR	c		LREAL	180
4	VAR	start		BOOL	TRUE



ST:

```
c: =MODABS( a , b) ;
```

2.6.10 RadToDeg（radian to degree）

This instruction is used to convert radian to degree. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
RadToDeg	Radian to degree	FUN	RadToDeg EN ENO In Out	Out: = RadToDeg(In);

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Radians	Input	The radian value to be converted	Depends on variable type
Out	Degree	Output	Converted degree value	Depends on variable type

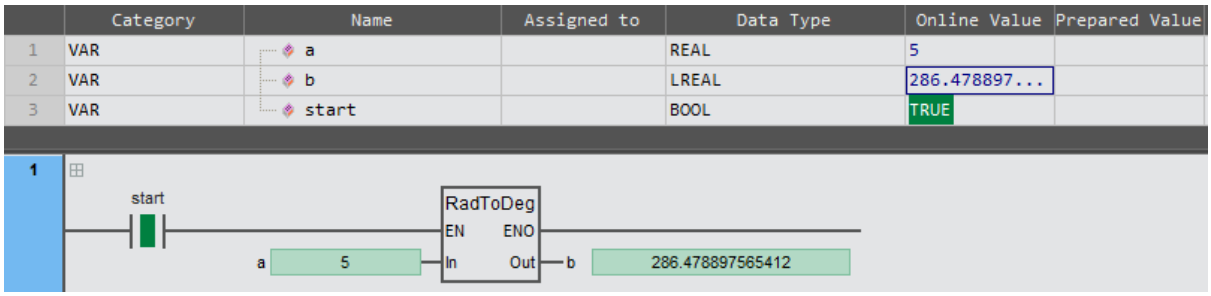
	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out														○	○					

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- This instruction is used to convert the radian (rad) value in the input variable 'In' to the degree value (°) and output it to 'Out'. The conversion formula Out=In * 180/π.
- The example program is shown below.

LD:



ST:

```
b: = RadToDeg( a );
```

2.6.11 DegToRad（degree to radian）

This instruction is used to convert degree to radian. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DegToRad	Degree to radian	FUN	DegToRad EN ENO In Out	Out: = DegToRad (In);

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Degree	Input	The degree value to be converted	Depends on variable type
Out	Radians	Output	The converted radian value	Depends on variable type

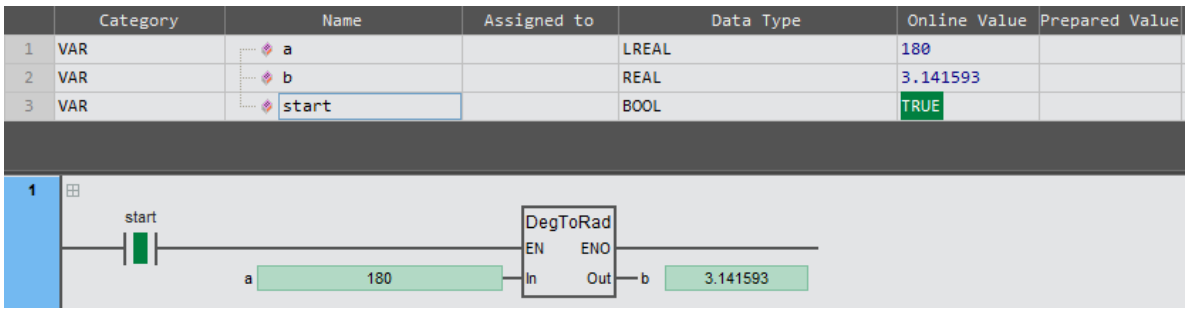
	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out														○	○					

*Note: The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- This instruction is used to convert the degree (°) value in the input variable 'In' to the radian value (rad) , and then output it to 'Out'.
- The conversion formula Out=(In/180) * π.
- The example program is as follows:

LD:



ST:

```
b: = DegToRad ( a );
```

2.6.12 SIN/COS/TAN（trigonometric function）

Calculate the trigonometric function of the input variable ' In ' and output the result to ' Out '. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
SIN	Sine	FUN	SIN EN ENO In Out	Out: = SIN(In);
COS	Cosine	FUN	COS EN ENO In Out	Out: = COS(In);
TAN	Tangent	FUN	TAN EN ENO In Out	Out: = TAN(In);

■ Input /output variable descriptions and data types

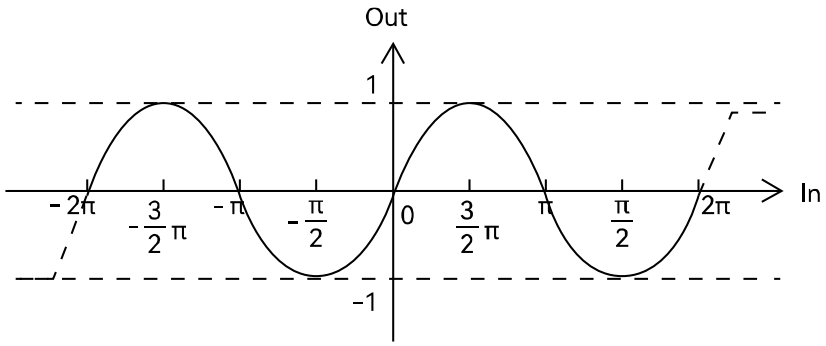
Name	Meaning	I/O	Description	Parameter scope
In	Input value	Input	Value to be converted. Unit: radians	Depends on variable type
Out	Output value	Output	The result of trigonometric function operation	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out														○	○					

***Note:** The ' ○ ' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

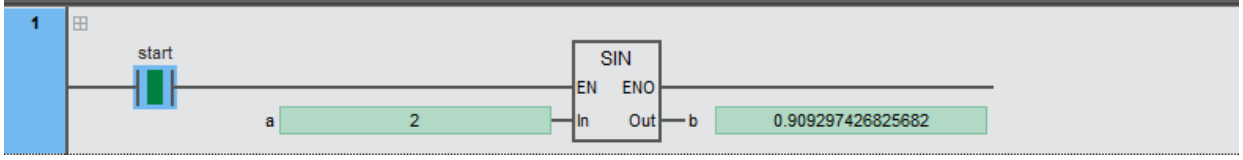
- The SIN instruction is used to perform a sine operation on the value of ' In ' and output the result to ' Out '. The unit of ' In ' is radians.



- The example program is as follows.

LD:

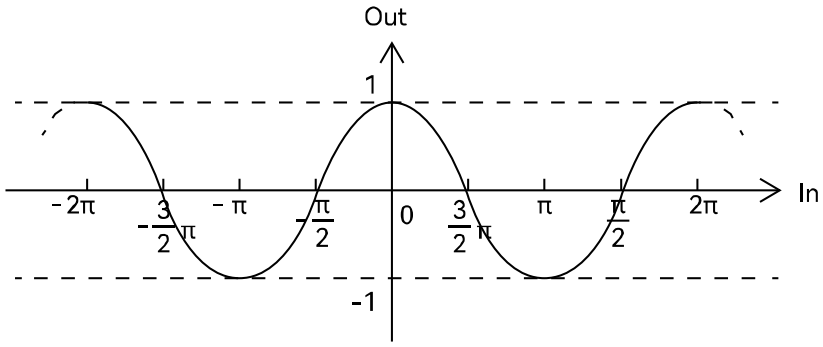
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		LREAL	2	
2	VAR	b		LREAL	0.90929742...	
3	VAR	start		BOOL	TRUE	



ST:

```
b: = SIN( a );
```

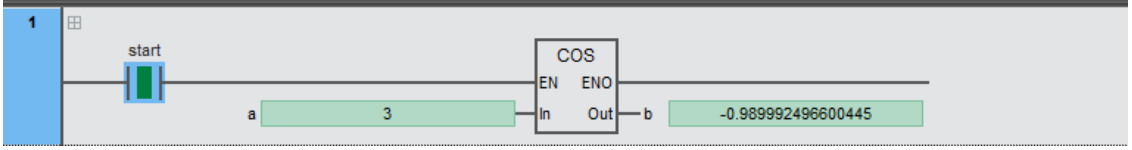
- The COS instruction is used to perform a cosine operation on the value of ' In ' and output the result to ' Out ' . The unit of ' In ' is radians.



- The example program is as follows

LD:

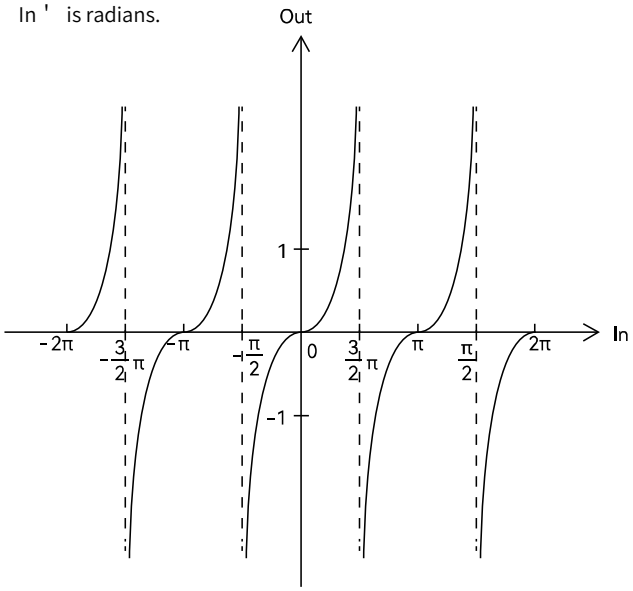
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		LREAL	3	
2	VAR	b		LREAL	-0.9899924...	
3	VAR	start		BOOL	TRUE	



ST:

```
b: = COS( a );
```

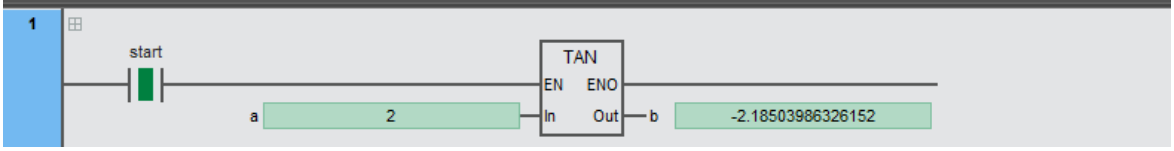
- The TAN instruction is used to perform a tangent operation on the value of 'In' and output the result to 'Out'. The unit of 'In' is radians.



- The example program is as follows:

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		LREAL	2	
2	VAR	b		LREAL	-2.1850398...	
3	VAR	start		BOOL	TRUE	



ST:

```
b: = TAN( a );
```

2.6.13 ASIN/ACOS/ATAN（inverse trigonometric function）

Perform inverse trigonometric function calculation on input variables and output the result to 'Out'.

Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
ASIN	Arcsine	FUN	ASIN EN ENO In Out	Out: =ASIN(In);
ACOS	Arccosine	FUN	ACOS EN ENO In Out	Out: = ACOS(In);
ATAN	Arctangent	FUN	ATAN EN ENO In Out	Out: = ATAN(In);

■ Input /output variable descriptions and data types

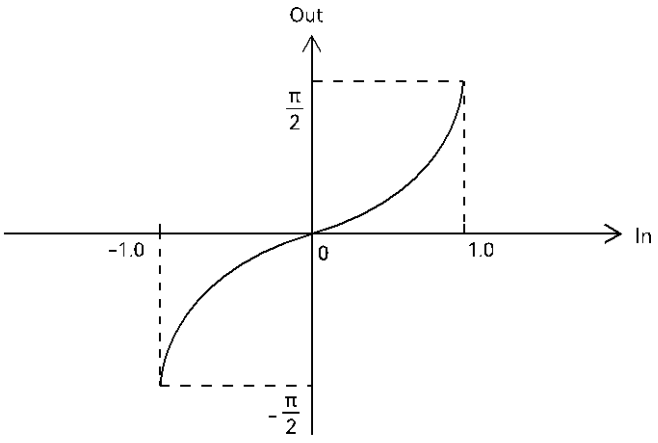
Name	Meaning	I/O	Description	Parameter scope
In	Input value	Input	Value to be converted	Depends on variable type
Out	Output value	Output	The result obtained through inverse trigonometric operation. Unit: radians.	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out														○	○					

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

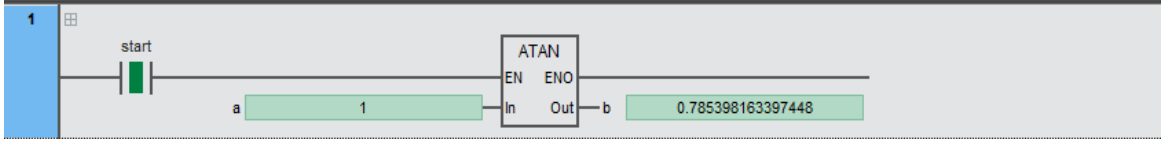
- The ASIN instruction is used to perform an anti-sine operation on the value of 'In' and output the result to 'Out'. The unit of 'Out' is radians.



- The example program is shown below.

LD:

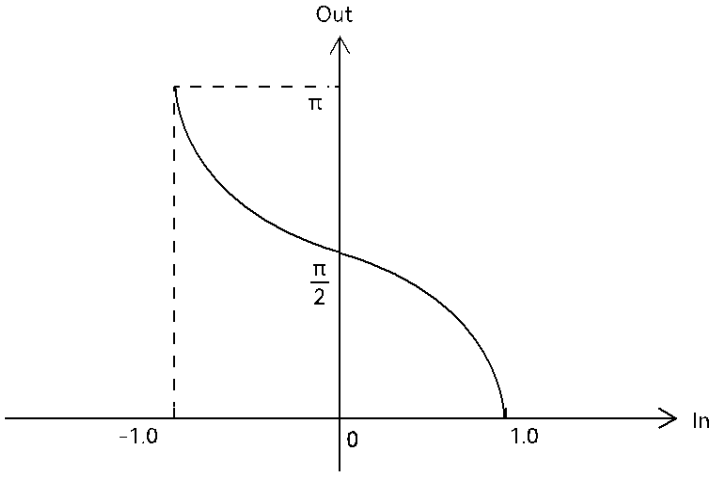
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		LREAL	1	
2	VAR	b		LREAL	0.78539816...	
3	VAR	start		BOOL	TRUE	



ST:

b: =ASIN (a);

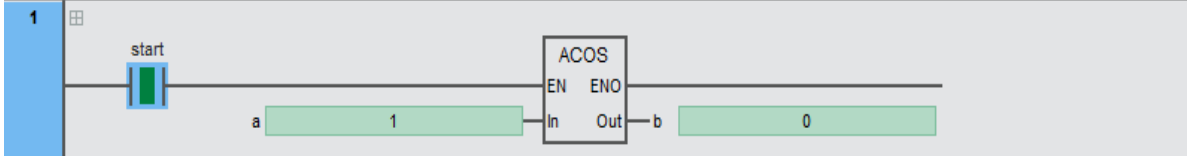
- The ACOS instruction is used to perform an inverse cosine operation on the value of ' In ' and output the result to ' Out ' . The unit of ' Out ' is radians.



- The example program is shown below.

LD:

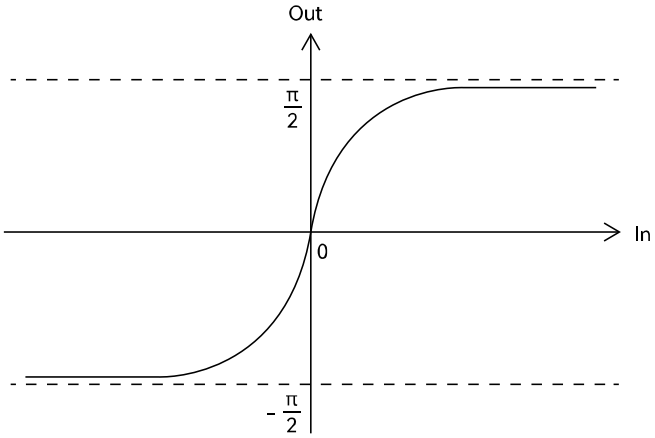
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		LREAL	1	
2	VAR	b		LREAL	0	
3	VAR	start		BOOL	TRUE	



ST:

b: = ACOS(a);

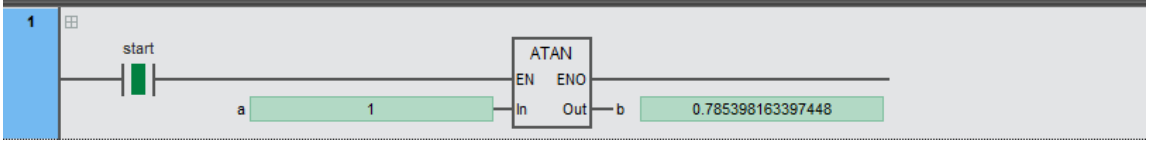
- The ATAN instruction is used to perform an arctangent operation on the value of ' In ' and output the result to ' Out '. The unit of ' Out ' is radians.



- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		LREAL	1	
2	VAR	b		LREAL	0.78539816...	
3	VAR	start		BOOL	TRUE	



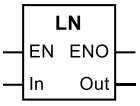
ST:

```
b: = ATAN( a );
```

2.6.14 LN（the natural logarithm of a number）

This instruction is used to calculate the natural logarithm of the input variable 'In' and output the result to 'Out'.

Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
LN	The natural logarithm of a number	FUN		Out: = LN(In);

■ Input /output variable descriptions and data types

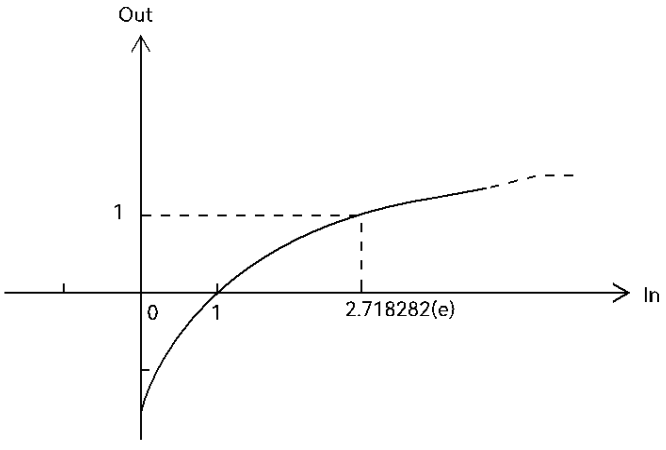
Name	Meaning	I/O	Description	Parameter scope
In	Input value	Input	Value to be converted	Depends on variable type
Out	Logarithm	Output	Natural logarithm	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out														○	○					

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

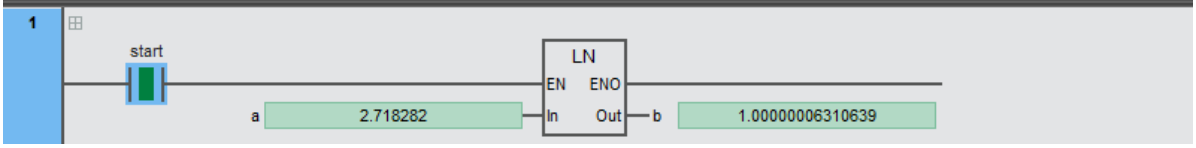
- The LN instruction is used to perform a natural logarithmic operation on the value of 'In', which calculates the logarithm of e (e=2.718282) and outputs the result to 'Out'.



- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		LREAL	2.718282	
2	VAR	b		LREAL	1.00000006...	
3	VAR	start		BOOL	TRUE	



ST:

```
b: = LN( a );
```

2.6.15 LOG（the base-10 logarithm of a number）

The LN instruction is used to perform a natural logarithmic operation on the value of ' In ', which calculates the logarithm of e (e=2.718282) and outputs the result to ' Out '. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
LOG	The base-10 logarithm of a number	FUN	LOG EN ENO In Out	Out: = LOG(In);

■ Input /output variable descriptions and data types

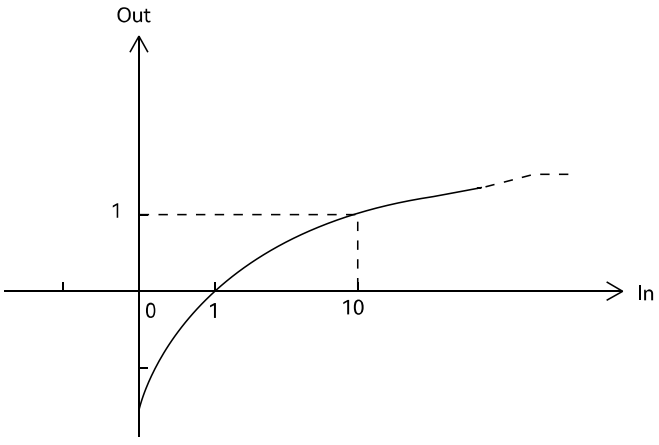
Name	Meaning	I/O	Description	Parameter scope
In	Input value	Input	Value to be converted	Depends on variable type
Out	Logarithm	Output	The base-10 logarithm of a number	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out														○	○					

***Note:** The ' ○ ' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

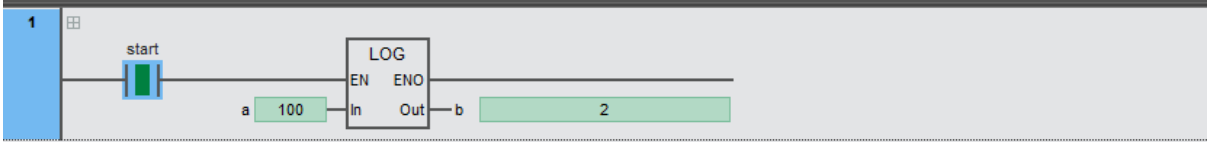
- The LOG instruction is used to perform a common logarithmic operation on the value of ' In ', which calculates the logarithm of ' In ' based on 10 and outputs the result to ' Out '.



- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		INT	100	
2	VAR	b		LREAL	2	
3	VAR	start		BOOL	TRUE	



ST:

```
b: = LOG ( a );
```

2.6.16 SQRT（the square root of a number）

This instruction is used to calculate the square root of the input variable ' In ' and output the result to ' Out '.

Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
SQRT	The square root of a number	FUN	SQRT EN ENO In Out	Out: = SQRT(In);

■ Input /output variable descriptions and data types

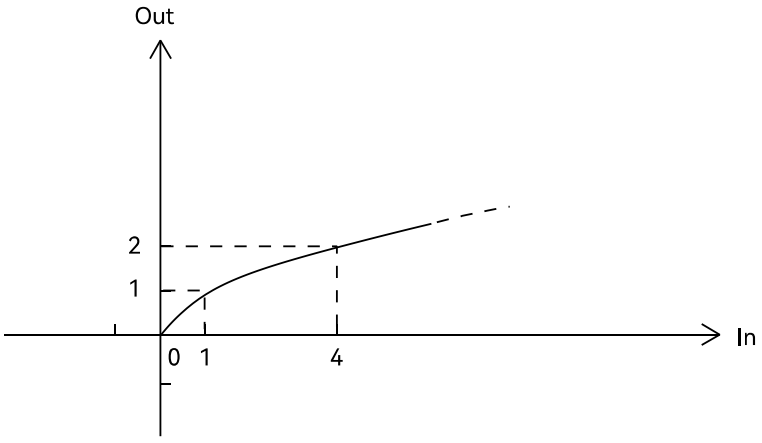
Name	Meaning	I/O	Description	Parameter scope
In	Operational value	Input	Operational value	Depends on variable type
Out	Square root	Output	Square root	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out														○	○					

***Note:** The ' ○ ' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

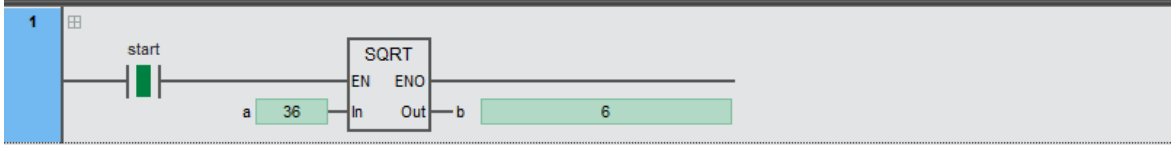
- This instruction is used to calculate the square root of the input variable ' In ' and output the result to ' Out '.



- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		INT	36	
2	VAR	b		LREAL	6	
3	VAR	start		BOOL	TRUE	

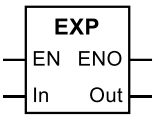


ST:

```
b: = SQRT( a );
```

2.6.17 EXP（exponential function）

This instruction is used to calculate a power operation with ' e ' as the base and ' In ' as the exponential, and output the result to ' Out ' . Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
EXP	Exponential function	FUN		Out: = EXP(In) ;

■ **Input /output variable descriptions and data types**

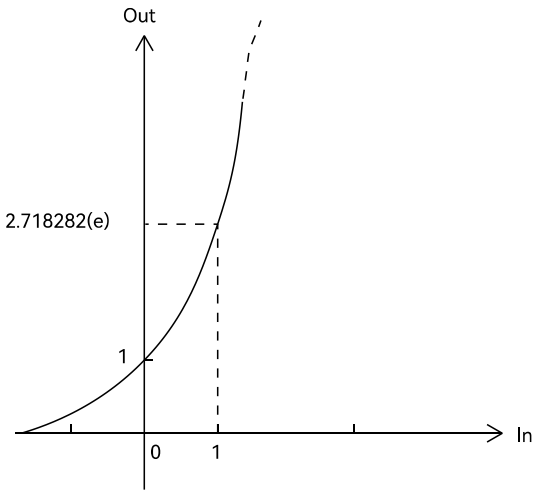
Name	Meaning	I/O	Description	Parameter scope
In	Exponential	Input	Exponential	Depends on variable type
Out	Operation result	Output	The result of powering with e as the base and In as the exponent	Depends on variable type, nonnegative number

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out														○	○					

***Note:** The ' ○ ' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ **Function description**

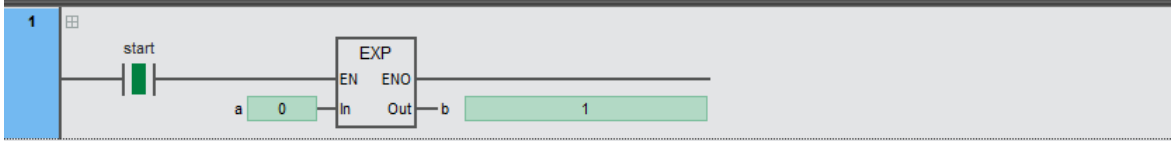
- This instruction is used to calculate a power operation with ' e ' as the base and ' In ' as the exponent, and output the result to ' Out ' .



- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		INT	0	
2	VAR	b		LREAL	1	
3	VAR	start		BOOL	TRUE	

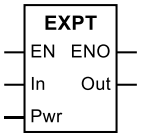


ST:

```
b: =EXP ( a );
```

2.6.18 EXPT（power-exponential function）

This instruction is used to calculate a power operation with ' In ' as the base and ' Pwr ' as the exponential, and output the result to ' Out ' . Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
EXPT	Power-exponential function	FUN		Out: = EXPT(In , Pwr);

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Base	Input	Base	Depends on variable type
Pwr	Exponential		Exponential	Depends on variable type
Out	Operation result	Output	The calculation result of powering with In as the base and Pwr as the exponential	Depends on variable type

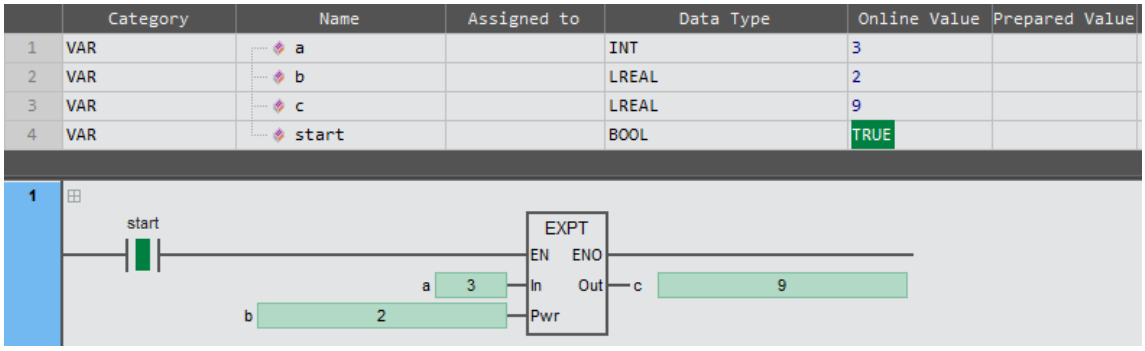
	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out														○	○					

***Note:** The ' ○ ' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- This instruction is used to calculate a power operation with ' In ' as the base and ' Pwr ' as the exponential, and output the result to ' Out ' .
- The example program is shown below.

LD:



ST:

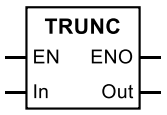
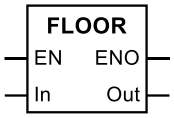
```
c: =EXPT ( a , b );
```

2.6.19 TRUNC/FLOOR（integer part of real numbers）

The TRUNC instruction is used to return the integer part of the input variable ' In ', discard the decimal part, and output the result to ' Out '.

The FLOOR instruction is used to return the integer part of the input variable ' In ' (treating positive and negative numbers differently), discard the decimal part, and output the result to ' Out '.

Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
TRUNC	Rounding real numbers	FUN		Out: = TRUNC(In);
FLOOR	Rounding real numbers and performing operations	FUN		Out: = FLOOR(In);

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Input value	Input	Input value	Depends on variable type
Out	Operation result	Output	The integer part after operation	Depends on variable type, nonnegative number

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														○	○					
Out													○							

***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

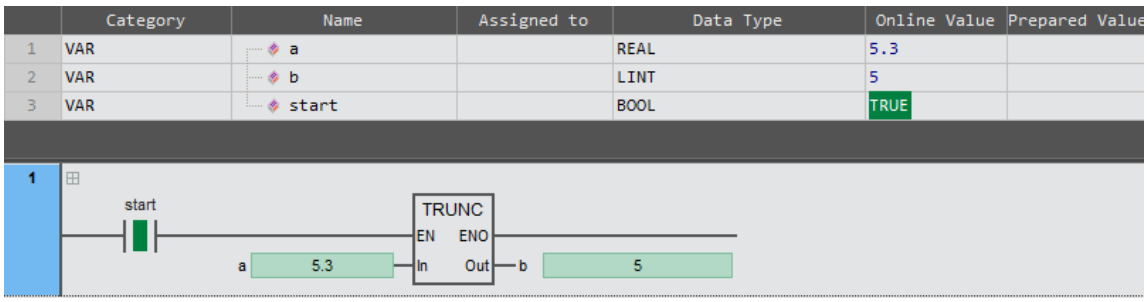
■ Function description

- The TRUNC instruction is used to return the integer part of the input variable ' In ', discard the decimal part, and output the result to ' Out '.
- The FLOOR instruction is used to return the integer part of the input variable 'In ' (treating positive and negative numbers differently), discard the decimal part, and output the result to ' Out '.

' In '	The value of ' Out ' in the TRUNC instruction	The value of ' Out ' in the FLOOR instruction
5.3	5	5
5	5	5
-5.3	-5	-6
-5	-5	-5

- The example program for the TRUNC instruction is as follows:

LD:

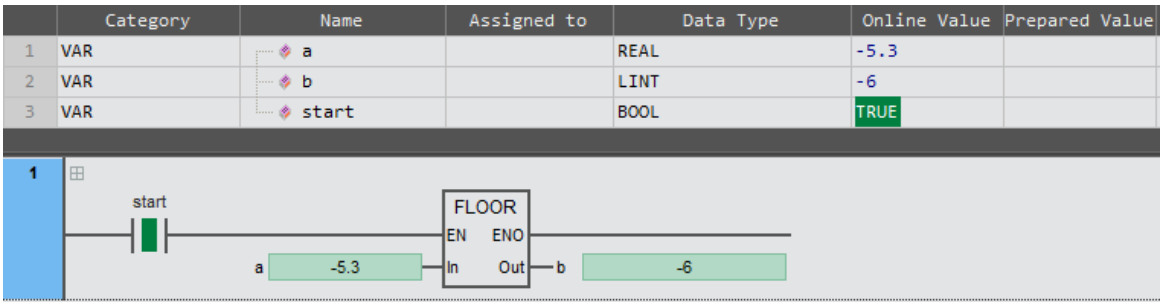


ST:

```
b: =TRUNC ( a );
```

- The example program for the FLOOR instruction is as follows:

LD:



ST:

```
b: =FLOOR ( a );
```

2.6.20 FRACTION（decimal parts of real numbers）

This instruction is used to return the decimal part of the input variable ' In ' and output the result to ' Out '.

Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
FRACTION	Decimal parts of real numbers	FUN	FRACTION EN ENO In Out	Out: = FRACTION(In);

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Input value	Input	Input value	Depends on variable type
Out	Operation result	Output	The decimal part after calculation	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														○	○					
Out														○	○					

***Note:** The ' ○ ' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

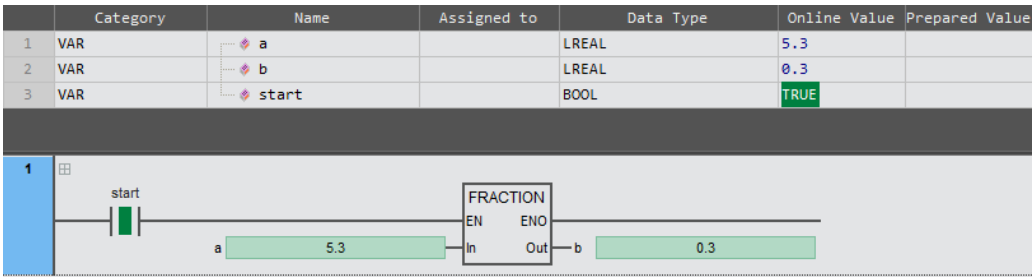
■ Function description

- This instruction is used to return the decimal part of the input variable ' In ' and output the result to ' Out '. The symbol for ' Out ' is consistent with the symbol for ' In '. For example, when the value of ' In ' is positive, the value of ' Out ' is positive; When the value of ' In ' is negative, the value of ' Out ' is negative.

' In '	' Out '
5.3	0.3
-5.3	-0.3

- The example program is shown below.

LD:



ST:

```
b: =FRACTION( a );
```

2.6.21 RAND（random number）

This instruction is used to generate a random number. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
RAND	Random number	FUN	RAND EN ENO In Out	Out: = RAND(In);

■ Input /output variable descriptions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Reserve	Input	Reserve	Depends on variable type
Out	Random number	Output	Random number	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In								○												
Out												○								

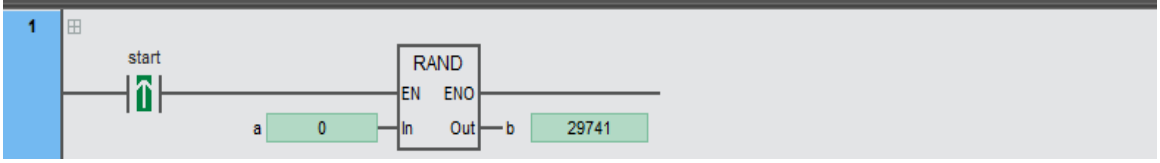
***Note:** The '○' in the above table indicates that the instruction parameter allows connection to variables or constants of that data type.

■ Function description

- This instruction is used to generate a random number in the range of 0-32767.
- The value of the input variable of this instruction has no effect on the number of output results.
- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		UDINT	0	
2	VAR	b		DINT	29741	
3	VAR	start		BOOL	TRUE	



ST:

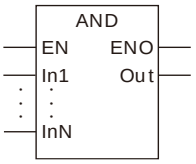
```
b: =RAND( a );
```

2.7 Logical operation instruction

2.7.1 AND (and)

This instruction is used for AND operations with multiple variables or constants, and the result is output as Out'.

Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
AND	And	FUN		Out: =In1 and In2 and …and InN; Out: =In1& In2 & …& InN;

■ Input/Output Variable instruction and data type

Name	Meaning	I/O	Description	Parameter scope
In1	Operation Object	Input	Operation Object	Depends on variable type
In2 to InN	Operation Object		The operation object can be increased or decreased through programming software when programming ladder diagrams, i.e. N=2 ~ 8	Depends on variable type
Out	Operation Result	Output	Operation Result	Depends on variable type

	Boolean	Bit String				Integer								Real number		Time，Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1to InN	○	○	○	○	○	○	○	○	○											
Out	○	○	○	○	○	○	○	○	○											

***Note:** The ' ○ ' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- This instruction is used for and operation of multiple variables or constants, and the result is output to 'Out', Out = In1 & In2 &…& InN. The operation rule of bitwise AND is: when the corresponding bits of the input variable are all TRUE, the corresponding bits of the output variable are TRUE; otherwise, it is FALSE.

  For example : 16#37 AND 16#42, result is 16#02。

- When the input variable type is bit string or integer, this instruction allows input variable 'In1' ~ 'InN' to be different data types When 'In1' ~ 'InN' are different data types, perform the operation using a data type that includes a range of values from 'In1' to 'InN', i.e. The data type of 'In1' is BYTE, 'In2' is DWORD type, then the OUT type is DWORD; During the operation, 'In1' is converted from BYTE to DWORD (Zero-padding), and then bitwise summed with 'In2'.
- If the input variable is BOOL type, then all input and output variables should be BOOL type, otherwise an error will be reported during software compilation.

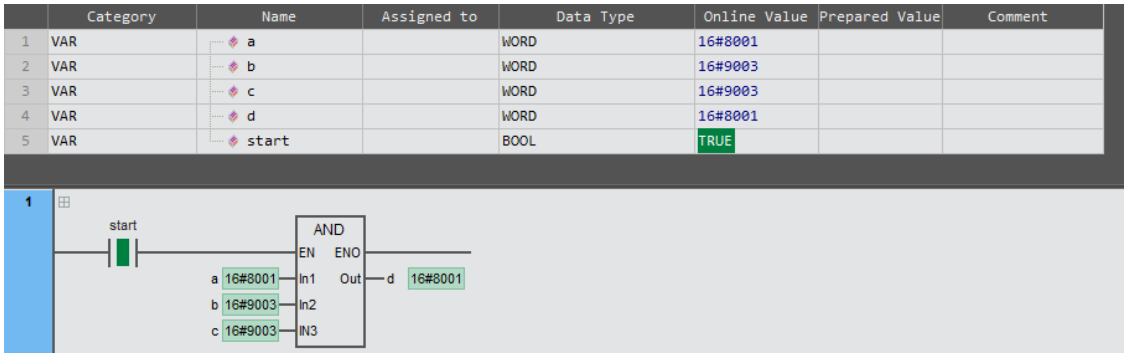
- The logical relationship between input and output operations is shown in the table below.

'In1'	'In2'	'Out'
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

- Example program 1 and schematic diagram are shown below.

LD:

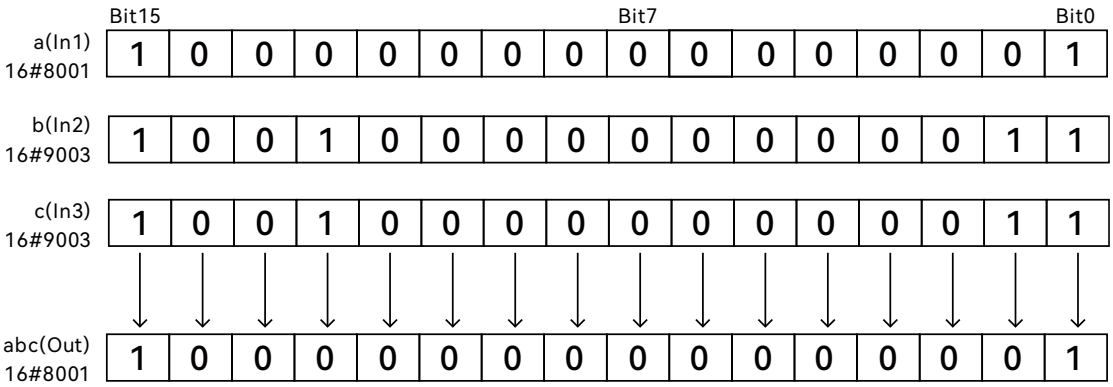
The following figure shows an example of adding three WORD type variables. The value of ' In1 ' is 16 # 8001, the value of ' In2 ' is 16 # 9003, and the value of ' In3 ' is 16 # 9003. After the instruction is executed, the value of ' Out ' is 16 # 8001. The execution process: 'In1' and 'In2' first perform AND operation, and the result of AND operation is then performed with ' In3 ', and the final result is output to ' Out '.



ST:

d: =a AND b AND c;

Schematic diagram

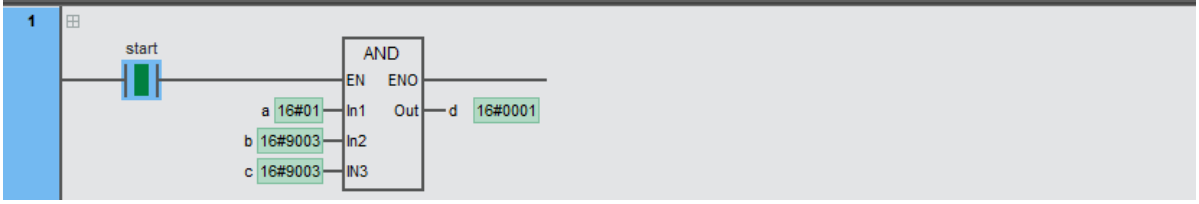


- Example program 2 and schematic diagram are shown below.

LD:

The following figure shows an example of a BYTE type variable and two WORD type variables. The value of ' In1 ' is 16 # 1, the value of ' In2 ' is 16 # 9003, and the value of ' In3 ' is 16 # 9003. After the instruction is executed, ' Out ' is 16 # 1. The execution process: 'In1' and 'In2' first perform AND operation, and the result of AND operation is then performed with ' In3 ', and the final result is output to ' Out '.

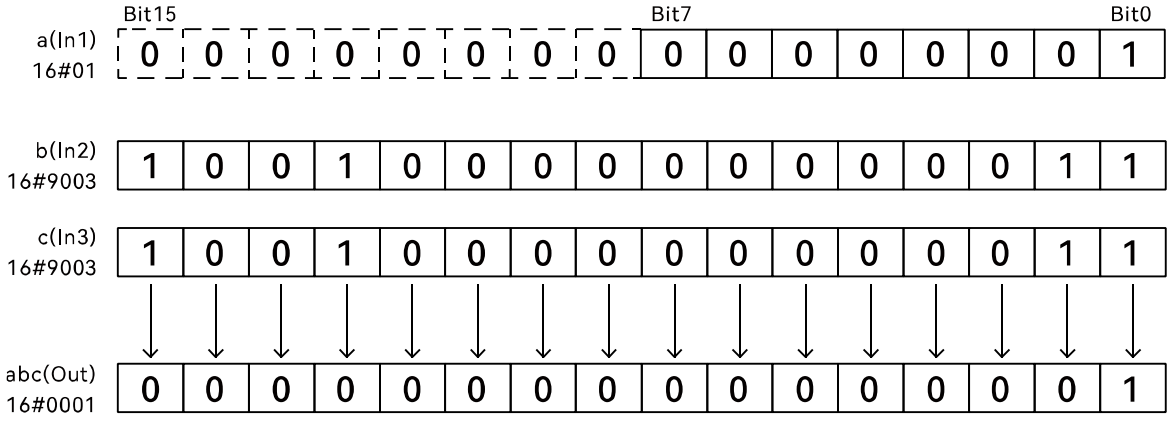
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		BYTE	16#01	
2	VAR	b		WORD	16#9003	
3	VAR	c		WORD	16#9003	
4	VAR	d		WORD	16#0001	
5	VAR	start		BOOL	TRUE	



ST:

d: =a AND b AND c;

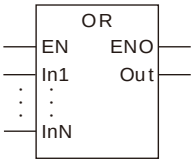
Schematic diagram:



2.7.2 OR（or）

This instruction is used for OR operations with multiple variables or constants, and the result is output to 'Out'.

Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
OR	or	FUN		Out: =In1 OR In2 OR …OR InN;

■ Input/Output variable instruction and data type

Name	Meaning	I/O	Description	Parameter scope
In1	Operation Object	Input	Operation Object	Depends on variable type
In2 to InN	Operation Object		The operation object can be increased or decreased through programming software when programming ladder diagrams, i.e. N=2 ~ 8	Depends on variable type
Out	Operation Result	Output	Operation Result	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1 to InN	○	○	○	○	○	○	○	○	○											
Out	○	○	○	○	○	○	○	○	○											

***Note:** The ' ○ ' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

Function description

- This instruction is used to perform a bitwise OR operation on two or more variables or constants, and the result is output to ' Out ', i.e., Out = In1 OR In2 OR…OR InN. The operation rule is that if any of the corresponding bits of the input variables are TRUE, the corresponding bit of the output variable is also TRUE; if all the corresponding bits of the input variables are FALSE, the corresponding bit of the output variable is FALSE. For example, 16#02 OR 16#03 results in 16#03.
- When the input variable type is Bit string or Integer, this instruction allows input variable 'In1' ~ 'InN' is to be different data types When 'In1' ~ 'InN' are different data types, perform the operation using a data type that includes a range of values from 'In1' to 'InN', i.e. The data type of 'In1' is BYTE, 'In2' is DWORD type, then the OUT type is DWORD; During the operation, 'In1' is converted from BYTE to DWORD (Zero-padding), and then bitwise summed with 'In2'.
- If the input variable is BOOL type, which requires that all input and output variables are BOOL type, otherwise an error will be reported during software compilation.

- The logical relationship between input and output operations is shown in the table below.

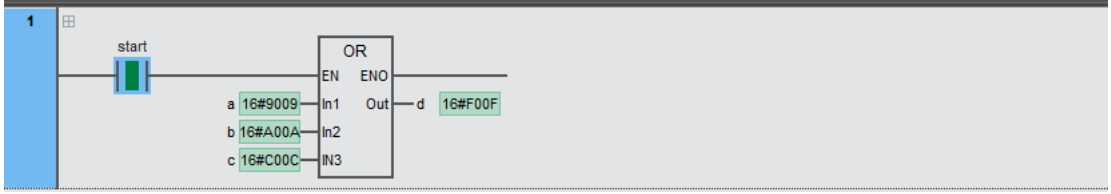
'In1'	'In2'	'Out'
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

- Example program 1 and schematic diagram are shown below.

LD:

The following figure shows an example of OR of three WORD-type variables. The value of 'In1 ' is 16 # 9009, the value of 'In2 ' is 16 # A00A, and the value of 'In3 ' is 16 # C00C, After the instruction is executed, the value of Out is 16 # F00F. The execution process: 'In1' and 'In2' first perform an OR operation, and the result of the OR operation is then performed with ' In3 ', and the final result is output to ' Out '.

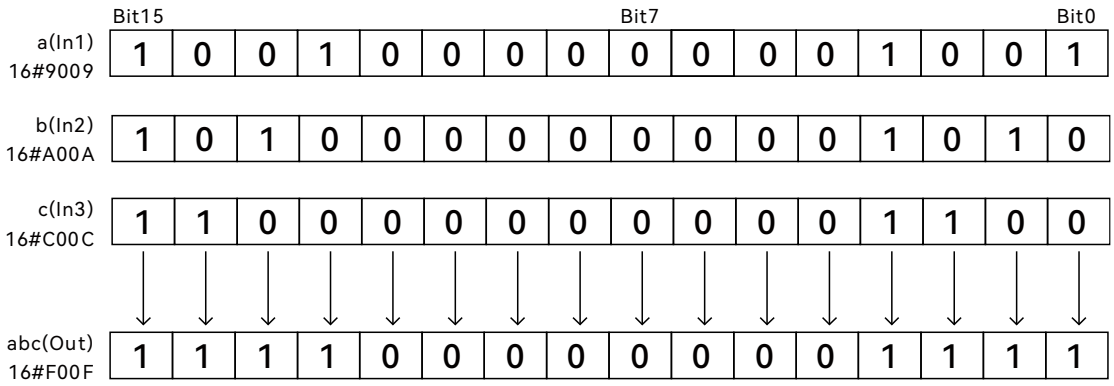
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		WORD	16#9009	
2	VAR	b		WORD	16#A00A	
3	VAR	c		WORD	16#C00C	
4	VAR	d		WORD	16#F00F	
5	VAR	start		BOOL	TRUE	



ST:

d: =a OR b OR c;

Schematic diagram:

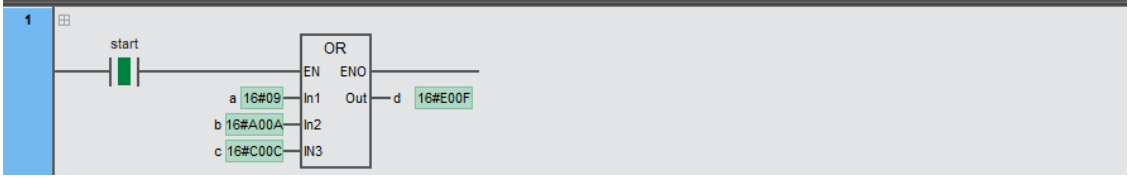


- Example program 2 and schematic diagram are shown below.

LD:

The following diagram shows an example of a BYTE type variable and two WORD type variables. The value of ' In1 ' is 16 # 9, the value of 'In2' is 16 # A00A, the value of 'In2' is 16 # C00C, and the value of 'Out' is 16 # E00F after the instruction is executed. The execution process: 'In1' and 'In2' first perform OR operation, and the result of OR operation is then performed with 'In3', and the final result is output to 'Out'.

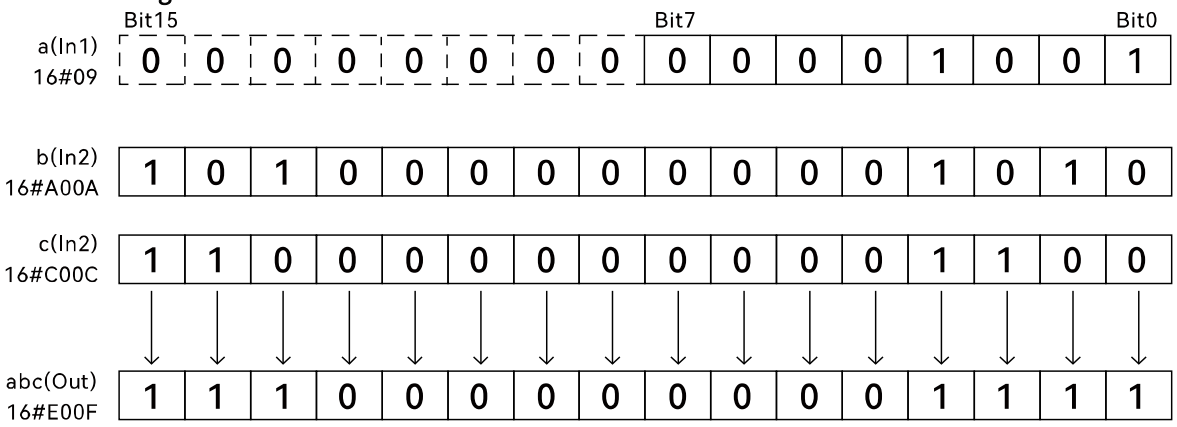
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value	Comment
1	VAR	a		BYTE	16#09		
2	VAR	b		WORD	16#A00A		
3	VAR	c		WORD	16#C00C		
4	VAR	d		WORD	16#E00F		
5	VAR	start		BOOL	TRUE		



ST:

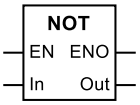
d: =a OR b OR c;

Schematic diagram:



2.7.3 NOT（negation）

This instruction is used to perform the operation of bitwise inversion on the variable or constant, and the result is output to 'Out'. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
NOT	Negation	FUN		Out: = NOT (In);

■ **Input / Output variable instructions and data types**

Name	Meaning	I/O	Description	Parameter scope
In	Operation Object	Input	Operation Object	Depends on variable type
Out	Operation Result	Output	Operation Result	Depends on variable type

	Boolean	Bit string				Integer								Real umber		Time，Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	○	○	○	○	○	○	○	○	○											
Out	○	○	○	○	○	○	○	○	○											

***Note:** The ' ○ ' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ **Function description**

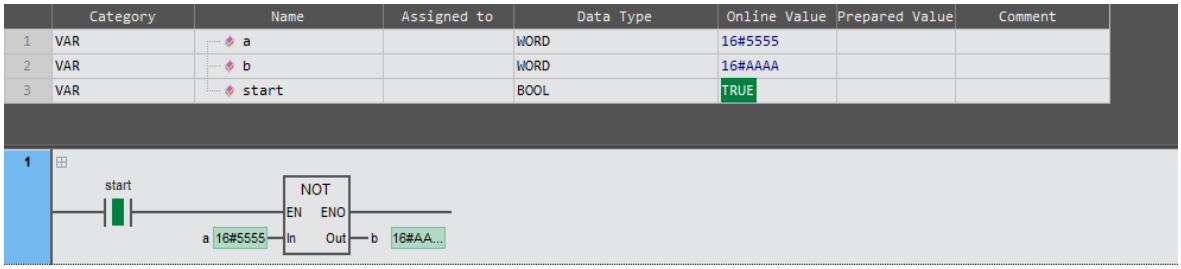
- This instruction is used to perform the operation of bitwise inversion on the variable or constant, and the result is output to 'Out', i.e., Out = NOT(In). The operation rule is to convert the TRUE bits of the input variable to FALSE, and the FALSE bits to TRUE, and then output the final result to 'Out'.
- The width of the 'Out' data type must be greater than or equal to the width of the 'In' data type, otherwise, a software compilation error will occur. Perform the operation using the data type of 'Out'. For example, if the data type of 'In' is BYTE and the data type of 'Out' is DWORD, then the data type of 'Out' will be DWORD. During the operation, first perform the inversion operation and then change 'In' from BYTE to DWORD (zero padding).
- If the input variable is BOOL type, which requires that all input and output variables are BOOL type, otherwise an error will be reported during software compilation.
- The logical relationship between input and output operations is shown in the table below.

'In'	'Out'
TRUE	FALSE
FALSE	TRUE

- Example program and schematic diagram are shown below.

LD:

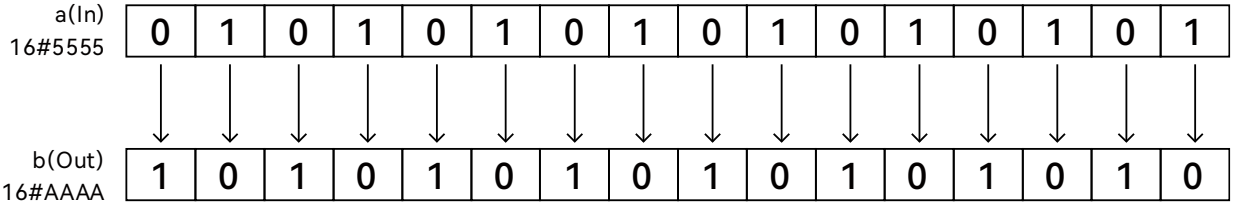
The following diagrams show an example of bitwise inversion for a WORD-type variable. The value of ' In ' is 16#5555. After the execution of this instruction, the value of ' Out ' is 16#AAAA..



ST:

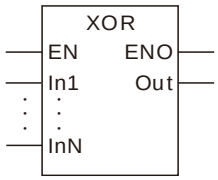
b: = NOT a;

Schematic Diagram:



2.7.4 XOR（exclusive OR）

This instruction is used to operate bitwise exclusive OR operation on multiple variables or constants, and the result is output to ' Out ' . Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
XOR	exclusive OR	FUN		Out = In1 XOR In2 XOR…XOR InN;

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Operation Object	Input	Operation Object	Depends on variable type
In2 to InN	Operation Object		The operation object can be increased or decreased through programming software when programming ladder diagrams, i.e. N=2 ~ 8	Depends on variable type
Out	Operation Result	Output	Operation Result	Depends on variable type

	Boolean	Bit String				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1 to InN	○	○	○	○	○	○	○	○	○											
Out	○	○	○	○	○	○	○	○	○											

***Note:** ' ○ ' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- This instruction is used to operate bitwise exclusive OR operation on multiple variables or constants, and the result is output to 'Out', i.e. Out = In1 XOR In2 XOR…XOR 'InN'. The operation rule is that if the values of the corresponding bits of the input variables are different, the corresponding bit of the output variable is TRUE; if the values of the corresponding bits of the input variables are the same, the corresponding bit of the output variable is FALSE.
- When the input variable type is a bit string or an integer, perform the operation using the data type that encompasses the entire range of values for 'In1' to 'InN'. For example, if the data type of 'In1' is WORD and the data type of 'In2' is DWORD, then the data type of 'Out' will be DWORD; During the operation, 'In1' is converted from WORD to DWORD (zero padding) and then bitwise exclusive OR operation is performed with ' In2 ' .
- If the input variable is BOOL type, which requires that all input and output variables are BOOL type, otherwise an error will be reported during software compilation.

- The logical relationship between input and output operations is shown in the table below.

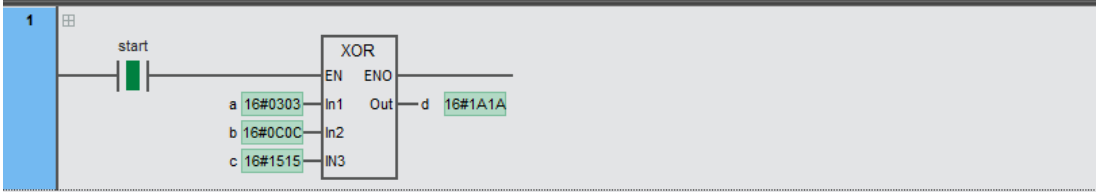
'In1'	'In2'	'Out'
TRUE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

- Example program 1 and schematic diagram are shown below.

LD:

The following figure shows an example of the exclusive OR operation of three WORD type variables. The value of ' In1 ' is 16#0303, the value of ' In2 ' is 16#0C0C, and the value of ' In3 ' is 16 # 1515. After the instruction is executed, the value of Out is 16#1A1A. Execution process: ' In1 ' and ' In2 ' perform XOR operations first, and the result of their XOR operation is then XOR with ' In3 ', and the final result is output to ' Out ' .

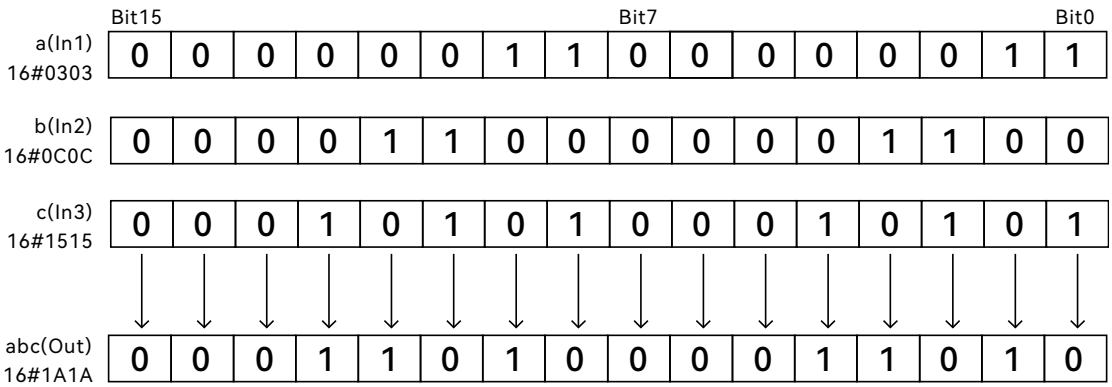
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		WORD	16#0303	
2	VAR	b		WORD	16#0C0C	
3	VAR	c		WORD	16#1515	
4	VAR	d		WORD	16#1A1A	
5	VAR	start		BOOL	TRUE	



ST:

d: =a XOR b XOR c;

Schematic diagram:

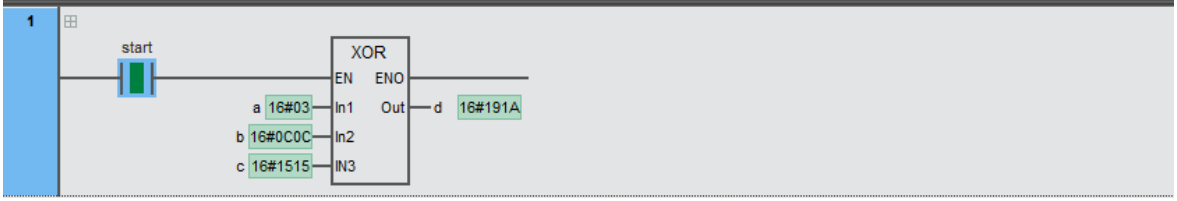


- Example program 2 and schematic diagram are shown below.

LD:

The following diagram shows an example of the exclusive OR of a variable of type BYTE and two variables of type WORD. The value of ' In1 ' is 16 # 03, the value of ' In2 ' is 16 # 0C0C, the value of ' In3 ' is 16 # 1515. After the instruction is executed, the value of ' Out ' is 16 # 191A. The execution process: ', 'In1' and 'In2' first perform XOR operation, and the result of XOR operation is then XOR operation with ' In3 ', and the final result is output to ' Out '.

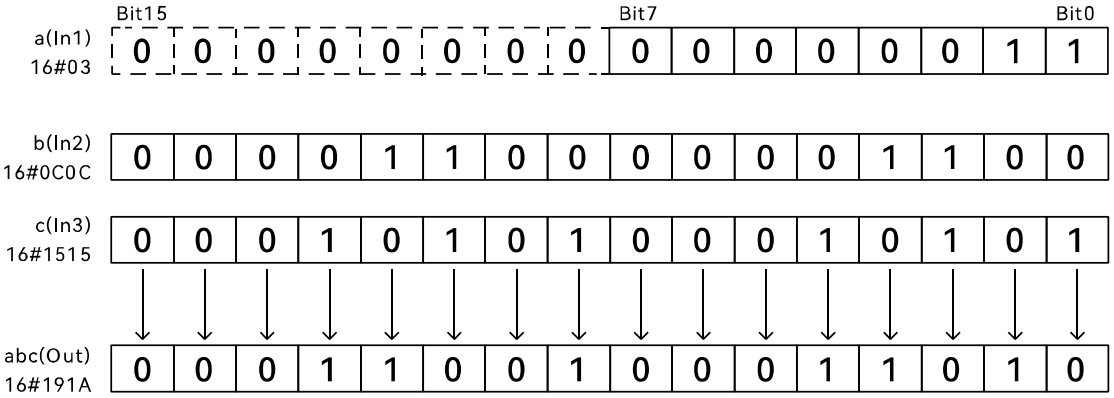
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		BYTE	16#03	
2	VAR	b		WORD	16#0C0C	
3	VAR	c		WORD	16#1515	
4	VAR	d		WORD	16#191A	
5	VAR	start		BOOL	TRUE	



ST:

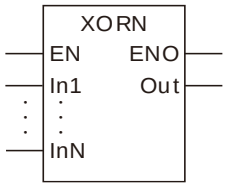
d: =a XOR b XOR c;

Schematic diagram:



2.7.5 XORN（exclusive NOR）

This instruction is used to operate bitwise exclusive NOR operation on multiple variables or constants, and the result is output to ' Out ' . Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
XORN	Exclusive NOR	FUN		Out =In1 XORN In2 XORN⋯XORN InN;

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter Scope
In1	Operation Object	Input	Operation Object	Depends on variable type
In to InN	Operation Object		The operation object can be increased or decreased through programming software when programming ladder diagrams, i.e. N=2 ~ 8	Depends on variable type
Out	Operation Result	Output	Operation Result	Depends on variable type

	Boolean	Bit String				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1 to InN	○	○	○	○	○	○	○	○	○											
Out	○	○	○	○	○	○	○	○	○											

***Note:** '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- This instruction is used to operate bitwise exclusive NOR operation on multiple variables or constants, and the result is output to ' Out ' . i.e., Out = In1 XORN In2 XORN⋯XORN InN. The operation rule is that when the corresponding bits of the input variables are all the same, the corresponding bits of the output variables are TRUE, and when the corresponding bits of the input variables are different, the corresponding bits of the output variables are FALSE.
- When the input variable type is a bit string and integer, the input variable ' In1 ' ~ ' InN ' data type is allowed to be different. When the data types of 'In1' ~ 'InN' are different, the data types including the value scope of ' In1 ' ~ ' InN ' are operated. For example, the ' In1 ' data type is WORD, the ' In2 ' data type is DWORD, then the ' Out ' data type is DWORD; in the operation, ' In1 ' is converted from WORD to DWORD (zero padding), and then the XORN operation is performed with ' In2 '.
- If the input variable is BOOL type, which requires that all input and output variables are BOOL type, otherwise an error will be reported during software compilation.

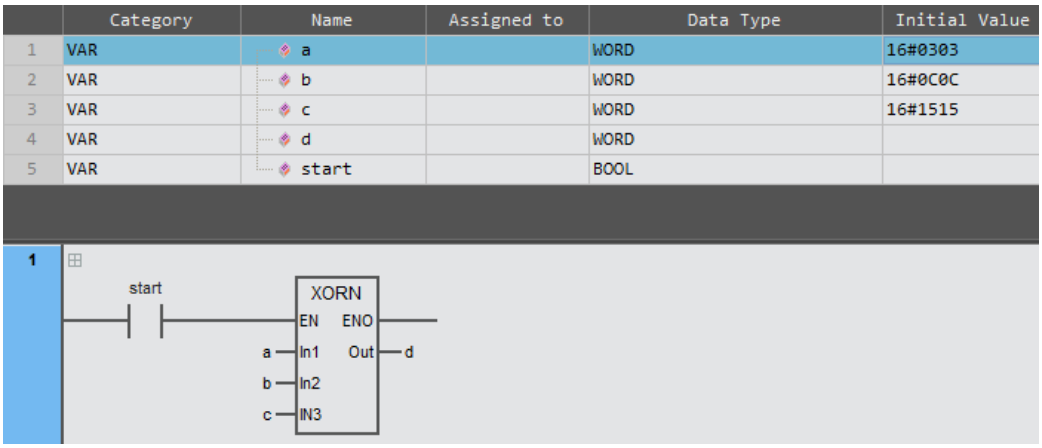
- The logical relationship between input and output operations is shown in the table below.

'In1'	'In2'	'Out'
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	TRUE

- Example program 1 and schematic diagram are shown below.

LD:

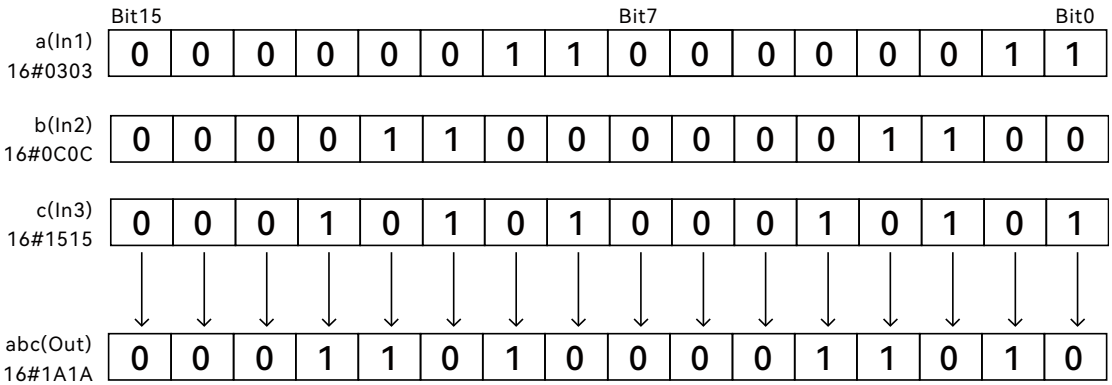
The following figure shows the Exclusive NOR operation of three WORD type variables. The value of ' In1 ' is 1 #0303, the value of ' In2 ' is 16#0C0C, the value of ' In3 ' is 16#1515, and the value of 'Out' is 16#1A1A after the instruction is executed. The execution process: ' In1 ' and ' In2 ' first perform the XORN operation, and the result of the XORN operation is then performed with ' In3 ', and the final result is output to ' Out '.



ST:

d: =a XORN b XORN c;

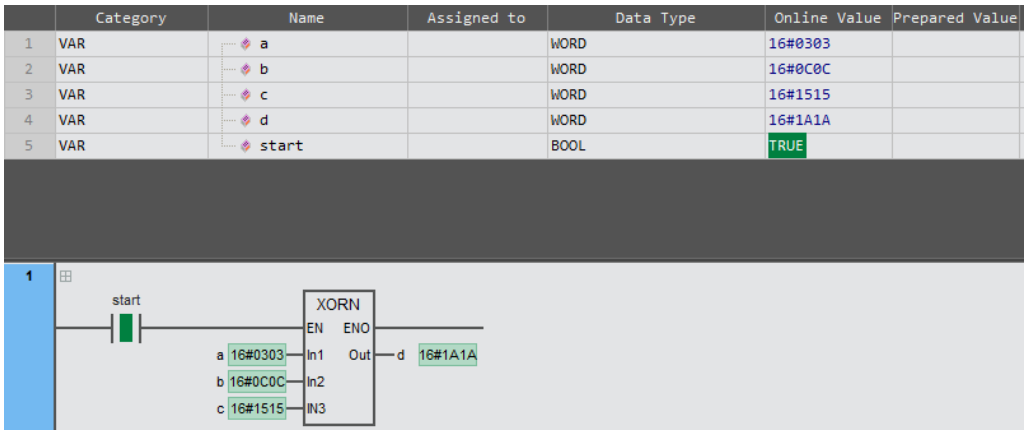
Schematic diagram:



- Example program 2 and schematic diagram are shown below.

LD:

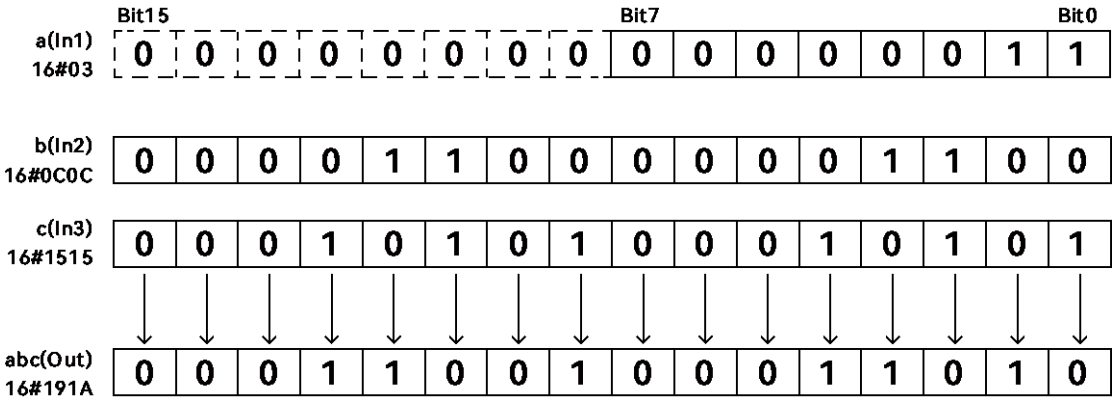
The following figure shows the Exclusive NOR operation of two WORD-type variables and a BYTE type variable. The value of ' In1 ' is 16 # 03, the value of ' In2 ' is 16 # 0C0C, and the value of ' In3 ' is 16 # 1515. After the instruction is executed, the value of Out is 16 # 191A. The execution process: ' In1 ' and ' In2 ' first perform the XORN operation, and the result of the XORN operation is then performed with ' In3 ', and the final result is output to ' Out '.



ST:

d: =a XORN b XORN c;

Schematic diagram:

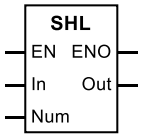
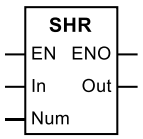


2.8 Data shift

2.8.1 SHL（shift to the left）/ SHR（shift to the right）

SHL（shift to the left）: Shift all bits of the input variable or constant from right to left by the specified number of bits (toward the higher bits)， and the resulting output to 'Out'.

SHR（shift to the right）: Shift all bits of the input variable or constant from left to right by the specified number of bits (toward the lower bits)，and the resulting output to 'Out'. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
SHL	Shift to the left	FUN		Out: =SHL(In , Num);
SHR	Shift to the right	FUN		Out: =SHR(In , Num);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data to shift	Input	Data to shift	Depends on variable type
Num	Number to shift		Number to shift	Depends on variable type
Out	Processing result	Output	Processing result	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○											
Num						○														
Out		○	○	○	○	○	○	○	○											

***Note:** The ' ○ ' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

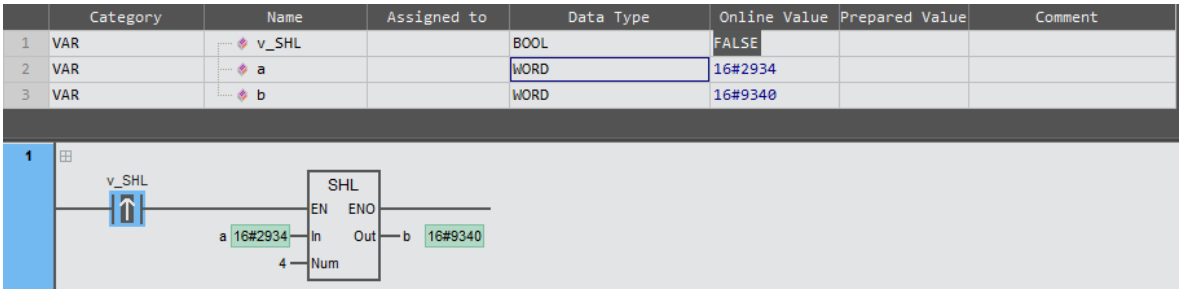
■ Function description

- SHL (shift to the left): Shift all bits of the input variable or constant from right to left (from low to high) by the number of bits specified by ' Num ' . The value of the rightmost ' Num ' bit becomes 0, and the shifted result is output as ' Out ' .
- SHR（shift to the right）: Shift all bits of the input variable or constant from left to right (from high to low) by the number of bits specified by ' Num ' . The value of the leftmost ' Num ' bit becomes 0, and the shifted result is output as ' Out ' .

- The SHL example program and schematic diagram are shown below.

LD:

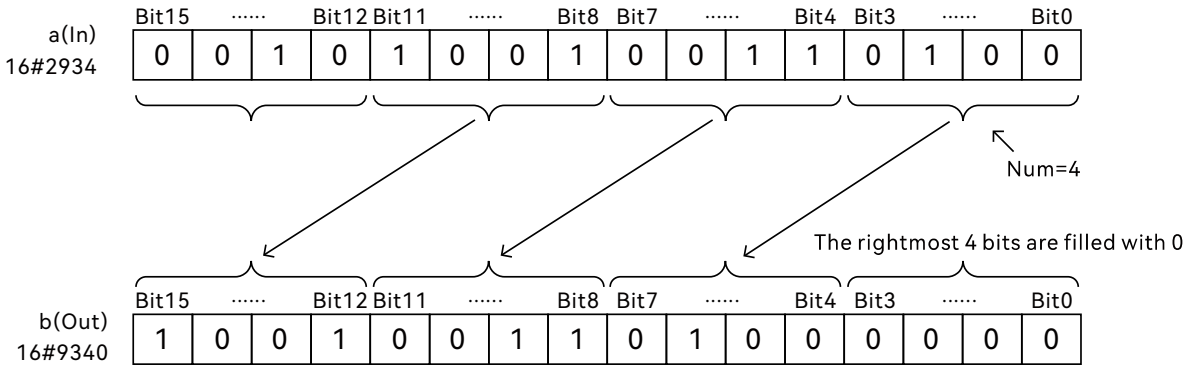
The following figures show an example of shifting a variable of WORD type to the left. When the v _ SHL variable changes from FALSE to TRUE, the SHL instruction is executed once. The value of ' In1 ' is 16#2934, the value of ' Num ' is 4, and the value of ' Out ' is 16 # 9340 after the instruction is executed.



ST:

```
IF   EDGEPOS(v_SHL)   THEN
      b: =SHL( a , 4 );
END_IF;
```

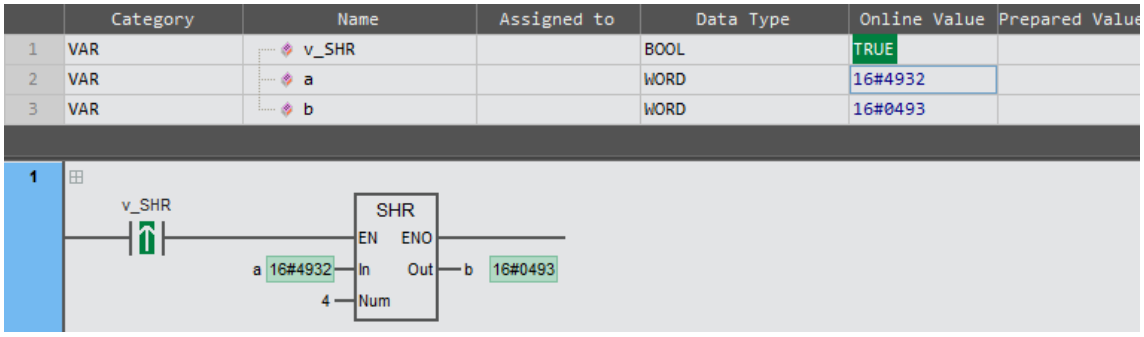
Schematic diagram:



- SHR Example program and schematic diagram are shown below.

LD:

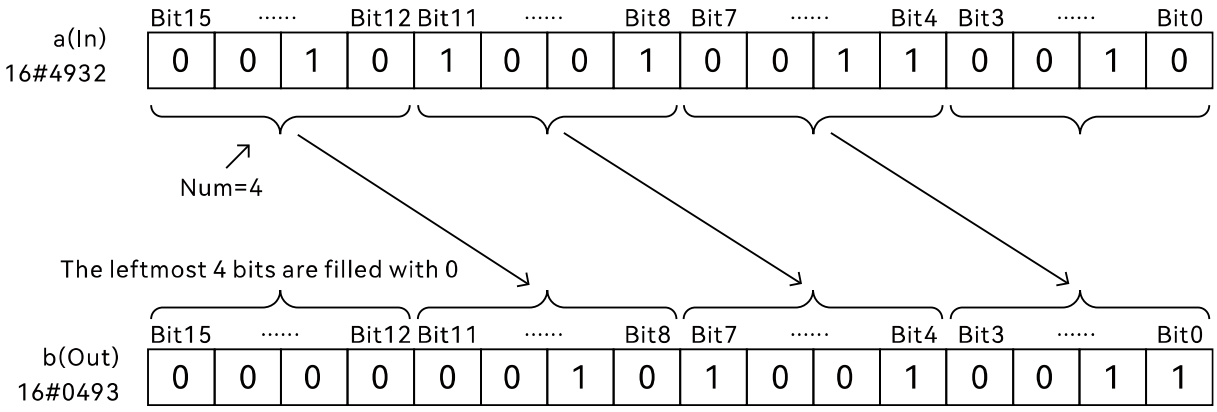
The following figure shows an example of shifting a variable of WORD type to the right. When the v_SHR variable changes from FALSE to TRUE, the SHR instruction is executed once.The value of ' In1 ' is 16#4932, the value of ' Num' is 4, and the value of ' Out ' is 16 # 0493 after the instruction is executed.



ST:

```
IF   EDGEPOS(v_SHR)   THEN
      b: =SHR( a , 4);
END_IF;
```

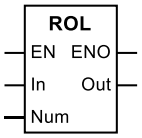
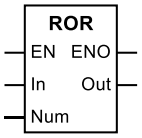
Schematic diagram :



2.8.2 ROL（rotate left）/ ROR（rotate right）

ROL(rotate left): Circularly shifts all bits of the input variable or constant from right to left according to a specified number of bits, the result output to 'Out'.

ROR(rotate right): Circularly shifts all bits of the input variable or constant from left t to right according to a specified number of bits, the result output to 'Out'. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
ROL	Rotate left	FUN		Out: =ROL(In, Num);
ROR	Rotate right	FUN		Out: =ROR(In , Num);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data to shift	Input	Data to shift	Depends on variable type
Num	Number to shift		Number to shift	Depends on variable type
Out	Processing result	Output	Processing result	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○	○	○	○	○											
Num						○														
Out		○	○	○	○	○	○	○	○											

***Note**: The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

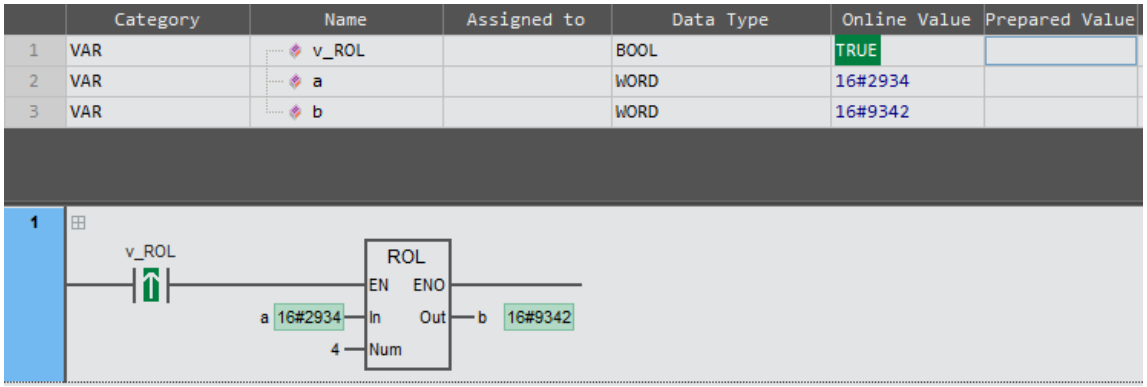
■ Function description

- ROL（rotate left）: Circularly shift all bits of the input variable or constant from right to left (from low to high) according to the number of bits specified by ' Num '. The value of the rightmost ' Num ' bit becomes the value of the leftmost ' Num ' bit, and the result is output to ' Out ' after shifting.
- ROR（rotate right）: Circularly shift all bits of the input variable or constant from left to right (from high to low) according to the number of bits specified by ' Num '. The value of the leftmost ' Num ' bit becomes the value of the rightmost ' Num ' bit, and the result is output to ' Out ' after shifting.

- The ROL example program and schematic diagram are shown below.

LD:

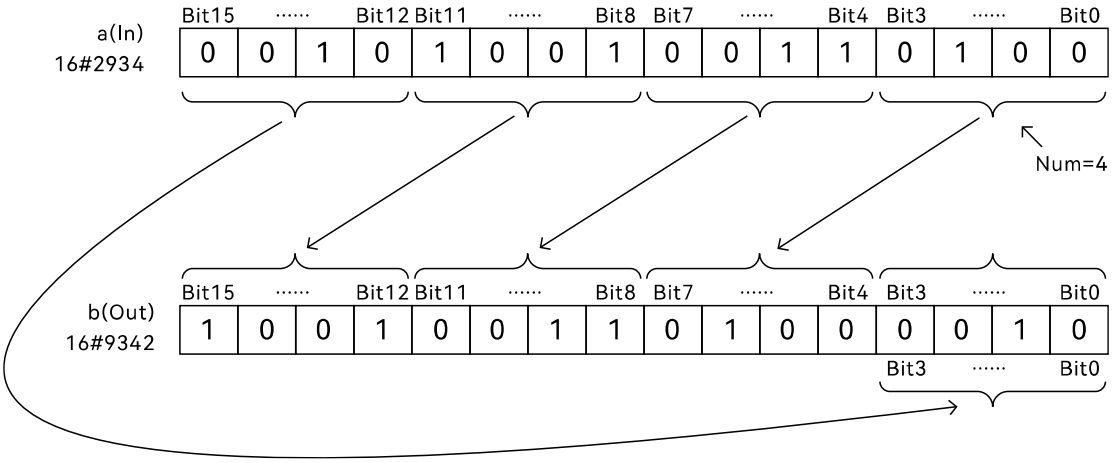
The following figure shows an example of a variable of WORD type circularly shifting to the left. When the v_ROL variable changes from FALSE to TRUE, the ROL command is executed once. The value of 'In1' is 16 # 2934, the value of 'Num' is 4, and the value of 'Out' is 16#9342 after the instruction is executed.



ST:

```
IF   EDGEPOS(v_ROL)   THEN
      b: =ROL ( a , 4 );
END_IF;
```

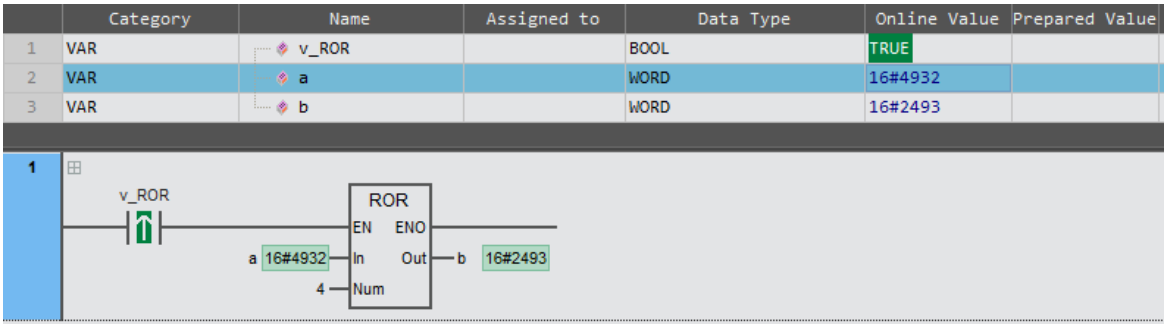
Schematic diagram:



- ROR Example program and schematic diagram are shown below.

LD:

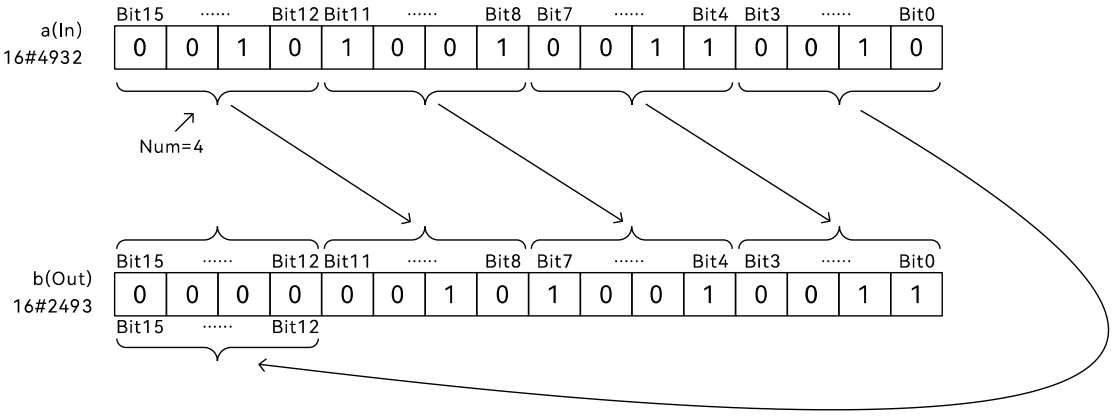
The following figure shows an example of a variable of WORD type circularly shifting to the right. When the v_ROR variable changes from FALSE to TRUE, the ROR command is executed once. The value of 'In1' is 16#4932, the value of 'Num' is 4, and the value of 'Out' is 16#2493 after the instruction is executed.



ST:

```
IF EDGEPOS(v_ROR)    THEN
      b: =ROR ( a , 4 );
END_IF;
```

Schematic diagram:



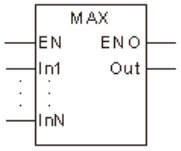
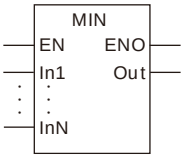
2.9 Selection operations

2.9.1 MAX/MIN（maximum/minimum）

MAX(maximum): Find the maximum value among multiple variables or constants, and output the result to 'Out'.

MIN(minimum): Find the minimum value among multiple variables or constants, and output the result to 'Out'.

Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
MAX	maximum	FUN		Out: =MAX(In1 , In2);
MIN	minimum	FUN		Out: =MIN(In1 , In2);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Comparison Object	Input	Comparison Object	Depends on variable type
In2 to InN	Comparison Object		The comparison object can be increased or decreased through programming software when programming ladder diagrams, i.e. N=2 ~ 8	Depends on variable type
Out	Maximum	Output	Maximum Value	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In2 to InN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Out	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

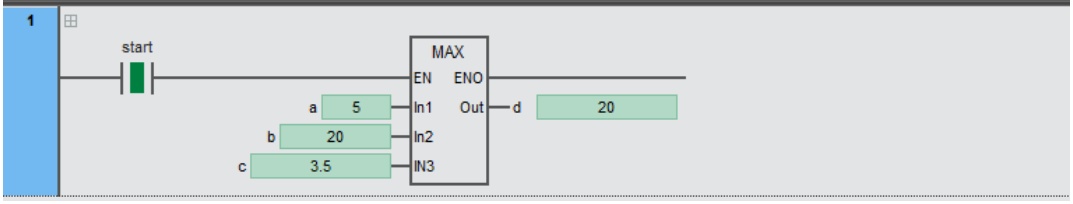
- MAX(maximum): This instruction is used to compare and find the maximum value among In1~InN variables or constants, and output the result to 'Out'.
- MIN(minimum): This instruction is used to compare and find the minimum value among In1~InN variables or constants, and output the result to 'Out'.
- When the data types of variables from 'In1' to 'InN' and 'Out' are bit strings, integers, or real numbers, different types of variables from 'In1' to 'InN' and 'Out' are allowed. During the operation, data types that can contain all values of 'In1' to 'InN' and 'Out' are used. If the data type of 'In1' is INT, the data type of 'In2' is INT, and the data type of 'Out' is DINT, then the operation is processed using the DINT data type.

- The scope of the ' Out ' data type must be greater than or equal to the scope of types ' In1 ' to ' In2 ', otherwise an error will be reported during software compilation. If the data types of ' In1 ' and ' In2 ' are INT and DINT respectively, then the data types of ' Out ' must be DINT, LINT, etc.; If the data type of the variable connected to ' Out ' is SINT, INT, etc., an error will be reported during software compilation.
- When the input variable is BOOL, TIME, DATE, TOD, DT or STRING data type, it is required that ' In1 ' ~ ' InN ' and ' Out ' are all of the same type of data type. If the data type of ' In1 ' is DATE, the data types of ' In2 ' and ' Out ' must also be DATE, otherwise an error will be reported during software compilation.

- The MAX example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		INT	5	
2	VAR	b		DINT	20	
3	VAR	c		REAL	3.5	
4	VAR	d		REAL	20	
5	VAR	start		BOOL	TRUE	



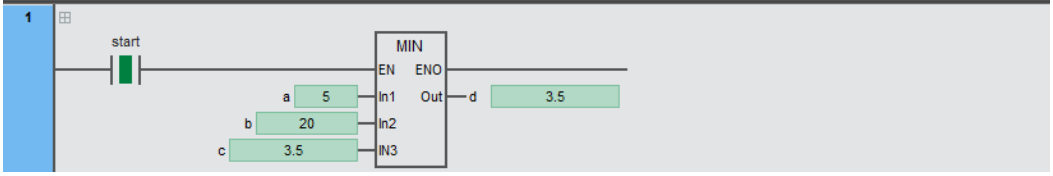
ST:

```
d: =MAX( MAX( a , b ) , c );
```

- The MIN example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		INT	5	
2	VAR	b		DINT	20	
3	VAR	c		REAL	3.5	
4	VAR	d		REAL	3.5	
5	VAR	start		BOOL	TRUE	

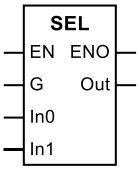


ST:

```
d: =MIN( MIN ( a , b ) , c );
```

2.9.2 SEL（bit selection）

The SEL instruction selects one of two options. Library: Standard

Instruction	Name	FB/FUN	ST expression	ST expression
SEL	Binary Selection	FUN		Out: =SEL(G , In0 , In1);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
G	Gate	Input	FALSE: Selects 'In0' TRUE: Selects 'In1'	FALSE or TRUE
In0、In1	Selection		Selection	Depends on variable type
Out	Selection result	Output	Selection result	Depends on variable type

	Boolean	Bit String				Integer								Real Number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
G	○																			
In0、In1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Out	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

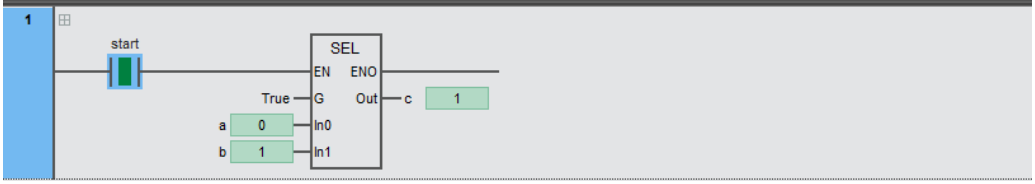
■ Function description

- The SEL instruction selects one of two options. The output value is decided by the input of 'G', If 'G' = FALSE, 'Out' = 'In0'; If 'G' =TRUE， 'Out' = 'In1'.
- When the data types of 'In0', 'In1', and 'Out' variables are bit strings, integers, or real numbers, 'In0', 'In1', and 'Out' are allowed to be different types of variables. During the operation, the data type that can contain all values of 'In0', 'In1', and 'Out' should be used. If the data type of 'In0' is INT, the data type of 'In1' is INT, and the data type of 'Out' is DINT, then the operation is processed using the DINT data type.
- The scope of the 'Out' data type must be greater than or equal to the scope of the 'In0' and 'In1' types, otherwise an error will be reported during software compilation. If the data types of 'In0' and 'In1' are INT and DINT respectively, then the data types of 'Out' must be DINT, LINT, etc.; If the data type of the variable connected to 'Out' is SINT, INT, etc., an error will be reported during software compilation.
- When the data type of the input variable is BOOL, TIME, DATE, TOD, DT or STRING, it is required that 'In0', 'In1', and 'Out' all have the same type of data type. If the data type of 'In0' is DATE, then the data types of 'In1' and 'Out' must also be DATE, otherwise an error will be reported during software compilation.

- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		INT	0	
2	VAR	b		INT	1	
3	VAR	c		INT	1	
4	VAR	start		BOOL	TRUE	

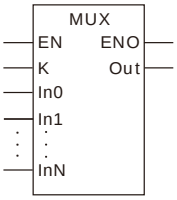


ST:

c: =SEL (TRUE , a , b);

2.9.3 MUX（multiplexer）

The MUX instruction selects one of two to five options. Library: Standard.

Instruction	Name	FB/FUN	ST expression	ST expression
MUX	multiplexer	FUN		Out: =MUX(K , In0 , In1);

■ **Input / Output variable instructions and data types**

Name	Meaning	I/O	Description	Parameter scope
K	Selector	Input	Selection option	Depends on variable type
In0 to InN	Selections		The selections can be increased or decreased through programming software when programming ladder diagrams, i.e. N=0 ~ 7	Depends on variable type
Out	Selection result	Output	Selection result	Depends on variable type

	Boolean	Bit string				Integer								Real Number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
K						○														
In0 to InN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Out	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ **Function description**

- Move the value of a specified input variable or constant to 'Out' by selecting a condition, which input variable to choose is determined by the selection option 'K'. The details are shown in the table below.

'K'	'Out'
0	In0（Out: =In0)
1	In1（Out: =In1)
2	In2（Out: =In2)
3	In3（Out: =In3)
4	In4（Out: =In4)
5	In5（Out: =In5)
6	In6（Out: =In6)
7	In7（Out: =In7)

- When the data types of variables from 'In0' to 'InN' and 'Out' are bit strings, integers, or real numbers, different types of variables from 'In0' to 'InN' and 'Out' are allowed. During the operation, data types that can include all value scopes

from 'In0' to 'InN' and 'Out' are used. If the data type of 'In0' is INT, the data type of 'In1' is INT, and the data type of 'Out' is DINT, then the operation is processed using the DINT data type.

- The scope of the 'Out' data type must be greater than or equal to the scope of the 'In0' to 'InN' types, otherwise an error will be reported during software compilation. If the data types of 'In0' and 'In1' are INT and DINT respectively, then the data types of 'Out' must be DINT, LINT, etc.; If the data type of 'Out' is SINT, INT, etc., an error will be reported during software compilation.
- l When the data type of the input variable is BOOL, TIME, DATE, TOD, DT, or STRING, it is required that 'In0'~'InN' and 'Out' all belong to the same type of data type. If the data type of 'In1' is DATE, the data types of 'In2' and 'Out' must be DATE, otherwise an error will be reported during software compilation.
- The example program is shown below.

LD:

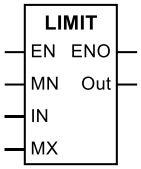
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	start		BOOL	TRUE	
2	VAR	k		USINT	1	
3	VAR	a		INT	0	
4	VAR	b		INT	6	
5	VAR	f		INT	6	

ST:

```
f: =MUX (k , a , b);
```

2.9.4 LIMIT（upper and lower limits）

This instruction limits the value of an input variable between the specified minimum and maximum values. Library: Standard

Instruction	Name	FB/FUN	ST expression	ST expression
LIMIT	upper and lower limits	FUN		Out: =LIMIT(MN , IN , MX);

■ **Input / Output variable instructions and data types**

Name	Meaning	I/O	Description	Parameter scope
MN	Minimum value	Input	Minimum value	Depends on variable type
IN	Data to limit		Data to limit	Depends on variable type
MX	Maximum value		Maximum value	Depends on variable type
Out	Processing result	Output	Processing result	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MN		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
In		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
MX		○	○	○	○	○	○	○	○	○	○	○	○	○	○					
Out		○	○	○	○	○	○	○	○	○	○	○	○	○	○					

***Note:** The ' ○ ' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ **Function description**

- This instruction limits the value of an input variable between the specified minimum and maximum values, the details are shown in the table below.

' In '	' Out '
' IN ' < ' MN '	' MN '
' MN ' ≤ ' IN ' ≤ ' MX '	' IN '
' MX ' < ' IN '	' MX '

- This instruction allows input and output variables to be different types, and the operation is performed using a data type that can contain all value ranges. For example, if the data type of ' MN ' is REAL, the data type of ' IN ' is REAL, the data type of ' MX ' is REAL, and the data type of ' Out ' is LREAL, the operation will be processed using the LREAL data type.
- The scope of the ' Out ' data type must be greater than or equal to the scope of the input variable data type, otherwise an error will be reported during software compilation. For example, if the data types for ' MN ', ' IN ', and ' MX ' are LREAL, REAL, and

REAL respectively, then the data type for ' Out ' must be LREAL; If the data type of the variable connected to ' Out ' is REAL, etc., an error will be reported during software compilation

- The example program is shown below.

LD:

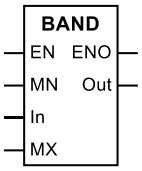
	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		INT	5	
2	VAR	b		INT	18	
3	VAR	c		INT	10	
4	VAR	d		INT	10	
5	VAR	start		BOOL	TRUE	

ST:

d: =LIMIT(a , b , c);

2.9.5 BAND（dead band limit）

The Band instruction performs dead band control. Library: Standard

Instruction	Name	FB/FUN	ST expression	ST expression
BAND	Dead band limitation	FUN		Out: =BAND(MN , In , MX);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
MN	Minimum value	Input	Minimum value	Depends on variable type
In	Data to control		Data to control	Depends on variable type
MX	Maximum value		Maximum value	Depends on variable type
Out	Processing result	Output	Processing result	Depends on variable type

	Boolean	Bit string				Integer								Real Number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MN														○	○					
In														○	○					
MX														○	○					
Out														○	○					

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- The Band instruction performs dead band control; the details are shown in the table below.

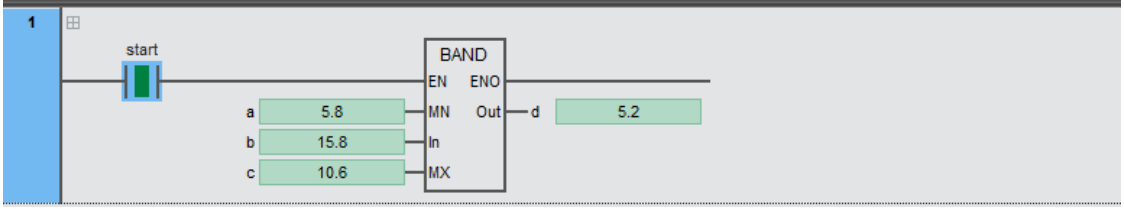
The value of 'In'	The value of 'Out'
'In' < 'MN'	'In' - 'MN'
'MN' ≤ 'In' ≤ 'MX'	0
'MX' < 'IN'	'In' - 'MX'

- This instruction allows input and output variables to be different types, and the operation is performed by using a data type that can contain all value ranges. For example, if the data type of 'MN' is REAL, the data type of 'In' is REAL, the data type of 'MX' is REAL, and the data type of 'Out' is LREAL, the operation will be processed by using the LREAL data type.
- The scope of the 'Out' data type must be greater than or equal to the scope of the input variable data type, otherwise an error will be reported during software compilation. For example, if the data types for 'MN', 'In', and 'MX' are LREAL, REAL, and REAL respectively, then the data type for 'Out' must be LREAL; If the data type of the variable connected to 'Out' is REAL, etc., an error will be reported during software compilation.

- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		REAL	5.8	
2	VAR	b		REAL	15.8	
3	VAR	c		REAL	10.6	
4	VAR	d		REAL	5.2	
5	VAR	start		BOOL	TRUE	

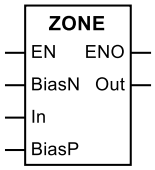


ST:

d: =BAND(a , b , c);

2.9.6 ZONE（input bias）

The Zone instruction adds a bias value to the input value. Library: Standard

Instruction	Name	FB/FUN	ST expression	ST expression
ZONE	Input bias	FUN		Out: =ZONE(BiasN , In1 , BiasP);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
BiasN	Negative bias	Input	Negative bias	Depends on variable type
In	Data to control		Data to control	Depends on variable type
BiasP	Positive bias		Positive bias	Depends on variable type
Out	Processing result	Output	Processing result	Depends on variable type

	Boolean	Bit String				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
BiasN														○	○					
In														○	○					
BiasP														○	○					
Out														○	○					

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- The Zone instruction adds a bias value to the input value，the details are shown in the table below.

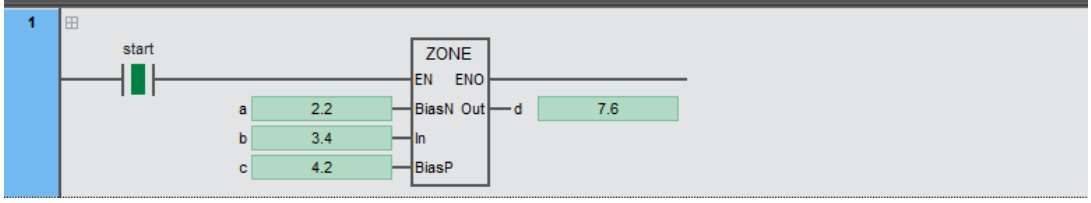
'In'	'Out'
'In' < 0	'In' + 'BiasN'
'In' = 0	0
'In' > 0	'In' + 'BiasP'

- This instruction allows input and output variables to be different types, and the operation is performed by using a data type that can contain all value ranges. If the data type of 'BiasN' is REAL, the data type of 'IN' is REAL, the data type of 'BiasP' is LREAL, and the data type of 'Out' is LREAL, then the operation will be processed by using the LREAL data type.
- The scope of the 'Out' data type must be greater than or equal to the scope of the input variable data type, otherwise an error will be reported during software compilation. For example, if the data types of 'BiasN', 'In', and 'BiasP' are REAL, REAL, and LREAL, respectively, then the data type of 'Out' must be LREAL; If the 'Out' data type is REAL, an error will be reported during software compilation.

- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	a		REAL	2.2	
2	VAR	b		REAL	3.4	
3	VAR	c		REAL	4.2	
4	VAR	d		REAL	7.6	
5	VAR	start		BOOL	TRUE	



ST:

d: =ZONE(a,b, c);

2.10 Data type conversion

2.10.1 BOOL_TO_*** (conversion of BOOL to other data types)

This instruction is used to convert bool type to other data types. Library: Standard

Instruction	Name	FB/FUN	Graphic expression	ST expression
BOOL_TO_***	BOOL conversion instruction	FUN		Out: = BOOL_TO_***(In); ***Represents different data type, i.e., Out: =BOOL_TO_INT(In);etc.

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data to convert	Input	Data to convert	FALSE OR TRUE
Out	Conversion result	Output	Conversion result	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	○																			
Out	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note :** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

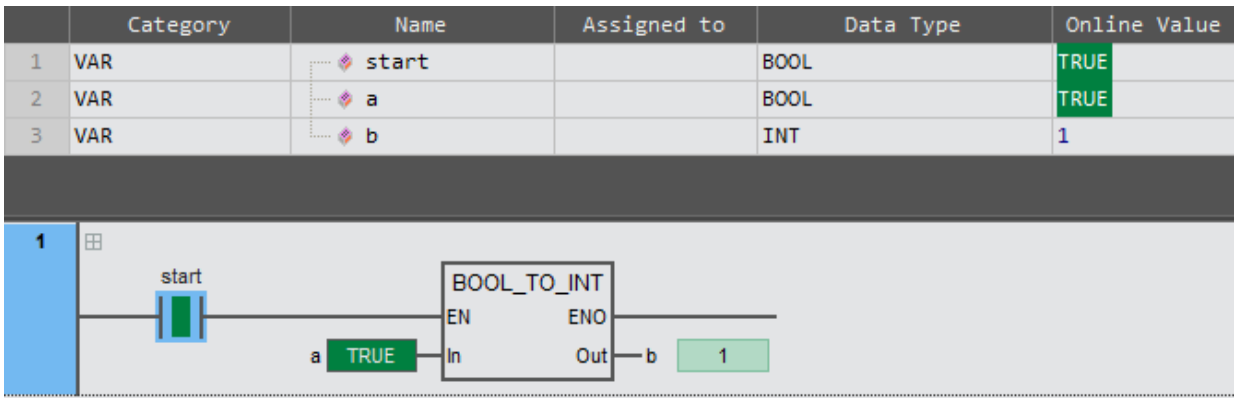
- This instruction is used to convert bool type to other data types.
- The instruction name varies depending on the data type of 'Out'. For example, if 'Out' is of INT type, the instruction name is BOOL_ TO_ INT.
- The correspondence between Bool data type and other data types is shown in the table below.

Data type		The data correspondence between 'In' and 'Out'	
'In'	'Out'	Valid range of 'In'	Valid range of 'Out'
BOOL	BOOL	FALSE	FALSE
		TRUE	TRUE
BOOL	BYTE	FALSE	0
		TRUE	1
BOOL	WORD	FALSE	0
		TRUE	1
BOOL	DWORD	FALSE	0
		TRUE	1
BOOL	LWORD	FALSE	0
		TRUE	1
BOOL	USINT	FALSE	0
		TRUE	1

Data type		The data correspondence between ' In ' and ' Out '	
' In '	' Out '	Valid range of ' In '	Valid range of ' Out '
BOOL	UINT	FALSE	0
		TRUE	1
BOOL	UDINT	FALSE	0
		TRUE	1
BOOL	ULINT	FALSE	0
		TRUE	1
BOOL	SINT	FALSE	0
		TRUE	1
BOOL	INT	FALSE	0
		TRUE	1
BOOL	DINT	FALSE	0
		TRUE	1
BOOL	LINT	FALSE	0
		TRUE	1
BOOL	REAL	FALSE	0
		TRUE	1
BOOL	LREAL	FALSE	0
		TRUE	1
BOOL	TIME	FALSE	T#0ms
		TRUE	T#0ms
BOOL	DATE	FALSE	D#1970-01-01
		TRUE	D#1970-01-01
BOOL	TOD	FALSE	TOD#0: 0: 0.000
		TRUE	TOD#0: 0: 0.000
BOOL	DT	FALSE	DT#1970-01-01-00: 00: 00
		TRUE	DT#1970-01-01-00: 00: 00
BOOL	STRING	FALSE	'FALSE'
		TRUE	'TRUE'

- The example program and the timing diagram are shown below.

LD:

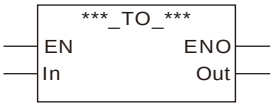


ST:

```
b: =BOOL_TO_INT( a );
```

2.10.2 ***_TO_*** (conversion of bit string to other data type)

This instruction converts bit strings to other data types. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
_TO_	Bit string conversion instruction	FUN		Out: = *** _TO_***(In); *** Represents different data type, i.e., Out: = BYTE _TO_INT(In);etc.

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data to convert	Input	Data to convert	Depends on variable type
Out	Conversion result	Output	Conversion result	Depends on variable type

	Boolean	Bit string				Integer								R		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		○	○	○	○															
Out	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note :** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- This instruction is used to convert data of the Bit string data type to other data types.
- The instruction name varies depending on the data type of 'In' and 'Out'. For example, if 'In' is a WORD type and 'Out' is a DINT type, the instruction name is WORD_ TO_ DINT.
- Bit string types convert to BOOL type.

  When converting a Bit string to BOOL, if the value of the Bit string type is 0, the Conversion result is FALSE; If the value is not 0, the conversion result is TRUE. The detailed rules are shown in the table below.

Data type		The data correspondence between 'In' and 'Out'	
'In'	'Out'	Valid range of 'In'	Valid range of 'Out'
BYTE	BOOL	0	FALSE
		16#1~16#FF	TRUE
WORD	BOOL	0	FALSE
		16#1~16#FFFF	TRUE
DWORD	BOOL	0	FALSE
		16#1~16#FFFF_FFFF	TRUE
LWORD	BOOL	0	FALSE
		16#1~16#FFFF_FFFF_FFFF_FFFF	TRUE

- Bit string types converts to Integer types.

  The relationship between Bit string types and Integer types conversion is shown in the table below.

Data type		The data correspondence between ' In ' and ' Out '	
' In '	' Out '	Valid range of ' In '	Valid range of ' Out '
BYTE	USINT	0~16#FF	0~255
	UINT	0~16#FF	0~255
	UDINT	0~16#FF	0~255
	ULINT	0~16#FF	0~255
	SINT	0~16#7F	0~127
	INT	0~16#FF	0~255
	DINT	0~16#FF	0~255
	LINT	0~16#FF	0~255
WORD	USINT	0~16#FF	0~255
	UINT	0~16#FFFF	0~65535
	UDINT	0~16#FFFF	0~65535
	ULINT	0~16#FFFF	0~65535
	SINT	0~16#7F	0~127
	INT	0~16#7FFF	0~32767
	DINT	0~16#FFFF	0~65535
	LINT	0~16#FFFF	0~65535
DWORD	USINT	0~16#FF	0~255
	UINT	0~16#FFFF	0~65535
	UDINT	0~16#FFFF_FFFF	0~4294967295
	ULINT	0~16#FFFF_FFFF	0~4294967295
	SINT	0~16#7F	0~127
	INT	0~16#7FFF	0~32767
	DINT	0~16# 7FFF_FFFF	0~2147483647
	LINT	0~16# FFFF_FFFF	0~4294967295
LWORD	USINT	0~16#FF	0~255
	UINT	0~16#FFFF	0~65535
	UDINT	0~16#FFFF_FFFF	0~4294967295
	ULINT	0~16#FFFF_FFFF_FFFF_FFFF	0~18446744073709551615
	SINT	0~16#7F	0~127
	INT	0~16#7FFF	0~32767
	DINT	0~16#7FFF_FFFF	0~2147483647
	LINT	0~16#7FFF_FFFF_FFFF_FFFF	0~9223372036854775807

- Bit string types convert to Real number types.

Data type		The data correspondence between ' In ' and ' Out '	
' In '	' Out '	Valid range of ' In '	Valid range of ' Out '
BYTE	REAL	0~16#FF	0~255
	LREAL	0~16#FF	0~255
WORD	REAL	0~16#FFFF	0~65535
	LREAL	0~16#FFFF	0~65535
DWORD	REAL	0~16#FFFF_FFFF	0~4294967295
	LREAL	0~16#FFFF_FFFF	0~4294967295
LWORD	REAL	0~16#FFFF_FFFF_FFFF_FFFF	0~1.844674e+19
	LREAL	0~16#FFFF_FFFF_FFFF_FFFF	0~1.84467440737096e+19

- Bit string types convert to Time or Date types.

The relationship between Bit string types and Time or Date types conversion is shown in the table below.

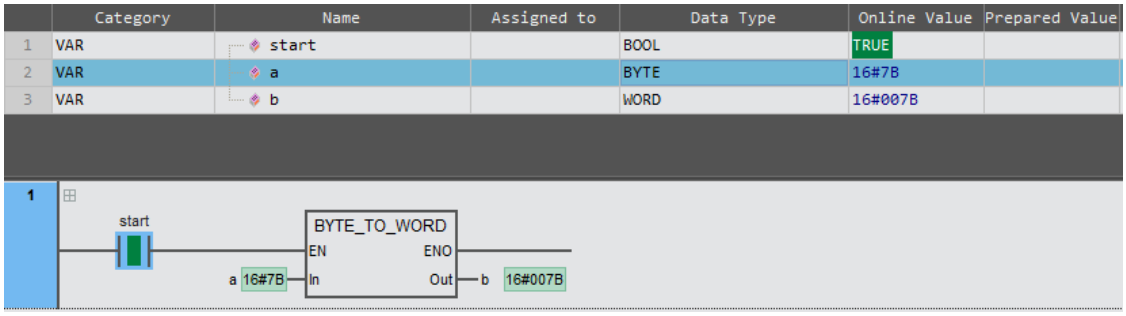
Data type		The data correspondence between ' In ' and ' Out '	
' In '	' Out '	Valid range of ' In '	Valid range of ' Out '
BYTE	TIME	0~16#FF	T#0ns~ T#255ns
	DATE	0~16#FF	D#1970-01-1~ D#1970-01-01
	TOD	0~16#FF	TOD#0: 0: 0.000~ TOD#0: 0: 0.255
	DT	0~16#FF	DT#1970-01-01-00: 00: 00~ DT#1970-01-01-00: 04: 15
WORD	TIME	0~16#FFFF	T#0ns~65us535ns
	DATE	0~16#FFFF	D#1970-01-01~ D#1970-01-01
	TOD	0~16#FFFF	TOD#0: 0: 0.000~ TOD#0: 1: 5.535
	DT	0~16#FFFF	DT#1970-01-01-00: 00: 00~ DT#1970-01-01-18: 12: 15
DWORD	TIME	0~16#FFFF_FFFF	T#0ms~T#4s294ms967us295ns
	DATE	0~16#FFFF_A500	D#1970-1-1~ D#2106-2-7
	TOD	0~16#526_5BFF	TOD#0: 0: 0.000~ TOD#23: 59: 59.999
	DT	0~16#FFFF_FFFF	DT#1970-01-01-00: 00: 00~ DT#2106-02-07-06: 28: 15
LWORD	TIME	0~16#FFFF_FFFF_ FFFF_ FFFF	T#213503d23h34m33s709ms551us615ns
	DATE	0~16#FFFF_A500	D#1970-01-01~ D#2106-02-07
	TOD	0~16#526_5BFF	TOD#0: 0: 0.000~ TOD#23: 59: 59.999
	DT	0~16#FFFF_FFFF	DT#1970-01-01-00: 00: 00~ DT#2106-02-07-06: 28: 15

- Bit string types convert to String type.

Data type		The data correspondence between ' In ' and ' Out '	
' In '	' Out '	Valid range of ' In '	Valid range of ' Out '
BYTE	STRING	0~16#FF	'0' ~ '255'
WORD	STRING	0~16#FFFF	'0' ~ '65535'
DWORD	STRING	0~16#FFFF_FFFF	'0' ~ '4294967295'
LWORD	STRING	0~16#FFFF_FFFF_FFFF_FFFF	'0' ~ '18446744073709551615'

- The example program is shown below.

LD:



ST:

```
b: =BYTE_TO_WORD( a );
```

2.10.3 ***_TO_*** (conversion of integer to other data types)

This instruction is used to convert integer type to other data types. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
_TO_	Integer conversion instruction	FUN	***_TO_*** EN ENO In Out	Out: = *** _TO_***(In); ***Represents different data type, i.e., Out: = =UINT _TO_BOOL(In);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data to convert	Input	Data to convert	Depends on variable type
Out	Conversion result	Output	Conversion result	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						○	○	○	○	○	○	○	○							
Out	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note :** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- This instruction is used to convert integer type 'In' to other data types 'Out'.
- The instruction name varies depending on the data type of 'In' and 'Out'. For example, if 'In' is a WORD type and 'Out' is a DINT type, the instruction name is WORD_ TO_ DINT.
- Integer types convert to Bool type

  When converting Integer types to BOOL type, if the value of the Integer type is 0, the Conversion result is FALSE; If the value is not 0, the conversion result is TRUE. The detailed rules are shown in the table below.

Data type		The data correspondence between 'In' and 'Out'	
'In'	'Out'	Valid range of 'In'	Valid range of 'Out'
USINT	BOOL	0	FALSE
		1~255	TRUE
UINT	BOOL	0	FALSE
		1~65535	TRUE
UDINT	BOOL	0	FALSE
		1~4294967295	TRUE
ULINT	BOOL	0	FALSE
		1~18446744073709551615	TRUE
SINT	BOOL	0	FALSE
		-128~-1, 1~127	TRUE
INT	BOOL	0	FALSE

Data type		The data correspondence between 'In' and 'Out'	
'In'	'Out'	Valid range of 'In'	Valid range of 'Out'
		-32768~-1, 1~32767	TRUE
DINT	BOOL	0	FALSE
		-2147483648~-1, 1~2147483647	TRUE
LINT	BOOL	0	FALSE
		-9223372036854775808~-1, 1~9223372036854775807	TRUE

- Integer types convert to Bit string types

The relationship between Integer types and Bit string types conversion is shown in the table below.

Data type		The data correspondence between 'In' and 'Out'	
'In'	'Out'	Valid range of 'In'	Valid range of 'Out'
USINT	BYTE	0~255	16#00~16#FF
	WORD	0~255	16#0000~16#00FF
	DWORD	0~255	16#0000_0000~16#0000_00FF
	LWORD	0~255	16#0000_0000_0000_0000~16#0000_0000_0000_00FF
UINT	BYTE	0~255	16#00~16#FF
	WORD	0~65535	16#0000~16#FFFF
	DWORD	0~65535	16#0000_0000~16#0000_FFFF
	LWORD	0~65535	16#0000_0000_0000_0000~16#0000_0000_0000_FFFF
UDINT	BYTE	0~255	16#00~16#FF
	WORD	0~65535	16#0000~16#FFFF
	DWORD	0~4294967295	16#0000_0000~16#FFFF_FFFF
	LWORD	0~4294967295	16#0000_0000_0000_0000~16#0000_0000_FFFF_FFFF
ULINT	BYTE	0~255	16#00~16#FF
	WORD	0~65535	16#0000~16#FFFF
	DWORD	0~4294967295	16#0000_0000~16#FFFF_FFFF
	LWORD	0~18446744073709551615	16#0000_0000_0000_0000~16#FFFF_FFFF_FFFF_FFFF
SINT	BYTE	0~127	16#00~16#7F
	WORD	0~127	16#0000~16#007F
	DWORD	0~127	16#0000_0000~16#0000_007F
	LWORD	0~127	16#0000_0000_0000_0000~16#0000_0000_0000_007F
INT	BYTE	0~127	16#00~16#7F
	WORD	0~32767	16#0000~16#7FFF
	DWORD	0~32767	16#0000_0000~16#0000_7FFF
	LWORD	0~32767	16#0000_0000_0000_0000~16#0000_0000_0000_7FFF
DINT	BYTE	0~127	16#00~16#7F
	WORD	0~32767	16#0000~16#7FFF
	DWORD	0~2147483647	16#0000_0000~16#7FFF _FFFF
	LWORD	0~2147483647	16#0000_0000_0000_0000~16#0000_0000_ 7FFF_FFFF
LINT	BYTE	0~127	16#00~16#7F
	WORD	0~32767	16#0000~16#7FFF
	DWORD	0~2147483647	16#0000_0000~16#7FFF _FFFF
	LWORD	0~9223372036854775807	16#0000_0000_0000_0000 ~16#7FFF_FFFF_FFFF_FFFF

- Integer types convert to Real number types

The relationship between Integer types and Real number types conversion is shown in the table below.

Data type		The data correspondence between ' In ' and ' Out '	
' In '	' Out '	Valid range of ' In '	Valid range of ' Out '
USINT	REAL	0~255	0~255
	LREAL	0~255	0~255
UINT	REAL	0~65535	0~65535
	LREAL	0~65535	0~65535
UDINT	REAL	0~4294967295	0~4.294967e+9
	LREAL	0~4294967295	0~4294967295
ULINT	REAL	0~18446744073709551615	0~1.844674e+19
	LREAL	0~18446744073709551615	0~1.84467440737096e+19
SINT	REAL	-128~127	-128~127
	LREAL	-128~127	-128~127
INT	REAL	-32768~32767	-32768~32767
	LREAL	-32768~32767	-32768~32767
DINT	REAL	-2147483648~2147483647	-2.147484e+9~2.147484e+9
	LREAL	-2147483648~2147483647	-2147483648~21474836488
LINT	REAL	-9223372036854775808~ 9223372036854775807	-9.223372e+18~9.223372e+18
	LREAL	-9223372036854775808~ 9223372036854775807	-9.22337203685478e+18~9.22337203685478e+18

- Integer types convert to Time or Date types

The relationship between Integer types and Time or Date types conversion is shown in the table below.

The data type of ' In ' and ' Out '		The data correspondence between ' In ' and ' Out '	
' In '	' Out '	Valid range of ' In '	Valid range of ' Out '
USINT	TIME	0~255	T#0ns~ T#255ns
	DATE	0~255	D#1970-01-01~ D#1970-01-01
	TOD	0~255	TOD#0: 0: 0.000~ TOD#0: 0: 0.255
	DT	0~255	DT#1970-01-01-00: 00: 00~ DT#1970-01-01-00: 04: 15
UINT	TIME	0~65535	T#0ns~T#65us535ns
	DATE	0~65535	D#1970-01-01~ D#1970-01-01
	TOD	0~65535	TOD#0: 0: 0.000~ TOD#0: 1: 5.535
	DT	0~65535	DT#1970-01-01-00: 00: 00~ DT#1970-01-01-18: 12: 15
UDINT	TIME	0~4294967295	T#0ms~T#4s294ms967us295ns
	DATE	0~4294944000	D#1970-01-01~ D#2106-02-07
	TOD	0~86399999	TOD#0: 0: 0.000~ TOD#23: 59: 59.999
	DT	0~4294967295	DT#1970-01-01-00: 00: 00~ DT#2106-02-07-06: 28: 15
ULINT	TIME	0~18446744073709551615	T#0ms~T#213503d23h34m33s709.551ms
	DATE	0~4294944000	D#1970-01-01~ D#2106-02-07
	TOD	0~86399999	TOD#0: 0: 0.000~ TOD#23: 59: 59.999
	DT	0~4294967295	DT#1970-01-01-00: 00: 00~ DT#2106-02-07-06: 28: 15
SINT	TIME	0~127	T#0ns~ T#127ns
	DATE	0~127	D#1970-01-01~ D#1970-01-01
	TOD	0~127	TOD#0: 0: 0.000~ TOD#0: 0: 0.127
	DT	0~127	DT#1970-01-01-00: 00: 00~ DT#1970-01-01-00: 02: 07
INT	TIME	0~32767	T#0ns~ T#32us767ns
	DATE	0~32767	D#1970-01-01~ D#1970-01-01

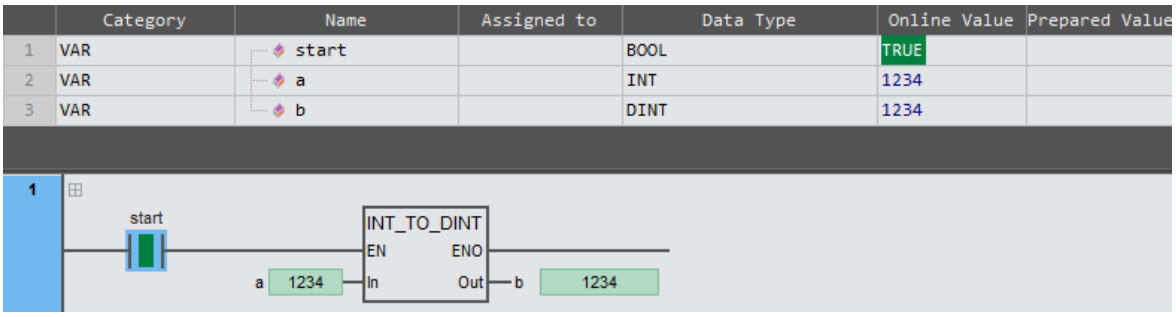
The data type of ' In ' and ' Out '		The data correspondence between ' In ' and ' Out '	
' In '	' Out '	Valid range of ' In '	Valid range of ' Out '
DINT	TOD	0~32767	TOD#0: 0: 0.000~ TOD#0: 0: 32.767
	DT	0~32767	DT#1970-01-01-00: 00: 00~ DT#1970-01-01-09: 06: 07
	TIME	0~2147483647	T#0ns~ T#2s147ms483ms647ns
	DATE	0~2147483647	D#1970-01-01~ D#2038-01-19
	TOD	0~86399999	TOD#0: 0: 0.000~ TOD#23: 59: 59.999
	DT	0~2147483647	DT#1970-01-01-00: 00: 00~ D#2038-01-19-03: 14: 07
LINT	TIME	0~9223372036854775807	T#0ms~106751d23h47m16s854.776ms
	DATE	0~4294944000	D#1970-01-01~ D#2106-02-07
	TOD	0~86399999	TOD#0: 0: 0.000~ TOD#23: 59: 59.999
	DT	0~4294967295	DT#1970-01-01-00: 00: 00~ D#2106-02-07-06: 28: 15

- Integer types convert to String type

Data type		The data correspondence between ' In ' and ' Out '	
In	Out	Valid range of ' In '	Valid range of ' Out '
USINT	STRING	0~255	' 0 ' ~ ' 255 '
UINT	STRING	0~65535	' 0 ' ~ ' 65535 '
UDINT	STRING	0~4294967295	' 0 ' ~ ' 4294967295 '
ULINT	STRING	0~18446744073709551615	' 0 ' ~ ' 18446744073709551615 '
SINT	STRING	-128~127	' -128 ' ~ ' 127 '
INT	STRING	-32768~32767	' -32768 ' ~ ' 32767 '
DINT	STRING	-2147483648~2147483647	' -2147483648 ' ~ ' 2147483647 '
LINT	STRING	-9223372036854775808~ 9223372036854775807	' -9223372036854775808 ' ~ ' 9223372036854775807 '

- The example program is shown below.

LD:

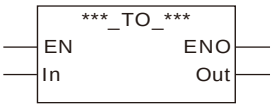


ST:

```
b: =INT _TO_DINT( a );
```

2.10.4 REAL/LREAL_TO_*** (conversion of real numbers to other data types)

This instruction is used to convert Real number types to other data types. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
_TO_	Real number conversion instruction	FUN		Out: = *** _TO_***(In); ***Represents different data type, i.e., Out: = REAL_TO_DINT(In);etc.

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data to convert	Input	Data to convert	Depends on variable type
Out	Conversion result	Output	Conversion result	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														○	○					
Out	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note :** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- This instruction is used to convert Real number types to other data types.
- The instruction name varies depending on the data type of 'In' and 'Out'. For example, if 'In' is a REAL type and 'Out' is a DINT type, the instruction name is REAL_ TO_ DINT.
- Real number types convert to Bool type

When converting Integer types to BOOL type, if the value of the Integer type is 0, the conversion result is FALSE; If the value is not 0, the conversion result is TRUE. The detailed rules are shown in the table below.

Data type		The data correspondence between 'In' and 'Out'	
'In'	'Out'	'In'	'Out'
REAL	BOOL	-3.1	TRUE
		0	FALSE
		3.1	TRUE
LREAL	BOOL	-6	TRUE
		0	FALSE
		6	TRUE

- Real number types convert to Integer or Bit string types

When converting Real number types to integers or Bit string types, the decimal part of the 'In' value is rounded, and the rounded integer of the 'In' value is the value of 'Out'. The relevant examples are shown in the table below.

Data type		The data correspondence between 'In' and 'Out'	
'In'	'Out'	'In'	'Out'
REAL	SINT	1.49	1
REAL	SINT	1.50	2
REAL	SINT	1.55	2
REAL	SINT	-1.49	-1
REAL	SINT	-1.50	-2
REAL	SINT	-1.55	-2
REAL	SINT	127	127
REAL	BYTE	127	16#7F

- Real number types convert to Bit string types

When converting Real number types to Bit string types, round the Input Real number type first, and the Output value is the rounded integer part. During the conversion operation of Real number type to Bit string type, the corresponding relationship within the valid range of Input and Output data is shown in the table below.

Data type		The data correspondence between 'In' and 'Out'	
'In'	'Out'	Valid range of 'In'	Valid range of 'Out'
REAL	BYTE	0~255	16#00~16#FF
	WORD	0~65535	16#0000~16#FFFF
	DWORD	0~4.294967e+9	16#0000_0000~16#FFFF_FFFF
	LWORD	0~1.844674e+19	16#0000_0000_0000_0000~16#FFFF_FFFF_FFFF_FFFF
LREAL	BYTE	0~255	16#00~16#FF
	WORD	0~65535	16#0000~16#FFFF
	DWORD	0~4294967295	16#0000_0000~16#FFFF_FFFF
	LWORD	0~1.84467440737095e+19	16#0000_0000_0000_0000~16#FFFF_FFFF_FFFF_FFFF

- Real number types convert to Integer types

When converting Real number types to Integer types, round the Input Real number type first, and the Output value is the rounded integer part. During the conversion operation of Real number type to Integer types, the corresponding relationship within the valid range of Input and Output data is shown in the table below.

Data type		The data correspondence between 'In' and 'Out'	
'In'	'Out'	Valid range of 'In'	Valid range of 'Out'
REAL	USINT	0~255	0~127
	UINT	0~65535	0~32767
	UDINT	0~4.294967e+9	0~4294967295
	ULINT	0~1.844674e+19	0~18446744073709551615
	SINT	- 128~ 127	- 128~ 127
	INT	-32768~32767	-32768~32767
	DINT	-2.147484e+9~2.147484e+9	-2147483648~2147483647
	LINT	-9.223372e+18~9.223372e+18	-9223372036854775808~ 9223372036854775807
LREAL	USINT	0~255	0~255
	UINT	0~65535	0~65535
	UDINT	0~4294967295	0~4294967295
	ULINT	0~1.84467440737095e+19	0~18446744073709551615
	SINT	-128~127	-128~127

	INT	-32768~32767	-32768~32767
	DINT	-2147483648~21474836488	-2147483648~2147483647
	LINT	-9.22337203685477e+18~9.22337203685477e+18	-9223372036854775808~9223372036854775807

- Real number types convert to Time or Date types。

  When converting the Real number type to Time or Date types, first convert the Real number type to ULINT data, and then convert the ULINT data to the corresponding time in units of the Output type. ULINT conversion to Time or Date type can refer to Note in the ' Convert integer type to other data types ' instruction.

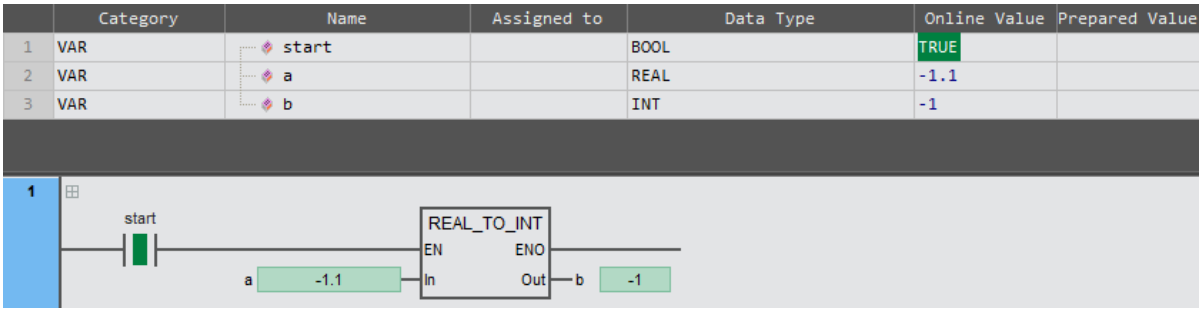
- Real number types convert to String type

  When converting the Real number type to the String type, the characters and symbols in the Input data are converted to corresponding characters. The example of converting the Real number type to the String type is shown in the table below.

Data type		The data correspondence between ' In ' and ' Out '	
In	Out	Value of ' In '	Value of ' Out '
REAL	STRING	123.123	'123.123'
REAL	STRING	-123.123	'-123.123'
REAL	STRING	1.23e+07	'1.23e+07'

■ The example program is shown below.

LD:



ST:

```
b: =REAL _TO_INT( a );
```

2.10.5 ***_TO_*** (conversion of time and date to other data types)

This instruction is used to convert Time or Date types to other data types. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
_TO_	Time, Date conversion instruction	FUN	***_TO_*** EN ENO In Out	Out: = ***_TO_***(In); ***Represents different data type, i.e., Out: =TIME_TO_INT(In);etc.

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data to convert	Input	Data to convert	Depends on variable type
Out	Conversion result	Output	Conversion result	Depends on variable type

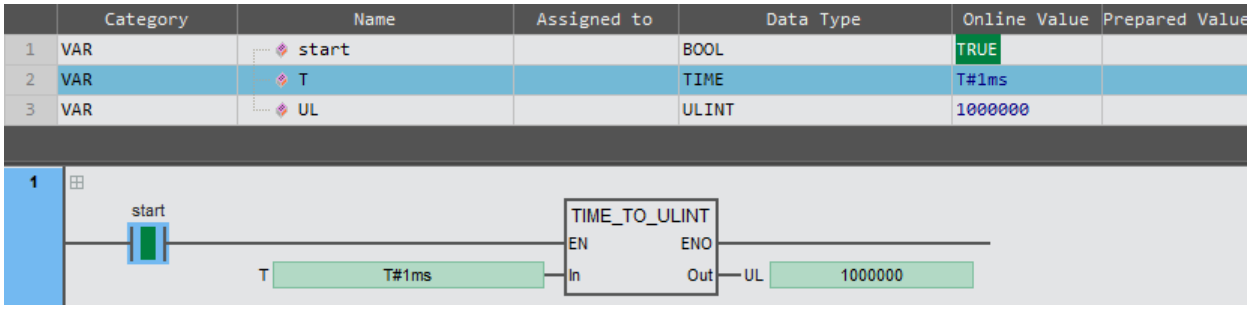
	Boolean	Bit string				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																○	○	○	○	
Out	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note :** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- This instruction is used to convert Time or Date types to other data types.
- The instruction name varies depending on the data type of 'In' and 'Out'. For example, if 'In' is a Time type and 'Out' is a DINT type, the instruction name is Time_ TO_ DINT.
- The example program and the timing diagram are shown below.

LD:

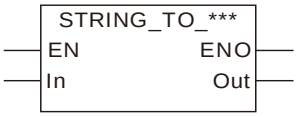


ST:

UL: =TIME _TO_ULINT(T);

2.10.6 STRING_TO_*** (conversion of STRING to other data types)

This instruction is used to convert String type to other data types. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
STRING_TO_***	STRING conversion instruction	FUN		Out: = STRING _TO_***(In); ***Represents different data type, i.e., Out: =STRING _TO_DINT(In), etc.

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Data to convert	Input	Data to convert	Depends on variable type
Out	Conversion result	Output	Conversion result	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				○
Out	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

***Note :** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

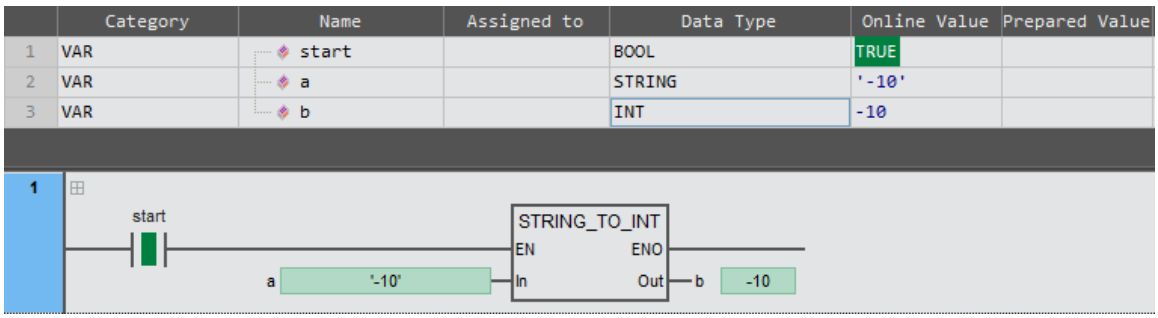
- This instruction is used to convert the String type to other data types.
- The instruction name varies depending on the data type of 'In' and 'Out'. For example, if 'In' is a String type and 'Out' is a INT type, the instruction name is String_ TO_ INT.
- The relationship between String type and other data types conversion is shown in the table below.

Data type		The data correspondence between 'In' and 'Out'	
'In'	'Out'	Valid range of 'In'	Valid range of 'Out'
STRING	BOOL	Other string	FALSE
		'TRUE' or' true'	TRUE
STRING	BYTE	'0' ~ '255'	0~16#FF
STRING	WORD	'0' ~ '65535'	0~16#FFFF
STRING	DWORD	'0' ~ '4294967295'	0~16#FFFF_FFFF
STRING	LWORD	'0' ~ '18446744073709551615'	0~16#FFFF_FFFF_FFFF_FFFF
STRING	USINT	'0' ~ '255'	0~255
STRING	UINT	'0' ~ '65535'	0~65535
STRING	UDINT	'0' ~ '4294967295'	0~4294967295
STRING	ULINT	'0' ~ '18446744073709551615'	0~18446744073709551615
STRING	SINT	'-128' ~ '127'	-128~127
STRING	INT	'--32768' ~ '32767'	-32768~32767
STRING	DINT	'-2147483648' ~ '2147483647'	-2147483648~2147483647

Data type		The data correspondence between ' In ' and ' Out '	
' In '	' Out '	Valid range of ' In '	Valid range of ' Out '
STRING	LINT	'-9223372036854775808' ~ '9223372036854775807'	-9223372036854775808~ 9223372036854775807
STRING	REAL	'-3.402823e+38' ~ '-1.175495e-38', '0', '1.175495e-38 ~ '3.402823e+38'	-3.402823e+38 ~ -1.175495e-38, 0, 1.175495e-38 ~ 3.402823e+38
STRING	LREAL	'-1.79769313486231e+308' ~ '-2.22507385850721e-308', '0', '2.22507385850721e−308' ~ '1.79769313486231e+308'	-1.79769313486231e+308 ~ -2.22507385850721e-308, 0, 2.22507385850721e−308 ~ 1.79769313486231e+308,
STRING	TIME	'T#0ms' ~ 'T#213503d23h34m33s709.551ms'	T#0ms~ T#213503d23h34m33s709.551ms
STRING	DATE	'D#1970-01-01' ~ 'D#2106-02-07'	D#1970-01-01~ D#2106-02-07
STRING	TOD	'TOD#0: 0: 0.000' ~ 'TOD#23: 59: 59.999'	TOD#0: 0: 0.000~ TOD#23: 59: 59.999
STRING	DT	'DT#1970-01-01-00: 00: 00' ~ 'DT#2106-02-07-06: 28: 15'	DT#1970-01-01-00: 00: 00~ DT#2106-02-07-06: 28: 15

- The example program and the timing diagram are shown below.

LD:



ST:

```
b: =STRING _TO_INT( a );
```

2.11 Text string Instructions

2.11.1 CONCAT(combination of strings)

Concatenates two strings into a single string. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
CONCAT	Combination of strings	FUN	CONCAT EN ENO In1 Out In2	Out : = CONCAT(In1 , In2);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	String	Input	String 1 to be concatenated	Depends on variable type
In2	String		String 2 to be concatenated	Depends on variable type
Out	String	Output	Concatenated string	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWOR	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1、In2																				○
Out																				○

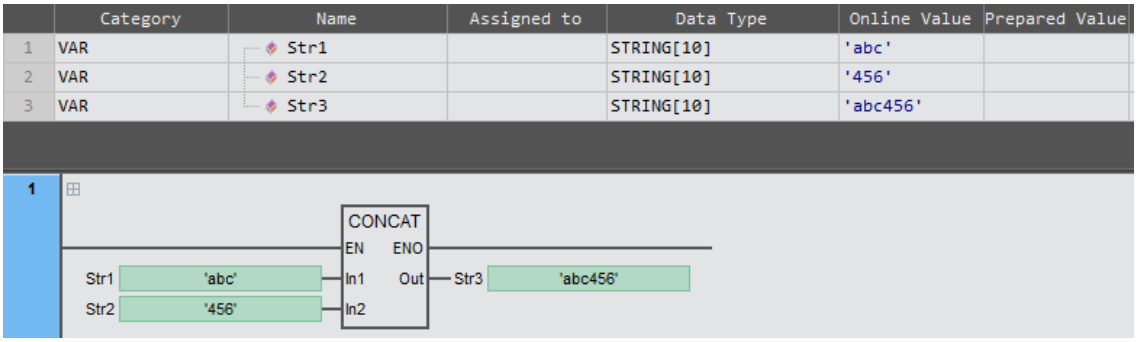
***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- This instruction concatenates 'In1' and 'In2' into a new single string in order.
- The example program is shown below.

LD:

When In1 is 'abc', In2 is '456', the output value will be 'abc456'.

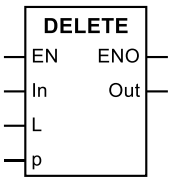


ST:

Str3: =CONCAT(Str1 ,Str2);

2.11.2 DELETE (deletion of strings)

This instruction deletes a number of characters from a string to produce a new string, beginning with the character in the specific position. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DELETE	Deletion of strings	FUN		Out : = DELETE(In , L , P);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	String for deletion	Input	String to be modified	Depends on variable type
L	Length of characters		Length of the partial string to be deleted, number of characters	0 ~ Maximum length of In
P	Deletion position		Position in input string after which the deletion starts. Counted from left, starting with 1	1 ~ Maximum length of In
Out	Resulting string	Output	String remaining after deletion	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				○
L							○													
P							○													
Out																				○

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

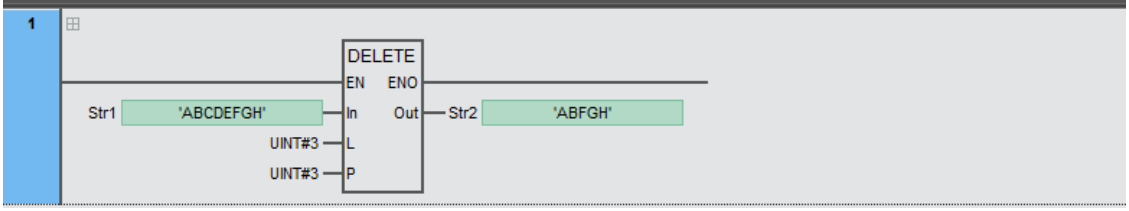
- This instruction deletes 'L' characters from 'In' to produce a new string 'Out', beginning with the character in the 'P' position.

- The example program is shown below.

LD:

When In is 'ABCDEFGH' , L is '3', P is '3'， the output value will be ' AEFGH'.

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	Str1		STRING[10]	'ABCDEFGH'	
2	VAR	Str2		STRING[10]	'ABFGH'	

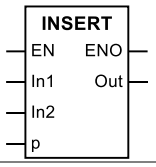


ST:

Str2: =DELETE(Str1 , UINT#3 , UINT#3);

2.11.3 INSERT(insert of strings)

This instruction inserts a string into another string at a specific position to produce a new string. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
INSERT	Insert of strings	FUN		Out : = INSERT(In1 , In2 , P);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Source string	Input	String into which In2 is inserted	Depends on variable type
In2	Insert string		String to be inserted into In1	Depends on variable type
P	Insert position		Insert position	0 ~ Maximum length of In1
Out	Resulting string	Output	Resulting string	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																				○
In2																				○
P							○													
Out																				○

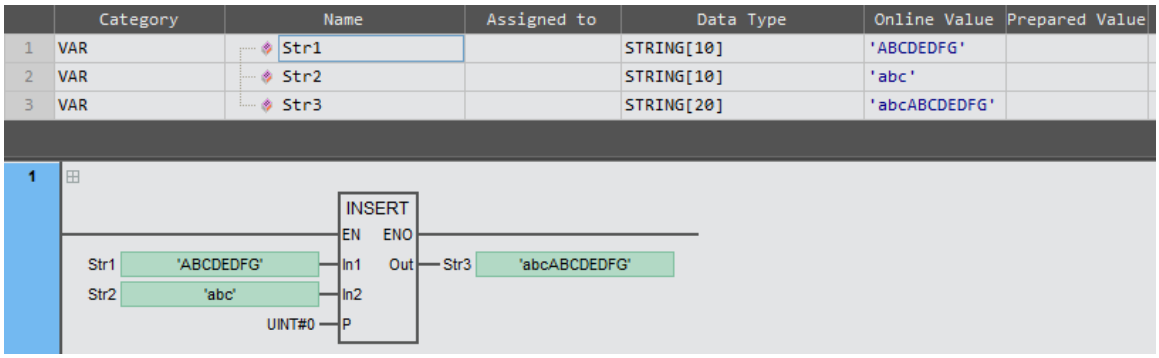
***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- This instruction inserts 'In2' into 'In1' after position 'P' to produce a new string 'Out'.
- The example program1 is shown below.

LD:

When In1 is 'ABCDEFGH', In2 is 'abc', P is 0，the value of Out will be 'abcABCDEFGH'.



ST:

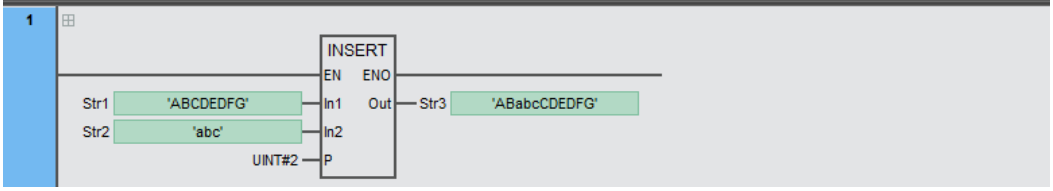
Str3: =INSERT(Str1 , Str2 , UINT#0);

- The example program2 is shown below.

LD:

When In1 is 'ABCDEFGH', In2 is 'abc', 'P' is 2，the output value will be 'ABabcCDEFGH'.

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	Str1		STRING[10]	'ABCDEDFG'	
2	VAR	Str2		STRING[10]	'abc'	
3	VAR	Str3		STRING[20]	'ABabcCDEDFG'	



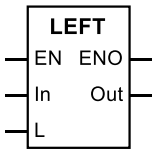
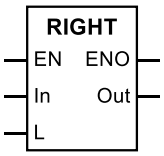
ST:

Str3: =INSERT(Str1 , Str2 , UINT#2);

2.11.4 LEFT/RIGHT (return of the left/right, source strings)

This instruction returns a specific number of characters in a string to produce a new string, starting from left/right.

Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
LEFT	Return of the left	FUN		Out : = LEFT(In , L);
RIGHT	Return of the right	FUN		Out : = RIGHT(In , L);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Source string	Input	String to be analyzed	Depends on variable type
L	Number of characters		Number of characters	0 ~ Maximum length of In
Out	Resulting string	Output	Resulting string	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				○
L							○													
Out																				○

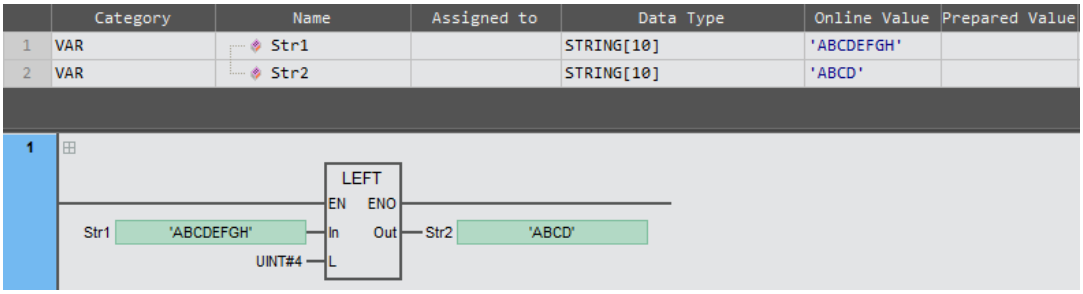
***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- The LEFT/RIGHT instruction returns the first 'L' characters from the left/right in the string 'In' to produce a new string 'Out'.
- The example program of LEFT is shown below.

LD:

LEFT returns the characters from the left in string In, when In is 'ABCDEFGH', L is 4， the output value will be 'ABCD'.



ST:

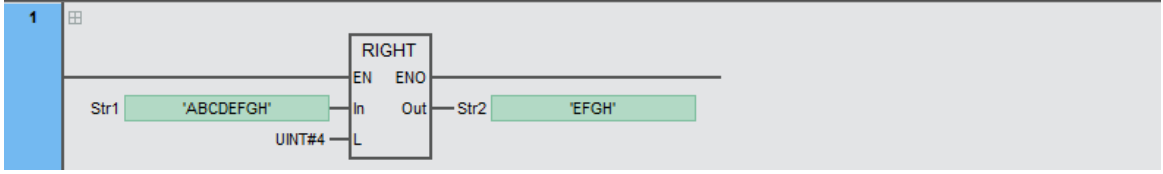
Str2: = LEFT(Str1 , UINT#4);

- The example program of RIGHT is shown below.

LD:

RIGHT returns the characters from the right in string In , when In is 'ABCDEFGH', L is 4， the output value will be 'EFGH'.

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	Str1		STRING[10]	'ABCDEFGH'	
2	VAR	Str2		STRING[10]	'EFGH'	

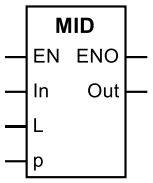


ST:

Str2: = RIGHT(Str1 , UINT#4);

2.11.5 MID (return of a partial string)

This instruction returns a specific number of characters in a string to produce a new string, starting from a specific position. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
MID	Return of a partial string	FUN		Out : = MID(In , L , P);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Source string	Input	String to be retrieved	Depends on variable type
L	Number of characters		Number of characters to delete, counting from left	0 ~ Maximum length of In
P	Starting position		Start position of the characters to be retrieved	1 ~ Maximum length of In
Out	Resulting string	Output	Resulting string	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				○
L							○													
P							○													
Out																				○

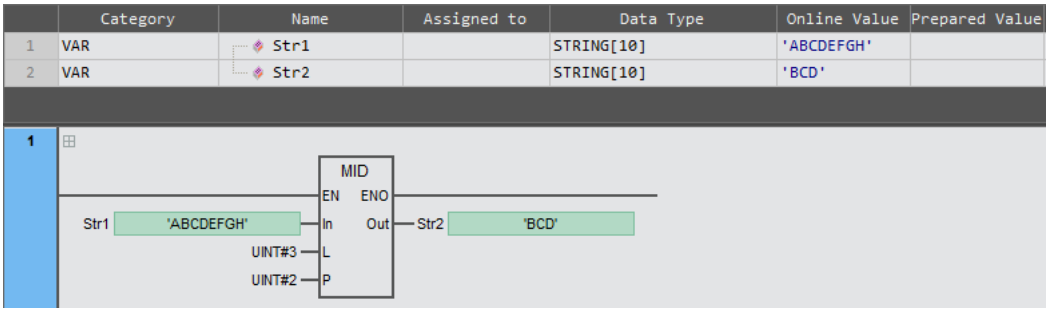
***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- The Mid instruction retrieves 'L' characters from the 'In'string beginning with the character at position 'P', and the result is assigned to 'Out'.
- The example program is shown below.

LD:

When In is 'ABCDEFGH',, L is 3, P is 2, the output value will be 'BCD '.

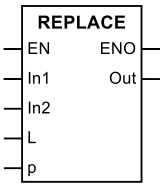


ST:

```
Str2: = MID(Str1 , UINT#3 , UINT#2 );
```

2.11.6 REPLACE (replacement of a partial string)

This instruction replaces part of a string at a specified position and length with another string. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
REPLACE	Replacement of a partial string	FUN		Out : = REPLACE(In1 , In2 , L , P);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Source string	Input	Source string	Depends on variable type
In2	Insert string		String to be inserted into In1	Depends on variable type
L	Number of characters		Number of characters to delete, counting from left	0 ~ Maximum length of In
P	Starting position		Start position of the characters to be replaced	1 ~ Maximum length of In
Out	Resulting string	Output	Resulting string	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																				○
In2																				○
L							○													
P							○													
Out																				○

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

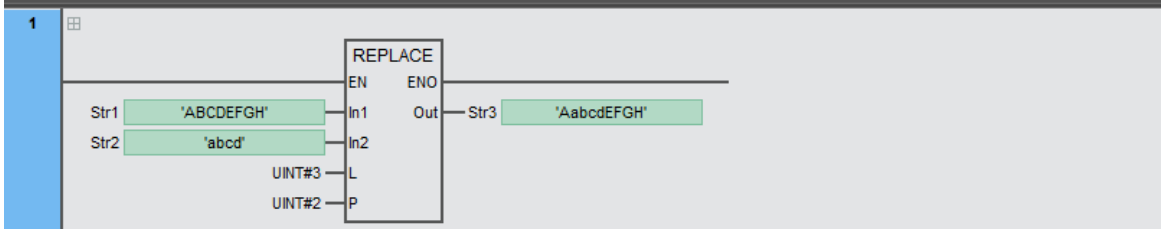
- The REPLACE instruction replaces part of the string for replacement In1 with string to insert In2. First, the number of characters specified by L from the position specified by P is deleted from In1. then In2 is inserted for the deleted character. The result is assigned to Out.

- The example program is shown below.

LD:

When In1 is 'ABCDEFGH', In2 is 'abcd', L is 3, P is 2, the output value will be ' AabcdEFGH'.

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	Str1		STRING[10]	'ABCDEFGH'	
2	VAR	Str2		STRING[10]	'abcd'	
3	VAR	Str3		STRING[10]	'AabcdEFGH'	



ST:

Str3: =REPLACE(Str1 , Str2 , UINT#3 , UINT#2);

2.11.7 LEN (calculation of string length)

This instruction calculates the number of characters in a string. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
LEN	Calculation of string length	FUN	LEN EN EN0 In OUT	Out : = LEN(In);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In	Source string	Input	String to be calculated	Depends on variable type
Out	Length of characters	Output	Length of string In1	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				○
Out							○													

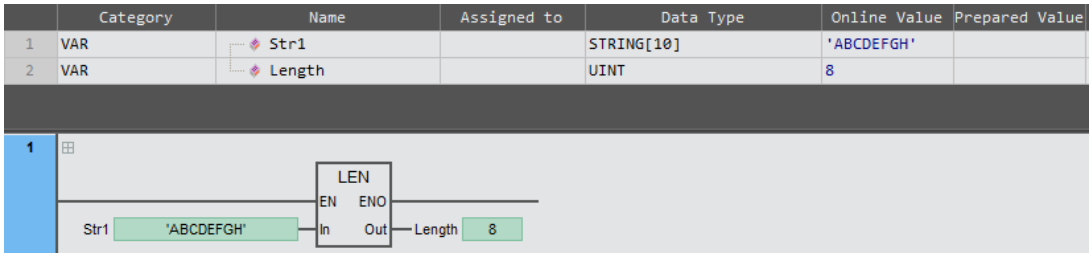
***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- This instruction calculates the number of characters in 'In' and assigns results to 'Out'.
- The example program is shown below.

LD:

When In is 'ABCDEFGH', the output value will be 8.

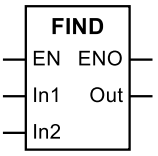


ST:

```
Length :=EN( Str1 );
```

2.11.8 FIND (string position finding)

This instruction searches for the position of a partial string within a string. Library: Standard.

Instruction	Name	FB/FUN	Graphic expression	ST expression
FIND	String position finding	FUN		Out : =FIND(In1 , In2);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
In1	Source string	Input	String to search	Depends on variable type
In2	Search key		String to search for	Depends on variable type
Out	Starting position	Output	Character position of the first occurrence of In2 in In1	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																				○
In2																				○
Out							○													

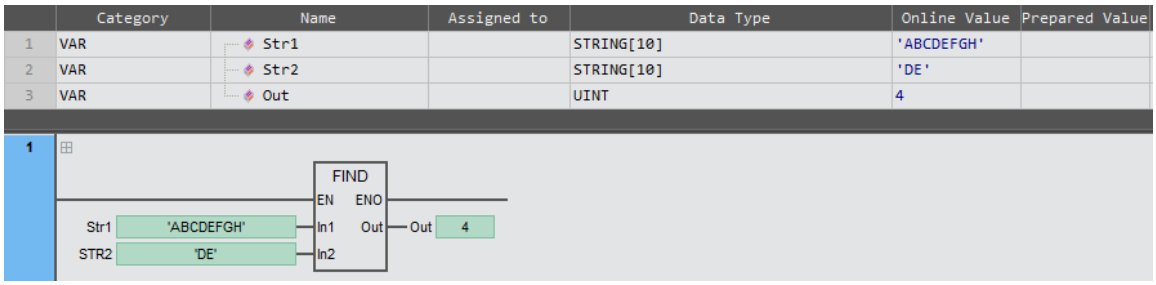
***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- The FIND instruction searches for the key In2 in the string to In1. The position of In2 from the start of In1 is assigned to the search result 'Out'.
- If 'In2' cannot be found in 'In1', then Out: =0. If there are multiple 'In2' in 'In1', then Out will be the position that 'In2' appears for the first time.
- The example program is shown below.

LD:

When In1 is 'ABCDEFGH', and In2 is 'DE', the output value will be 4.



ST:

Out: = FIND(Str1 , Str2);

2.12 IO refreshing and PID

2.12.1 UpdataInput (input channel immediate refreshing)

This instruction is used to immediately refresh the state of the input channels. Library: Standard_Part2

Instruction	Name	FB/FUN	Graphic expression	ST expression
UpdateInput	input channel immediate refreshing	FUN	UpdateInput EN ENO Input Num	UpdateInput(Input , Num);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Input	Input channel	Input	Input channel of PLC	0~15 (0: %IX0.0、1: %IX0.1、7: %IX0.7、8: %IX1.0、14: %IX1.6, 15: %IX1.7)
Num	Number		The number of input channels need to refresh, counting from Input	1~16

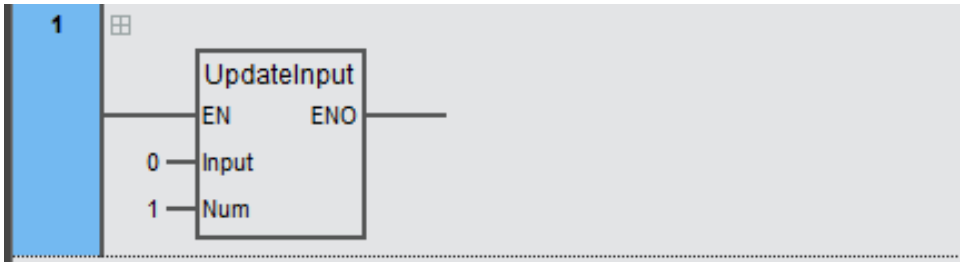
	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input							○													
Num						○														

*Note: The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- The UpdataInput instruction is used to immediately refresh the state of the PLC input channels (module input channels are not supported). When the instruction is executed, the state of the external input channels can be immediately detected; If this instruction is not executed, the state of the external input channels will be refreshed at the beginning of the next task period.
- The example program is shown below.

LD:

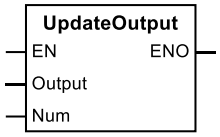


ST:

UpdateInput(0 , 1);

2.12.2 UpdataOutput (output channel immediate refreshing)

This instruction is used to immediately refresh the state of the output channels. Library: Standard_Part2

Instruction	Name	FB/FUN	Graphic expression	ST expression
UpdateOutput	Output channel immediate refreshing	FUN		UpdateOutput (Output , Num);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Output	Output channel	Input	Output channel of PLC	0~15 (0: %QX0.0、1: %QX0.1、7: %QX0.7、8: %QX1.0、14: %IX1.6, 15: %IX1.7)
Num	Number		The number of output channels need to refresh, counting from Output	1~16

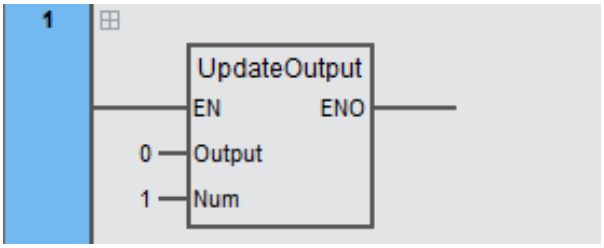
	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Output							○													
Num						○														

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- The UpdateOutput instruction is used to immediately refresh the state of the PLC output channels (module output channels are not supported). When the instruction is executed, the state of the external output channels can be immediately detected; If this instruction is not executed, the state of the external output channels will be refreshed at the beginning of the next task period.
- The example program is shown below.

LD:

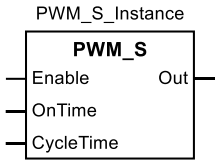


ST:

```
UpdateOutput (0 , 1 );
```

2.12.3 PWM_S (variable duty cycle pulse output)

This instruction adjusts the variable duty cycle pulse output. Library: Standard_Part2.

Instruction	Name	FB/FUN	Graphic expression	ST expression
PWM_S	Variable duty cycle pulse output	FB		PWM_S_Instance (Enable :=Parameter, OnTime :=Parameter, CycleTime:=Parameter , Out=>Parameter);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Enable	Enable	Input	Enables the execution of the function block	TRUE or FALSE
OnTime	Pulse width		Output pulse width (unit: ms)	0~32767
CycleTime	Pulse period		Output pulse period (unit: ms)	1~32767
Out	Pulse output	Output	Output pulse variable	TRUE or FALSE

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	○																			
OnTime											○									
CycleTime											○									
Out	○																			

*Note: The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

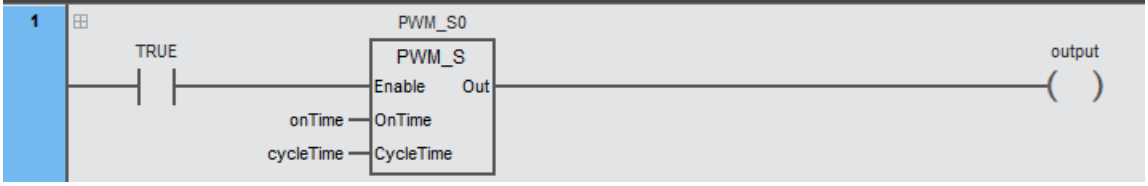
■ Function description

- This instruction adjusts the variable duty cycle pulse output. 'OnTime' is output pulse width (unit: ms), the time for high level. 'CycleTime' is the output pulse period (unit: ms), the total time for high and low levels.
- 'OnTime' and 'CycleTime' can be changed when the instruction is executed.
- If 'OnTime' is greater than or equal to 'CycleTime', the value of 'Out' has always been TRUE.
- When this instruction is used to adjust the variable duty cycle pulse output, the values of 'OnTime' and 'CycleTime' are inaccurate. Do not use this instruction if the values of 'OnTime' and 'CycleTime' are required to be accurate.

- The example program and the timing diagram are shown below.

LD:

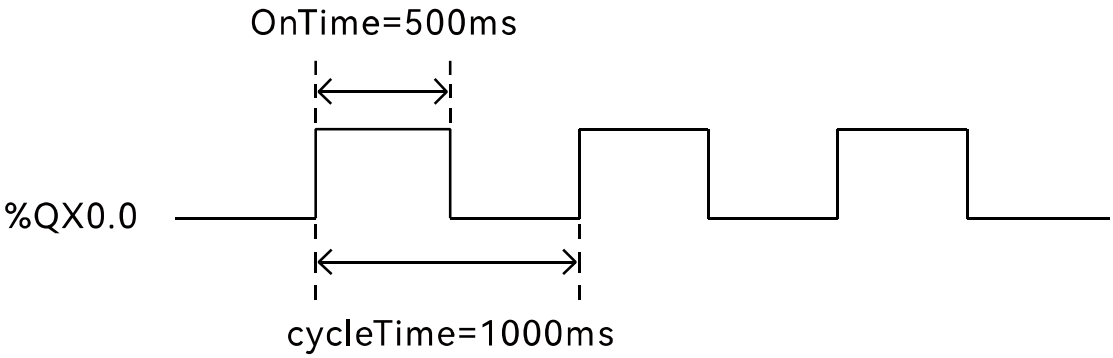
	Category	Name	Assigned to	Data Type	Initial Value
1	VAR	PWM_S0		PWM_S	
2	VAR	onTime		INT	
3	VAR	cycleTime		INT	
4	VAR	output	%QX0.0	BOOL	



ST:

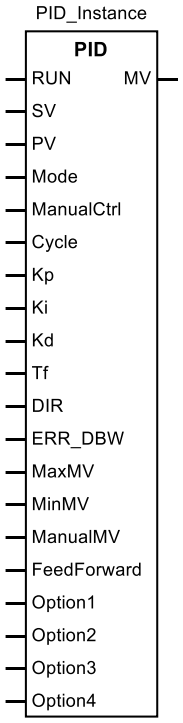
```
PWM_S0(Enable: = TRUE , OnTime: = onTime , CycleTime: = cycleTime , Out=> output );
```

Timing diagram:



2.12.4 PID(self-tuning PID)

This instruction performs PID control. Library: Standard_Part2.

Instruction	Name	FB/FUN	Graphic expression	ST expression
PID	Self-tuning PID	FB		PID_Instance(RUN:=Parameter , SV:=Parameter , PV:=Parameter , Mode:=Parameter , ManualCtrl:=Parameter , Cycle:=Parameter , Kp:=Parameter , Ki :=Parameter, Kd:=Parameter , Tf:=Parameter , DIR:=Parameter , ERR_DBW :=Parameter, MaxMV:=Parameter , MinMV:=Parameter , ManualMV :=Parameter, FeedForward:=Parameter , Option1:=Parameter , Option2:=Parameter , Option3 :=Parameter, Option4:=Parameter , MV =>Parameter);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Run	Execution condition	Input	TRUE: Execute FALSE: Stop	TRUE or FALSE
SV	Set value		Set value	Depends on variable type
PV	Process value		Process value	Depends on variable type
Mode	PID Mode		0: PID normal mode; 1: Self-tuning PID mode，this mode will calculate appropriate value of 'Kp'、'Ki'、'Kd'、'Tf' and other parameters ，then write this value to input variables。Mode will change to Mode0(PID normal mode) when self-tuning PID mode is end.	0、1
ManualCtrl	Manual/auto control		TRUE: Manual operation FALSE: Automatic operation	TRUE or FALSE
Cycle	Period		Period	1ms-40000ms
Kp	Proportion coefficient		Proportion coefficient	Depends on variable type
Ki	Integral coefficient		Integral coefficient	Depends on variable type
Kd	Differential coefficient		Differential coefficient	Depends on variable type
Tf	Derivative-action time constant		Derivative-action time constant	Depends on variable type
DIR	Direction		TRUE: Positive direction（E=SV-PV） FALSE: Negative direction（E=PV-SV）	TRUE or FALSE

Name	Meaning	I/O	Description	Parameter scope
ERR_DBW	Ignore Extents		Ignore Extents	Depends on variable type
MaxMV	Maximum		Maximum value of 'MV'	Depends on variable type
MinMV	Minimum		Minimum value of 'MV'	Depends on variable type
ManualMV	Manual		Manual value of 'MV'	Depends on variable type
FeedForword	FeedForword		FeedForword	Depends on variable type
Option1	Reserve		Reserve	—
Option2	Reserve		Reserve	—
Option3	Reserve		Reserve	—
Option4	Reserve		Reserve	—
MV	Output Value	Output	Output Value 'MV' between 'MaxMV' and 'MinMV'	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Run	○																			
SV														○						
PV														○						
Mode												○								
ManualCtrl	○																			
Cycle												○								
Kp														○						
Ki														○						
Kd														○						
Tf														○						
DIR	○																			
ERR_DBW														○						
MaxMV														○						
MinMV														○						
ManualMV														○						
FeedForword														○						
Option1	○																			
Option2	○																			
Option3	○																			
Option4	○																			
MV														○						

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

- Generally, the set value of the 'Cycle' is the same as the interval of Cycle Task, Please use this instruction in Cycle Task
- When 'Mode' is 0 (PID normal mode), please set the value of 'Kp' according to the experience value and set the values of 'Ki', and' Td' to 0 before manually adjusting 'Kp'、'Ki' and 'Kd'. Wait for the 'Kp' value to be determined and the system to run relatively smoothly, then adjust the values of 'Ki' and 'Kd' from small to large. If 'Kp' is 1(100%), the gain of E I will be 1, if 'Kp' is less than 100%, the gain of E will decrease; if 'Kp' is greater than 100%, the gain of E will increase.
- When 'Mode' is 1(Self-tuning PID mode), please set the type of main Input variables to persistent variable in case that the self-tuning parameters fail to restore to the initial value after a black-out. Self-tuning parameters may not be the optimal parameters in some cases, users can adjust the value of 'Ki' and 'Kd'.

⚠NOTE

- When using the PID instruction, the 'MODE' is usually set to self-tuning mode (Mode=1) first. The self-tuning process will calculate the appropriate value for 'Kp', 'Ki', 'Kd', and 'Td', these values will be automatically written into the input variable. Mode will change to Mode0(PID normal mode) when self-tuning PID mode ends. The input value of the temperature module must be converted to units of °C, then write this value to 'SV'.
- When the value of 'PV' is within the range of 'ERR-DBW', PLC will perform PID calculation according to the deviation (E) value until 'PV' reaches the 'SV' value. At this point, the deviation (E) value will be regarded as 0 in the PID calculation, until the 'PV' value exceeds the range of 'ERR-DBW', then the deviation (E) value will restore.
- The example program is shown below.

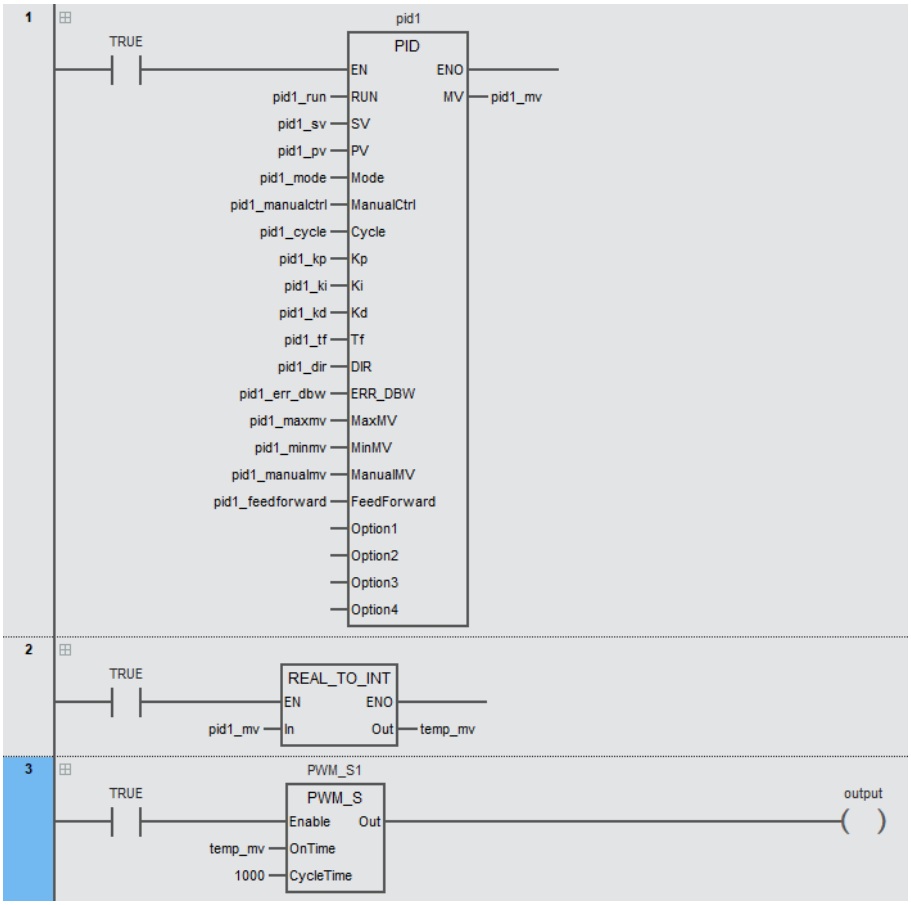
PID temperature adjustment function block:

Write appropriate values for 'Kp', 'Ki', 'Kd', 'Td', etc. in the input variables, set 'EN' to TRUE, set ' Mode ' to 0, and the PID instruction executes based on'Mode', 'ManualCtrl', 'PV', 'SV', 'MaxMV', 'MinMV' etc. The range of the output value 'MV' is between 'MaxMV' and 'MinMV'. Generally, the PWM_S instruction is used to make the PLC output channels to output pulses, which can heat or cool the external devices. The output Value 'MV' can be used to input the variable 'OnTime' of the PWM_S instruction. The input parameter 'Cycle' can be used to Input the variable 'CycleTime' of the PWM_S instruction. When 'MV': = 500, 'Cycle': =1000, the 'CycleTime' of PWM_S instruction is 1000ms, 'OnTime' is 500ms.

■ Variable Table

Category	Name	Assigned to	Date Type	Initial value	Comment
VAR	pid1		PID		
VAR	pid1_en		BOOL		
VAR	pid1_run		BOOL		
VAR	pid1_sv		REAL	180	
VAR	pid1_pv		REAL		
VAR	pid1_mode		DINT	1	
VAR	pid1_manualctrl		BOOL		
VAR	pid1_cycle		DINT	100	
VAR RETAIN	pid1_kp		REAL		
VAR RETAIN	pid1_ki		REAL		
VAR RETAIN	pid1_kd		REAL		
VAR RETAIN	pid1_tf		REAL		
VAR	pid1_dir		BOOL	1	
VAR	pid1_err_dbw		REAL		
VAR	pid1_maxmv		REAL	1000	
VAR	pid1_minmv		REAL	0	
VAR	pid1_manualmv		REAL		
VAR	pid1_feedforward		REAL		
VAR	pid1_mv		REAL		
VAR	pwm_s1		PWM_S		
VAR	temp_mv		REAL		
VAR	output	%QX0.0	BOOL		

LD:



ST:

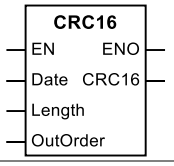
```
pid1(RUN: = pid1_en ,
     SV: = pid1_sv ,
     PV: = pid1_pv ,
     Mode: = pid1_mode ,
     ManualCtrl: = pid1_manualctrl ,
     Cycle: = pid1_cycle ,
     Kp: = pid1_kp ,
     Ki: = pid1_ki ,
     Kd: = pid1_kd ,
     Tf: = pid1_tf ,
     DIR: = pid1_dir ,
```

```
ERR_DBW: = pid1_err_dbw ,

MaxMV: = pid1_maxmv ,

MinMV: = pid1_minmv ,

ManualMV: = pid1_manualmv ,

FeedForward: = pid_feedforward ,

MV=> pid1_mv

);

temp_mv: = REAL_TO_INT(pid1_mv);

pwm_s1(Enable: = TRUE ,OnTime: = pid1_mv, CycleTime: = 1000 ,Out=> output);
```

2.13 CheckSum

2.13.1 CRC16 (CRC16 CheckSum)

This instruction calculates the CRC16 CheckSum of an array. Library: Standard_Part2

Instruction	Name	FB/FUN	Graphic expression	ST expression
CRC16	CRC16 CheckSum	FUN		CRC16: =CRC16(Data , Length , OutOrder);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Data	Data pointer	Input	Data pointer	-
Length	Number of elements		Number of elements to process (unit: byte)	1 ~ Maximum length of Data
OutOrder	Byte order		FALSE: Upper byte is in front and followed by the lower byte TRUE: Lower byte is in front and followed by the upper byte last	TRUE or FALSE
CRC16	CRC16 CheckSum	Output	CRC16 CheckSum	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				Pointer
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	Pointer To BYTE
Data																				○
Length							○													
OutOrder	○																			
CRC16			○																	

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

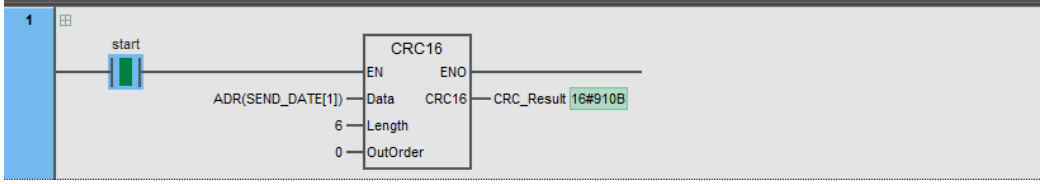
- The CRC16 instruction calculates the CRC16 CheckSum of an array

- The example program is shown below.

LD:

When an array is {16#01,16#03,16#10,16#01,16#00,16#02}，Length is 6，CRC16 is '16#910B'.

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	SEND_DATE		ARRAY[1..6] OF BYTE		
2	VAR	SEND_DATE[1]		USINT	16#01	
3	VAR	SEND_DATE[2]		USINT	16#03	
4	VAR	SEND_DATE[3]		USINT	16#10	
5	VAR	SEND_DATE[4]		USINT	16#01	
6	VAR	SEND_DATE[5]		USINT	16#00	
7	VAR	SEND_DATE[6]		USINT	16#02	
8	VAR	CRC_Result		WORD	16#910B	
9	VAR	start		BOOL	TRUE	

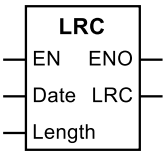


ST:

```
CRC_Result: =CRC16( ADR( SEND_DATE[1] ) , 6 , 0 );
```

2.13.2 LRC (LRC CheckSum)

This instruction calculates the LRC CheckSum of an array. Library: Standard_Part2.

Instruction	Name	FB/FUN	Graphic expression	ST expression
LRC	LRC CheckSum	FUN		LRC: = LRC(Data , Length);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Data	Data pointer	Input	Data pointer	-
Length	Number of elements		Number of elements to process (unit: byte)	1 ~ Maximum length of Data
LRC	LRC CheckSum	Output	LRC CheckSum	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time, date				Pointer
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	Pointer To BYTE
Data																				○
Length							○													
LRC			○																	

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

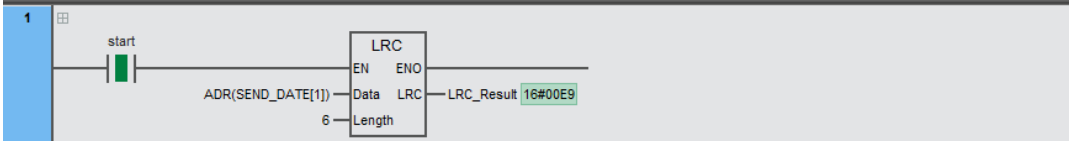
- The LRC instruction calculates the LRC CheckSum of an array

- The example program is shown below.

LD:

When an array is {16#01,16#03,16#10,16#01,16#00,16#02}，Length is 6，CRC16 is '16#E9'.

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	SEND_DATE		ARRAY[1..6] OF BYTE		
2	VAR	SEND_DATE[1]		USINT	16#01	
3	VAR	SEND_DATE[2]		USINT	16#03	
4	VAR	SEND_DATE[3]		USINT	16#10	
5	VAR	SEND_DATE[4]		USINT	16#01	
6	VAR	SEND_DATE[5]		USINT	16#00	
7	VAR	SEND_DATE[6]		USINT	16#02	
8	VAR	LRC_Result		WORD	16#00E9	
9	VAR	start		BOOL	TRUE	



ST:

LRC_Result: =LRC16(ADR(SEND_DATE[1]) , 6);

2.14 Bit-to-word conversion and word-to-bit conversion

2.14.1 GetBitofWord (read the values of the specified bit in the variable)

This instruction reads the values of the specified bit in a bit string or Integers variable. Library: Standard_Part2

Instruction	Name	FB/FUN	Graphic expression	ST expression
GetBitofWord	Read the values of the specified bit in the variable	FUN		GetBitofWord(StartDevice , Offse);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
StartDevice	Data pointer	Input	Data pointer	-
Offset	Bit offset		Bit position to read	0~1023
GetBitofWord	Bit status	Output	Bit status	TRUE or FALSE

	Boolean	Bit string				Integer								Real number		Time, date				Pointer
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	Pointer To UINT
StartDevice																				○
Offset			○				○													
GetBitofWord	○																			

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

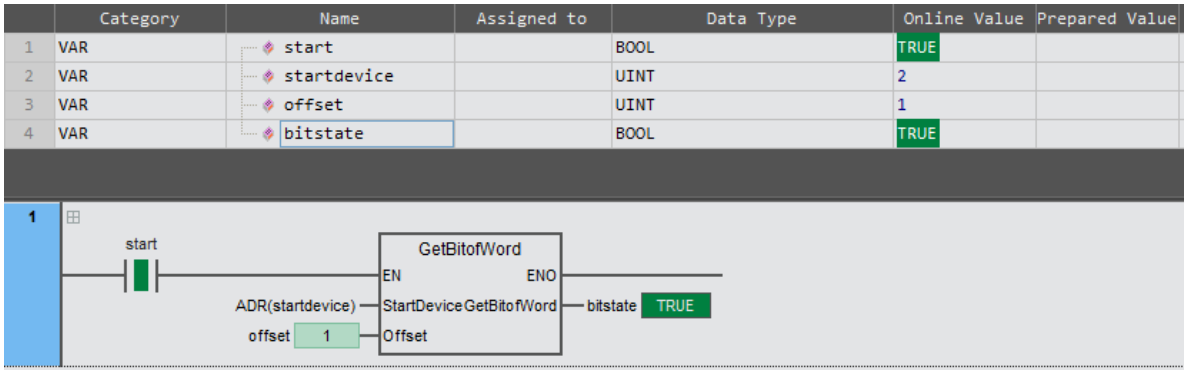
■ Function description

- The GetBitofWord instruction reads the values of the specified bit in a bit string or Integer variable.
- The variable type of 'StartDevice' is 'Pointer to UINT', Ordinary variables cannot be directly used as input variables without the operator 'ADR'.
- 'Offset' is the bit position to read, 0 is Bit0, 1 is Bit1, and so on.

- The example program is shown below.

LD:

Read the values of the bit1 in 'StartDevice', when 'StartDevice' is 2, 'Offset' is 1, the value of GetBitofWord will be TRUE; When 'StartDevice' is 0, 'Offset' is 1, the value of GetBitofWord will be FALSE;



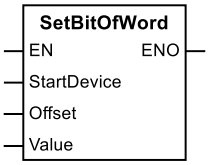
ST:

bitstate: =GetBitofWord(ADR(startDevice) , 1);

Set the variable type of the 'StartDevice ' to ARRAY [1.. 10] of UINT when users need to read larger bit states, for example, when 'offset' is 15. Set 'StartDevice' to 'ADR(startDevice)', and change the value of 'Offset' to read the value specified in the array.

2.14.2 SetBitofWord(set the values of the specified bit in the variable)

This instruction sets the values of the specified bit in a bit string or Integers variable. Library: Standard_Part2

Instruction	Name	FB/FUN	Graphic expression	ST expression
SetBitofWord	Set the values of the specified bit in the variable	FUN		SetBitOfWord(StartDevice , Offset , Value);

■ Input/Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
StartDevice	Data pointer	Input	Data pointer	-
Offset	Bit offset		Bit position to set	0~1023
Value	Set status		Set status	TRUE or FALSE

	Boolean	Bit string				Integer								Real number		Time, date				Pointer
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	Pointer To UINT
StartDevice																				○
Offset			○				○													
Value	○																			

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to connect with the variables or constants of the data type.

■ Function description

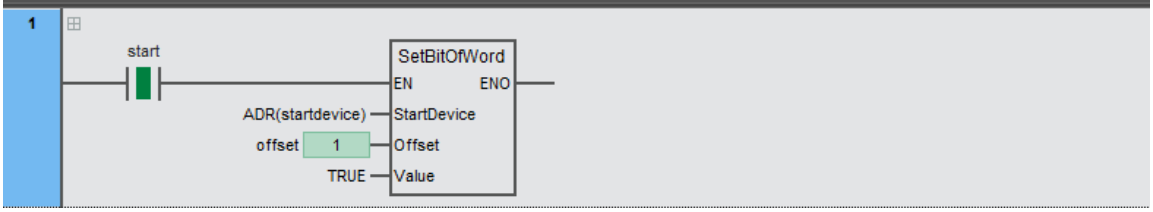
- The SetBitofWord instruction sets the values of the specified bit in a bit string or Integer variable. The set value is specified by 'Value', which can be either TRUE or FALSE.
- The variable type of 'StartDevice' is 'Pointer to UINT', and ordinary variables cannot be directly used as input variables without the operator 'ADR'.
- 'Offset' is bit position to set, 0 is Bit0, 1 is Bit1, and so on.

- The example program is shown below.

LD:

Set the values of the bit1 in 'StartDevice': When 'StartDevice' is 1, 'Offset' is 1, 'Value' is TRUE, the bit1 of 'StartDevice' is TRUE after the instruction is executed. When 'StartDevice' is 2, 'Offset' is 1, 'Value' is FALSE, and the bit1 of 'StartDevice' is FALSE after the instruction is executed.

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
1	VAR	start		BOOL	TRUE	
2	VAR	startdevice		UINT	2	
3	VAR	offset		UINT	1	
4	VAR	bitstate		BOOL	FALSE	



ST:

Bit Status: =GetBitofWord(StartDevice: = ADR(Parameters) , Offset: = Offset).

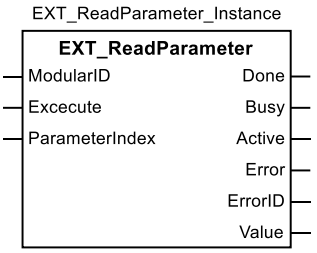
SetBitOfWord(StartDevice: = ADR(Parameters) , Offset: = Offset , Value: = Bit Setting Value)

Set the variable type of the 'StartDevice ' to ARRAY [1.. 10] of UINT when users need to set larger bit states, for example, when 'offset' is 15. Set 'StartDevice' to 'ADR(startDevice)', change the value of 'Offset', to set the value specified in the array.

2.15 Expansion Module Communication

2.15.1 EXT_ReadParameter(Read the expansion module parameter)

This instruction is used to read the parameter value of the module. Library: Standard_Part2.

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
EXT_ReadParameter	Read the parameter of the expansion module	FB		EXT_ReadParameter_ Instance (ModularID:= Parameter , Execute:= Parameter , ParameterIndex := Parameter, Done=> Parameter, Busy=> Parameter , Active=> Parameter , Error=> Parameter , ErrorID=> Parameter , Value=> Parameter);

■ **Input / Output variable instructions and data types**

Name	Meaning	I/O	Description	Parameter scope
ModularID	Module serial number	Input	Module serial number	1-32
Execute	Execution bits		Execution bits	TRUE or FALSE
ParameterIndex	Parameter number		Module parameter number	Depends on module parameter scope supported
Done	Finish bits	Output	Finish bits	TRUE or FALSE
Busy	Execution status bits		Execution status bits	TRUE or FALSE
Active	Activate bits		Activate bits	TRUE or FALSE
Error	Error bits		Error bits	TRUE or FALSE
ErrorID	Error codes		Error codes	0~65535
Value	Read the numeric value		Read the numeric value	Depends on variable type

	Boolean	Bit string				Integer								Real number		Time，Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ModularID						○														
Execute	○																			
Index						○														
Done	○																			
Busy	○																			
Active	○																			
Error	○																			
ErrorID			○																	
Value			○																	

*** Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this

data type.

■ Error code description

If an instruction executes incorrectly, the ErrorID (error code) will have a corresponding value. Refer to the following sheet for the meaning of the ErrorID value and handling measures.

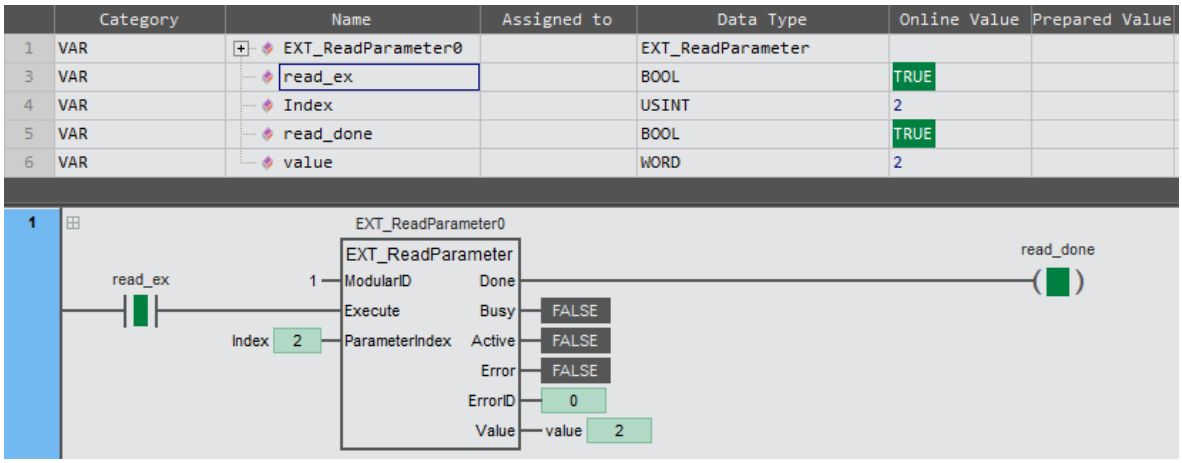
Error code		Meaning of the ErrorID	Handling measures
Hexadecimal system	Decimal system		
4000	16384	The number of expansion modules exceeds the scope	Set the expansion module number(modularID)to the allowed range
4001	16385	The communication between the host and the expansion module has timed out	Check and ensure that the host connects with the expansion module properly
4002	16386	The host reads or writes abnormally to module parameters	1. Check whether the module parameter number(ParameterIndex) exists 2. Check and ensure that the host connects with the expansion module properly 3. Avoid exposing the product to a highly disturbing environment

■ Function description

- This instruction is used to read the parameter value of the module.
- The first module on the right of the controller is numbered 1 and the second one is numbered 2, and so on
- Please refer to the specific module manual for the meaning of the parameter number
- The example program is shown below.

LD:

Set the parameter value of the first module on the right of the controller whose parameter number is 2, and the read value is 1.

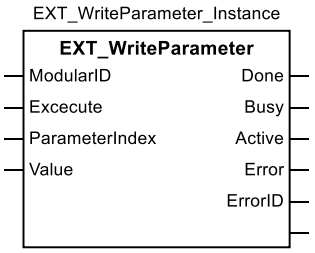


ST:

```
EXT_ReadParameter0（
ModularID: =1 ,
Execute: = read_ex ,
ParameterIndex: = Index ,
Done=> read_done ,
Value=> value
);
```

2.15.2 EXT_WriteParameter (Set the expansion module parameter)

This instruction is used to set the parameter value of the module. Library: Standard_Part2

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
EXT_WriteParameter	Set the parameter of the expansion module	FB		EXT_WriteParameter_ ModularID:= Parameter , Execute:= Parameter , ParameterIndex := Parameter, Value:= Parameter Done=> Parameter, Busy=> Parameter , Active=> Parameter , Error=> Parameter , ErrorID=> Parameter ,);

■ **Input / Output variable instructions and data types**

Name	Meaning	I/O	Description	Parameter scope
ModularID	Module serial number	Input	Module serial number	1-32
Execute	Execution bits		Execution bits	TRUE or FALSE
ParameterIndex	Parameter number		Module parameter number	Depends on module parameter scope supported
Value	Finish bits		Read the numeric value	Depends on variable type
Done	Execution status bits	Output	Finish bits	TRUE or FALSE
Busy	Activate bits		Execution status bits	TRUE or FALSE
Active	Error bits		Activate bits	TRUE or FALSE
Error	Error codes		Error bits	TRUE or FALSE
ErrorID	Read the numeric value		Error codes	0~65535

	Boolean	Bit string				Integer								Real number		Time，Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ModularID						○														
Execute	○																			
Index						○														
Value			○																	
Done	○																			
Busy	○																			
Active	○																			
Error	○																			
ErrorID			○																	

*** Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Error code description

If an instruction executes incorrectly, the ErrorID (error code) will have a corresponding value. Refer to the following sheet for the meaning of the ErrorID value and handling measures.

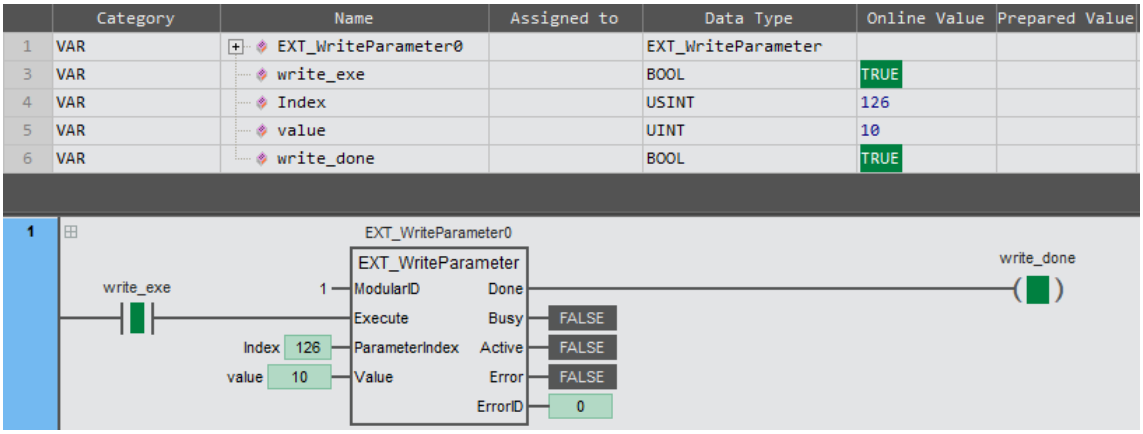
Error code		Meaning of the ErrorID	Handling measures
Hexadecimal system	Decimal system		
4000	16384	The number of expansion modules exceeds the scope	Set the expansion module number(modularID)to the allowed range
4001	16385	The communication between the host and the expansion module has timed out	Check and ensure that the host connects with the expansion module properly
4002	16386	The host reads or writes abnormally to module parameters	1. Check whether the module parameter number (ParameterIndex) exists. 2. Whether the value (set parameter) exceeds the allowable range of the module parameters. 3. Check and assure that the host connect with the expansion module properly. 4. Avoid exposing the product to a highly disturbing environment.

■ Function description

- This instruction is used to read the parameter value of the module.
- The first module on the right of the controller is numbered 1 and the second one is numbered 2, and so on
- Please refer to the specific module manual for the meaning of the parameter number
- The example program is shown below.

LD:

Set the parameter value for the first module on the right side of the controller, parameter number 126, to a set value of 10.



ST:

```
EXT_WriteParameter0(
    ModularID: =1 ,
    Execute: = write_ex ,
    ParameterIndex: = Index ,
    Value: = value ,
    Done=> write_done
    );
```

2.16 System Functions

2.16.1 SYS_GetTotalWorkTime（Cumulative power-on time）

This instruction is used to read the cumulative power-on time. Library: Standard_Part2.

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
SYS_GetTotalWorkTime	Cumulative Power-on Time	FB	SYS_GetTotalWorkTime_Instance SYS_GetTotalWorkTime Enable Valid WorkTime	SYS_GetTotalWorkTime_ Instance(Enable:=Parameter, Valid=>Parameter, WorkTime=>Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Enable	Execution bits	Input	Execution bits	TRUE or FALSE
Valid	Execution status	Output	Execution status	TRUE or FALSE
WorkTime	Cumulative power-on time		Cumulative power-on time	Depends on variable type

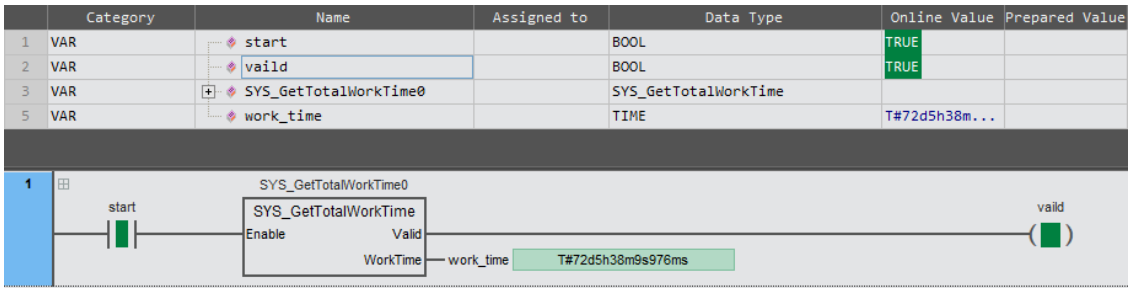
	Boolean	Bit string				Integer								Real number		Time，Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	○																			
Valid	○																			
WorkTime																○				

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function descriptions

- This instruction is used to gain the cumulative working time of the controller since it left the factory.
- The example program and the timing diagram are shown below.

LD:



ST:

```
SYS_GetTotalWorkTime0(Enable: = start , Valid=> valid , WorkTime=> work_time );
```

2.16.2 SYS_GetWorkTime（Single power-on time）

This instruction is used to read the power-on time after each power-up of the controller. Library: Standard_Part2

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
SYS_GetWorkTime	Single power-on time	FB	SYS_GetWorkTime_Instance SYS_GetWorkTime Enable Valid WorkTime	SYS_GetWorkTime_ Instance(Enable:=Parameter, Valid=>Parameter, WorkTime=>Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Enable	Execution bits	Input	Execution bits	TRUE or FALSE
Valid	Execution status	Output	Execution status	TRUE or FALSE
WorkTime	Power-on time		Power-on time	Depends on variable type

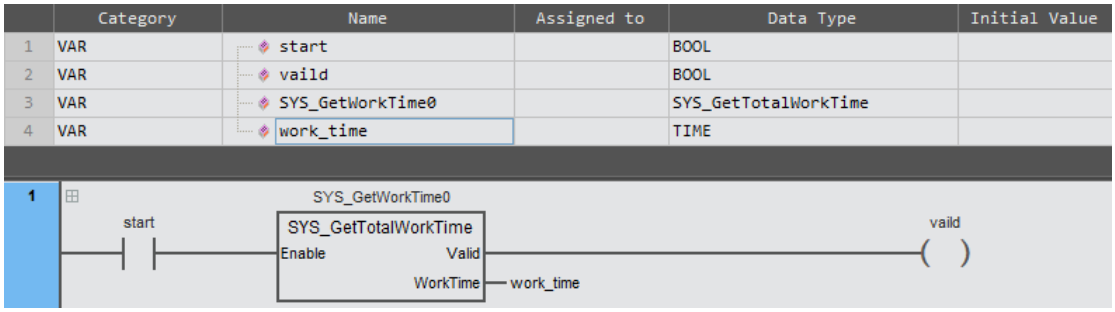
	Boolean	Bit string				Integer								Real number		Time，Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	○																			
Valid	○																			
WorkTime																○				

***Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- This instruction is used to gain the single power-on working time of the controller, this time is timed from 0 every time the controller is power-on.
- The example program is shown below.

LD:



ST:

```
SYS_GetWorkTime0 (Enable: = start , Valid=> valid , WorkTime=> date_time );
```

2.16.3 SYS_GetRTCTime (Read the real-time clock)

This instruction is used to read the real-time clock time of the controller. Library: Standard_Part2.

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
SYS_GetRTCTime	Read the real-time clock time	FB	SYS_GetRTCTime_Instance SYS_GetRTCTime Enable Valid Year Month Day Week Hour Minute Second	SYS_GetRTCTime_ Instance(Enable:=Parameter , Valid =>Parameter, Year =>Parameter , Month =>Parameter , Day =>Parameter , Week =>Parameter , Hour =>Parameter , Minute =>Parameter, Second =>Parameter);

■ Input / Output variable instructions and data types

Name	Meaning	I/O	Description	Parameter scope
Enable	Execution bits	Input	Execution bits	TRUE or FALSE
Valid	Execution status	Output	Execution status	TRU or FALSE
Year	Year		Year	1970-2106
Month	Month		Month	1~12
Day	Day		Day	1~31
Week	Week		Week	1~7
Hour	Hour		Hour	0~23
Minute	Minute		Minute	0~59
Second	Second		Second	0~59

	Boolean	Bit string				Integer								Real number		Time，Date				String
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	○																			
Valid	○																			
Year							○													
Month							○													
Day							○													
Week							○													
Hour							○													
Minute							○													
Second							○													

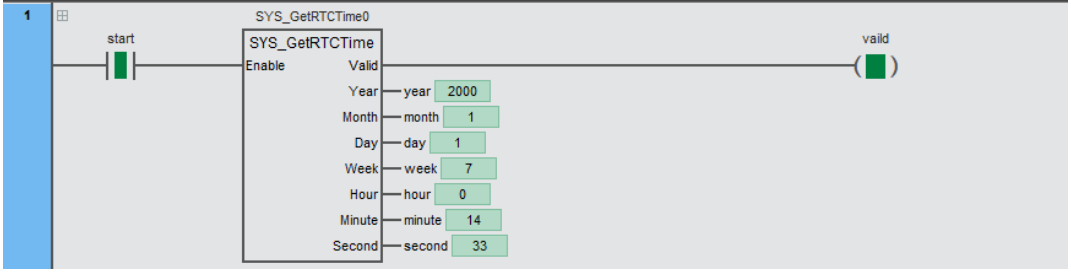
*** Note:** The '○' in the above table indicates that the instruction parameters are allowed to be connected to variables or constants of this data type.

■ Function description

- This instruction is used to gain the real-time clock time of the controller itself, please note that the real-time clock is stored by the battery or the capacitor of the body to continue the timing.
- The example program is shown below.

LD:

	Category	Name	Assigned to	Data Type	Online Value	Prepared Value
2	VAR	SYS_GetRTCTime0		SYS_GetRTCTime		
4	VAR	vaild		BOOL	TRUE	
5	VAR	year		UINT	2000	
6	VAR	month		UINT	1	
7	VAR	day		UINT	1	
8	VAR	week		UINT	7	
9	VAR	hour		UINT	0	
10	VAR	minute		UINT	14	
11	VAR	second		UINT	33	



ST:

```
SYS_GetRTCTime0（
    Enable: = start ,
    Valid=> vaid ,
    Year=> year ,
    Month=> month ,
    Day=> day ,
    Week=> week ,
    Hour=> hour ,
    Minute=> minute ,
    Second=> second
);
```

HCFA

创新 诚信 服务

HCFA

HCFA_ATC