AYDIN
ADNAN MENDERES UNIVERSITY
FACULTY OF ENGINEERING
Department of Electrical and Electronics Engineering

EE441 – Programmable Logic Controller (PLC)

Prof. Dr. Yılmaz KALKAN



- Basic of Industrial Automation
- Understanding Automation & Control System
- History of PLC
- Basic Components of Automation
- Hardware / Software Classification Of Automation

outline

- Introduction to PLC
- Need of PLC in Designing
- Architecture of PLC
- Application and Advantage of PLC
- Automation Concept And Basic Design
- NO/ NC Concept

- PLC Programming
  - Introduction of Ladder Logic
  - Introduction of SFC
  - Introduction of Instruction list
- Creating & Editing Ladder Logic Program
- Different Types Of Sensors-Sinking, Sourcing, NPN, PNP
- PLC Fault Finding, Trouble Shooting
- Forcing I/O's
- Detailed Sequence of Wiring
- PLC Control Panel Wiring
- Interfacing With Relay Logics

PLC
Programmable Logic Controller

Mitsubishi FA e-learning – PLC Programming Basic (Ladder)

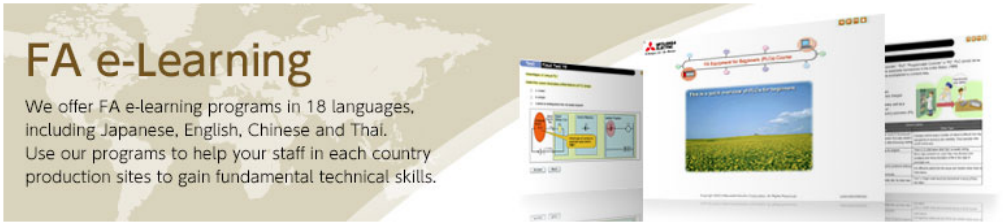
Support

Font Size : S M L ▸ Print This Page

- Education and support
- e-learning courses
  - Japanese
  - English
  - Chinese
  - Korean
  - Thai
  - Indonesian
  - Vietnamese
  - Malay
  - Myanmar
  - Spanish (for Latin America)
  - Portuguese (for South America)
  - Russian
  - Polish
  - Czech
  - Slovakian
  - Hungarian
  - Turkish
  - Hindi
- Dictionaries
- Glossaries
- Phrasebook
- On-site lexicon
- Satellite Training Series

Inquiries

e-learning courses (English)



The PC/Tablet edition requires a screen resolution of XGA (1024 x 768) or higher.
The PC edition can be downloaded by clicking on the following icon.
For further information about methods of expanding data in the PC edition, refer to "Using the PC edition data" located at the bottom of the page.

▾ Using the PC edition data

English

Courses		PC	PC/Tablet	PDF
FA Equipment for Beginners	PLCs NEW	-	✓	
	HMIs	↓	✓	
	Inverters	-	✓	
	GX Works2 Basics	-	✓	
	GX Works2 Advanced	-	✓	
	GX Works3 (Ladder)	-	✓	
	PLC Programing Basic (ST)	-	✓	
	PLC Programing Basic (Ladder)	-	✓	

PLC

Programming Basics

This course is for participants who will create control programs for programmable controllers for the first time.

Introduction Purpose of the Course

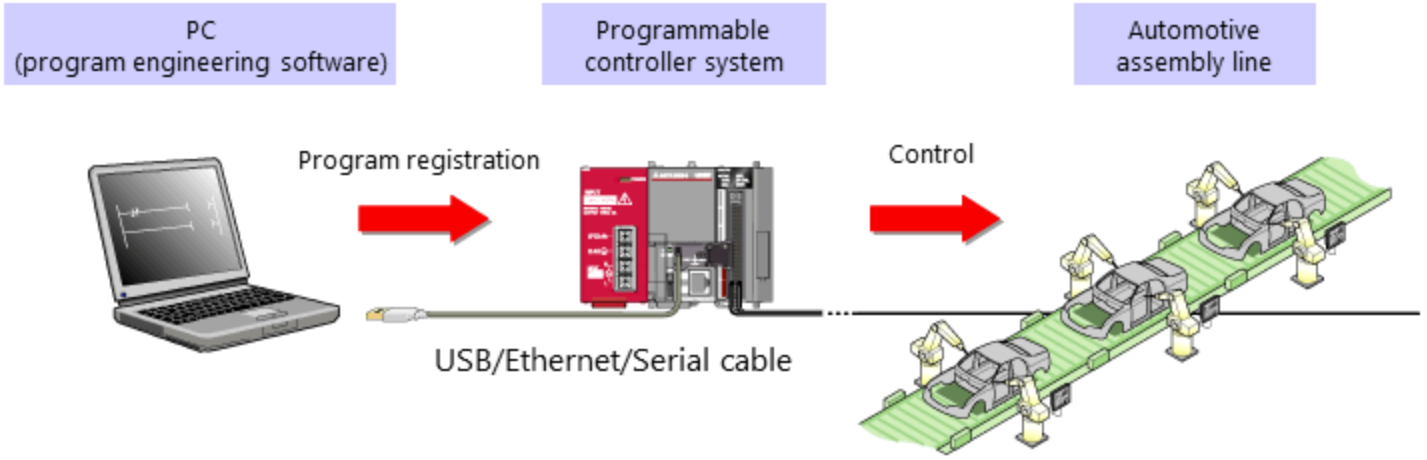
This course explains about programming, which could be used for the MELSEC programmable controllers.
One of the main programming languages is Ladder Diagram (LD). This course covers essential points of ladder programming including its main instructions.

Some sections of this course are based on the basic courses of the MELSEC programmable controllers. It is recommended to take the relevant basic course prior to taking this course.

Chapter 1 Control program

Operations executed by a programmable controller are written as control programs.
These programs are registered into the CPU module, controlling various input and output (I/O) signals.
Programming languages that are used for programmable controllers include Ladder, Instruction List (IL), and Sequential Function Chart (SFC).

This course explains essential points of ladder programming including its main instructions.

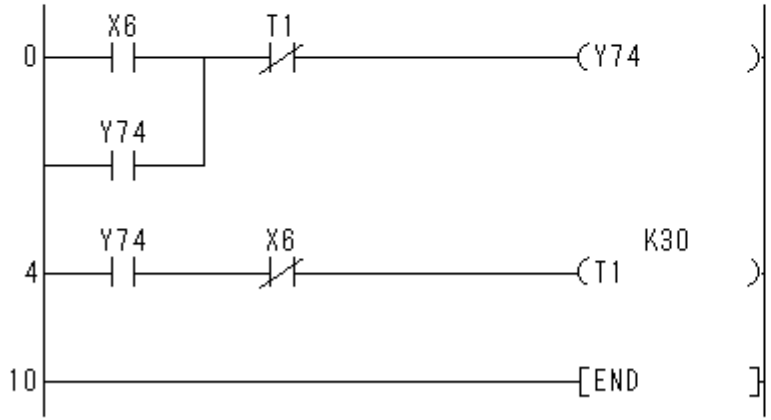


In this course, the programmable controller engineering software, GX Works2 or GX Works3, is used to create programs.
To learn how to use the programmable controller engineering software, please take the "GX Works2 Basics" or the "Engineering Software MELSOFT GX Works3 (Ladder)" course.
GX Works2 supports MELSEC-Q/L/F Series.
GX Works3 supports MELSEC iQ-R/iQ-F Series.

1.1 Programming languages

Programming languages that are used for programmable controllers include Ladder, Instruction List (IL), and Sequential Function Chart (SFC).

A ladder program is a graphical logic diagram based on an electrical circuit. In ladder programs, symbols representing instructions are connected with lines, similar to a circuit diagram, and operation flows are easily recognizable. Additionally, ladder programming does not require special programming knowledge such as with C and BASIC languages, and can be easily understood by those who have experience in working with electrical circuits and relays.



The table below shows the same program in IL.
IL requires some programming knowledge to express operations in the form of instructions.

Step No.	Instruction	Device
0	LD	X6
1	OR	Y74
2	ANI	T1
3	OUT	Y74
4	LD	Y74
5	ANI	X6
6	OUT	T1 K30
10	END	

1.2 Values used in programs

Programs for programmable controllers can handle two types of values.

Bit	Bit is expressed in two types of electric signals, ON and OFF. These can also be expressed as "1" (ON) and "0" (OFF). Bit values are often used to show statuses of I/O devices such as switches and lamps.
Word	Numbers and characters. Word values are often used to show quantity and time. *This course will only explain about numbers. For the details of characters used as word values, please refer to the relevant product manual.

The following numerical formats are used to show values.

- Decimal
- Binary
- Hexadecimal
- Octal

1.2.1 Decimal notation

In decimal notation, the magnitude (quantity) of a number is represented using a base 10 format from "0 to 9".

In the MELSEC programmable controllers, decimal numbers are preceded by the letter "K".
For example, "K153" represents the decimal number "153".

1.2.2 Binary notation

While decimal notation is normally used to express quantities and time, programmable controllers and personal controllers use binary data, which are combinations of "0" and "1".
The table below shows correspondence between decimal and binary values, up to the decimal number "8".

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
⋮	⋮

Whenever an instruction of 1 word is used in a program, it would be saved and processed as 16-bit binary data by the actual programmable controller. This 16-bit binary data is synonym for "1 word".
For example, decimal "157" is expressed as "0000000010011101" in binary.

In decimal notation, bits are written from the right. (Rightmost bit is the start bit.)

b15	～	b8	b7	b6	b5	b4	b3	b2	b1	b0	← Bit position
0	・・・	0	1	0	0	1	1	1	0	1	← Binary
2^{15}		2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	← Power of 2
32768		256	128	64	32	16	8	4	2	1	← Weight for decimal values

To convert binary values to decimal, multiply each bit status ("0" or "1") by the corresponding weight, and add up all the products.

$= \underline{1\times128} + 0\times64 + 0\times32 + \underline{1\times16} + 1\times8 + \underline{1\times4} + 0\times2 + \underline{1\times1}$
$= 128 + 16 + 8 + 4 + 1$
$= 157$

From above, binary can be considered as a numbering format that is based on weights.

1.2.3 Hexadecimal notation

In hexadecimal notation, the magnitude (quantity) of a number is represented using base 16 or indicated using 16 alphanumeric characters: 0 to 9 and A to F. Each digit in hexadecimal notation increases as 0, 1...9, A...E, then F. When the value exceeds the radix "F", the one is carried to the left and becomes "10".
In the MELSEC programmable controllers, hexadecimal numbers are preceded by "H" .
For example, "H4A9D" represents the hexadecimal number "4A9D".

Binary notation could be long and difficult to be used in programs and on monitor displays.
In such a case, hexadecimal notation is useful.
One digit of a hexadecimal value can express 4 bits (4 digits) of binary values.

The figure below shows how a hexadecimal value is expressed as a decimal value.

4	3	2	1	←Digit number
4	A	9	D	←Hexadecimal value
16^3	16^2	16^1	16^0	←Power of 16

$$= 4 \times 16^3 + A \times 16^2 + 9 \times 16^1 + D \times 16^0$$
$$(4096)\ (10)\ (256)\qquad (16)\ (13)\ (1)$$
$$= 19101$$

* One digit of a hexadecimal value can express 4 bits of binary.

Decimal	Binary	Hexadecimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
•	•	•
•	•	•
•	•	•
19101	0100 1010 1001 1101	4 A 9 D

1.2.4 Octal notation

In octal notations, the magnitude (quantity) of a number is represented using a base 8 format ("0 to 7").
When the value increases from "0", "1", "2" to "7", the one is carried to the left and becomes "10".
This octal notation is used for the MELSEC iQ-F/F Series I/O numbers.

Decimal	Binary	Hexadecimal	Octal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
16	1101	D	15
14	1110	E	16
15	1111	F	17
16	1000	10	20
17	10001	11	21
18	10010	12	22
.	.	.	.
.	.	.	.
.	.	.	.
26	011 010	1A	32

The figure below shows how an octal value is expressed as a decimal value.

2	1	←Digit number
3	2	←Octal number
8^1	8^0	←Power of 8

$$= 3 \times 8^1 + 2 \times 8^0$$
$$(8) \quad (1)$$
$$= 26$$

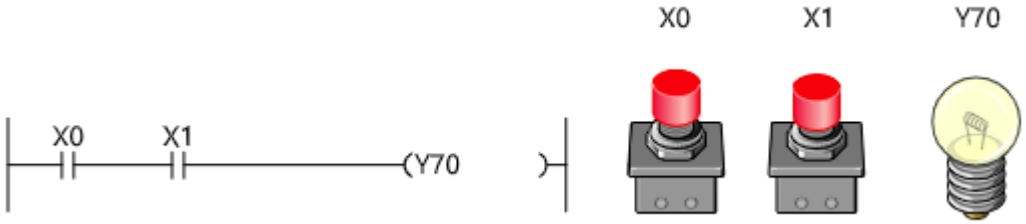
* One digit of an octal value can express 3 bits of binary.

1.3 PLC programming basics

In sequential control, a series of operations are executed based on the ON/OFF signals received from equipment connected to an input module, and then the operation results are output to equipment connected to an output module.

To perform such a control, a control program must have input conditions and outputs, which will be executed at satisfactory input conditions.

The program below instructs the following operations:
- When the push-button switches connected to terminals X1 and X2 are both ON, turn ON the terminal Y70
- The operation result is output to the terminal Y70 to turn the connected lamp ON



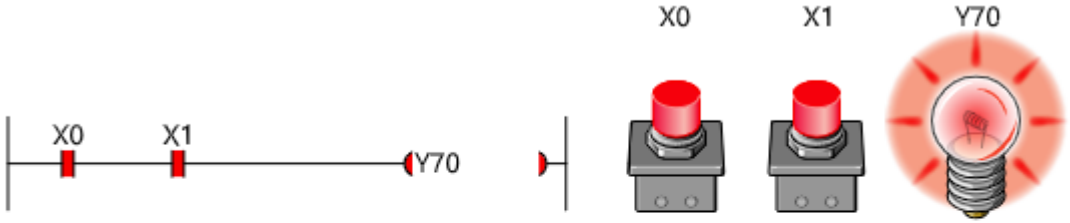
If only X0 or X1 is pushed down, Y70 does not turn ON.

1.3 PLC programming basics

In sequential control, a series of operations are executed based on the ON/OFF signals received from equipment connected to an input module, and then the operation results are output to equipment connected to an output module.

To perform such a control, a control program must have input conditions and outputs, which will be executed at satisfactory input conditions.

The program below instructs the following operations:
- When the push-button switches connected to terminals X1 and X2 are both ON, turn ON the terminal Y70
- The operation result is output to the terminal Y70 to turn the connected lamp ON



Pushing down the switches X0 and X1 together turns Y70 lamp ON.

1.4 I/O numbers and devices

The programs described in Chapter 1.3 used alphanumeric symbols such as X0, X1, and Y70 to identify I/O equipment. These alphanumeric characters are called I/O numbers.

This chapter explains about I/O numbers and devices, which are required for creating control programs.

MELSEC iQ-R/Q/L/iQ-F Series and MELSEC-F Series use different formats in expressing device numbers. The table below summarizes the differences.

MELSEC Series	Bit			Word
	X (input number)	Y (output number)	M (internal relay)	D (data register)
iQ-R/Q/L Series	Hexadecimal	Hexadecimal	Decimal	Decimal
iQ-F/F Series	Octal	Octal	Decimal	Decimal

1.4.2 I/O numbers and I/O signals (MELSEC iQ-F/F Series)

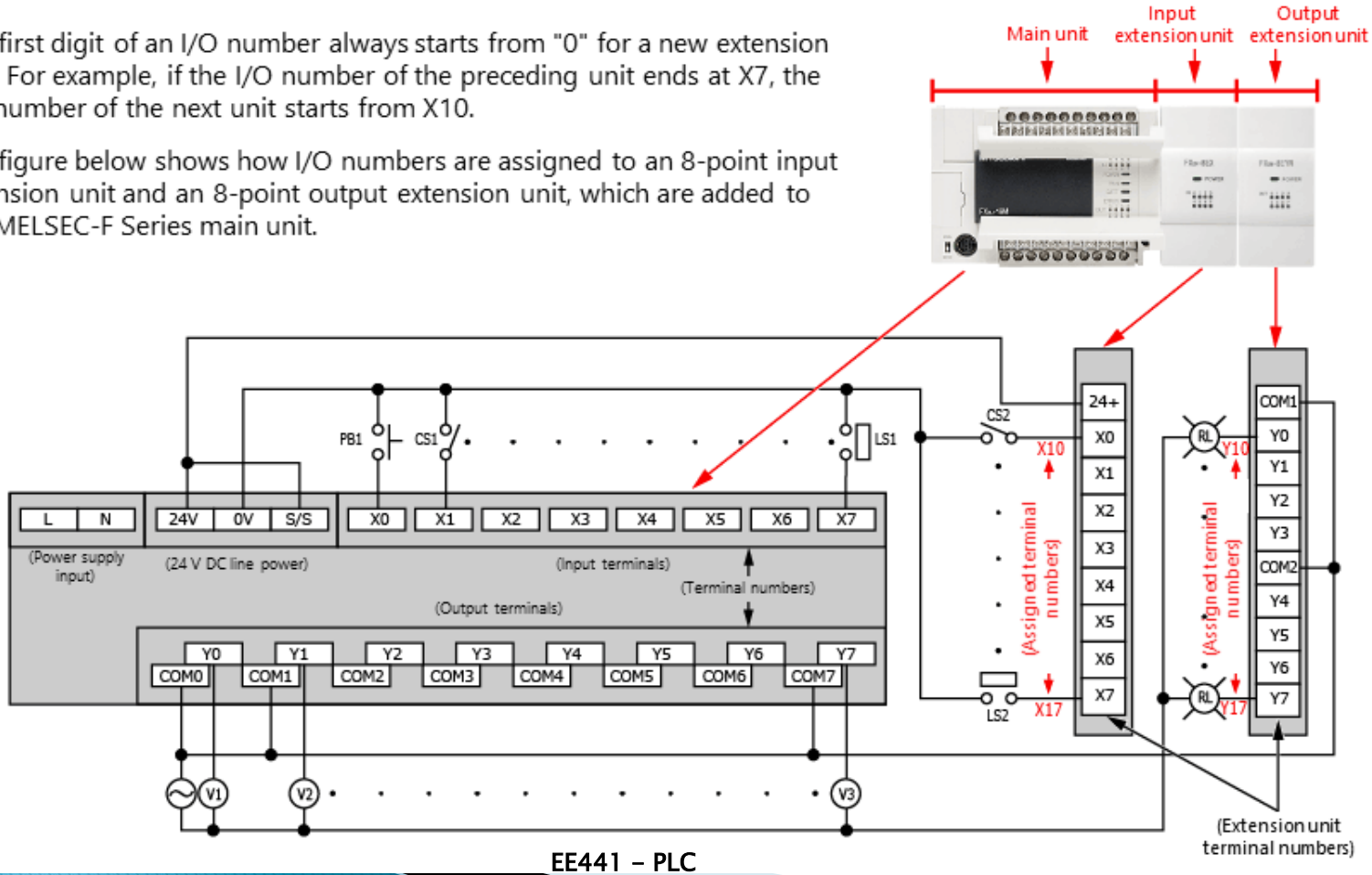
MELSEC iQ-F/F Series

An I/O number consists of an alphabet, which indicates input (X) or output (Y), and an octal value that represents a terminal number.

An I/O number is first determined based on the end I/O number of the main unit or the preceding I/O extension block. The range of I/O numbers is then determined based on the number of occupied I/O points of the unit.
(The number of occupied I/O points is proportional to the number of I/O points held by the I/O extension unit.)

The first digit of an I/O number always starts from "0" for a new extension unit. For example, if the I/O number of the preceding unit ends at X7, the I/O number of the next unit starts from X10.

The figure below shows how I/O numbers are assigned to an 8-point input extension unit and an 8-point output extension unit, which are added to the MELSEC-F Series main unit.

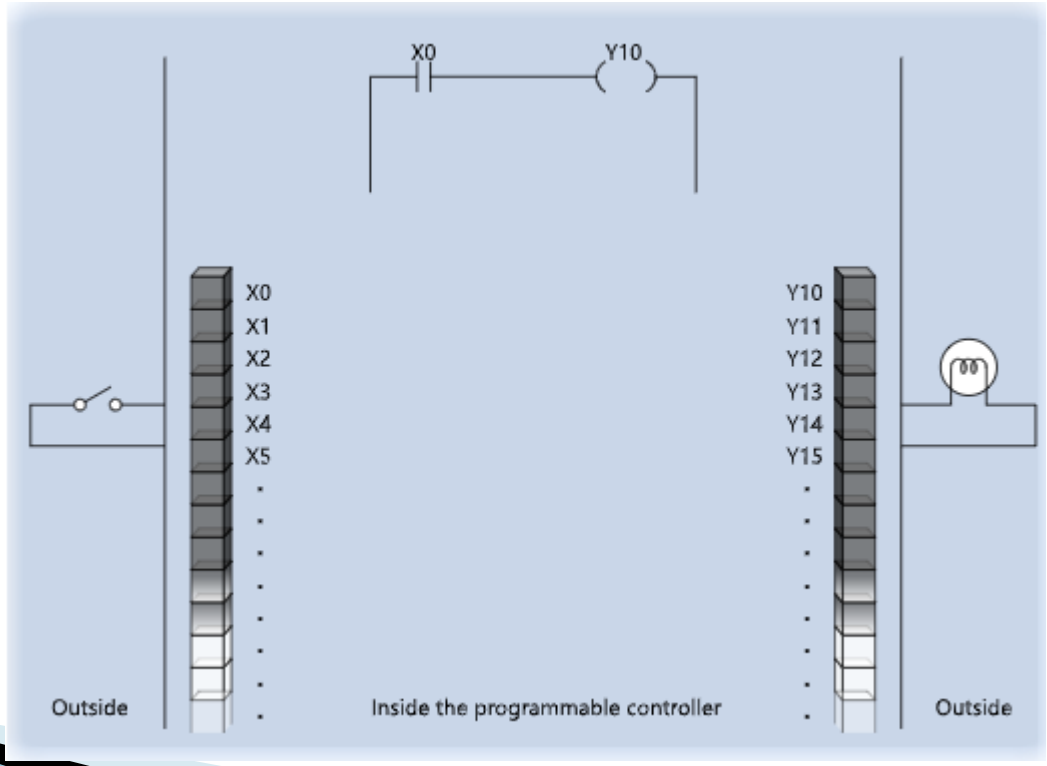


1.4.3 I/O numbers and devices

Statuses of equipment connected to the unit are saved in the programmable controller memory area called "devices". Likewise for the output, output equipment operates in accordance with the statuses of the devices. As explained above, control programs are often executed based on the device statuses.

Devices that store bit information (ON/OFF) such as input (X) and output (Y) are called "bit devices".

The device numbers correspond with I/O numbers.
For example, the status of the terminal assigned with I/O number X0 is saved in the device X0.
Likewise, the status of the device Y10 corresponds with the terminal assigned with the I/O number Y10.

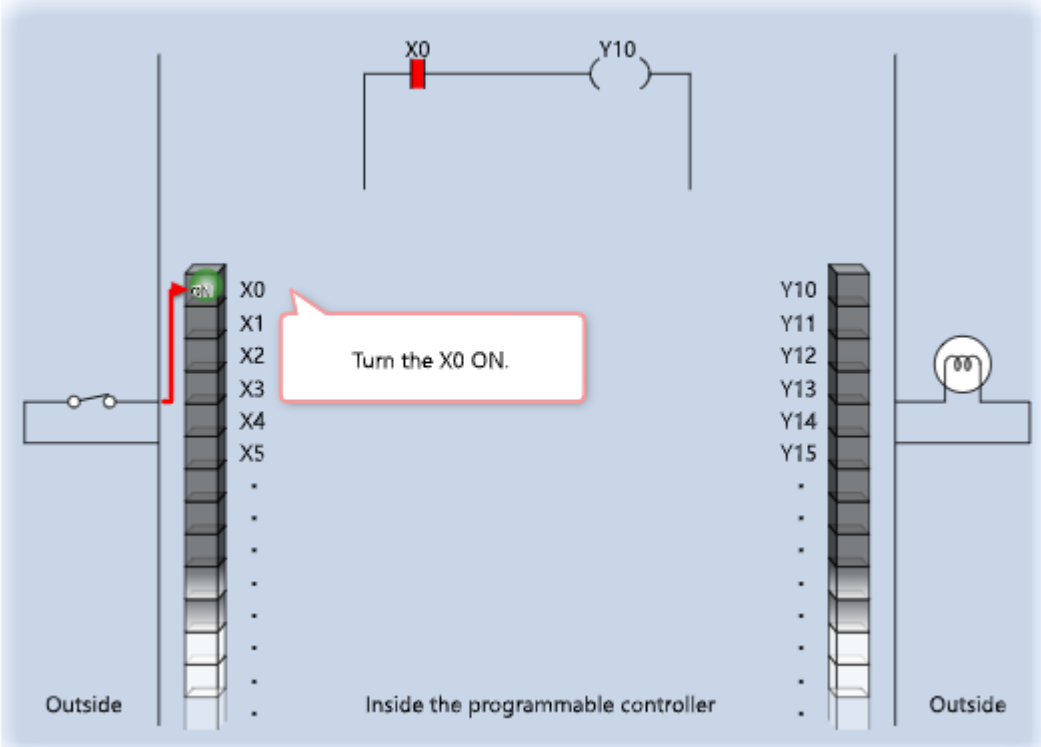


1.4.3 I/O numbers and devices

Statuses of equipment connected to the unit are saved in the programmable controller memory area called "devices". Likewise for the output, output equipment operates in accordance with the statuses of the devices. As explained above, control programs are often executed based on the device statuses.

Devices that store bit information (ON/OFF) such as input (X) and output (Y) are called "bit devices".

The device numbers correspond with I/O numbers.
For example, the status of the terminal assigned with I/O number X0 is saved in the device X0.
Likewise, the status of the device Y10 corresponds with the terminal assigned with the I/O number Y10.

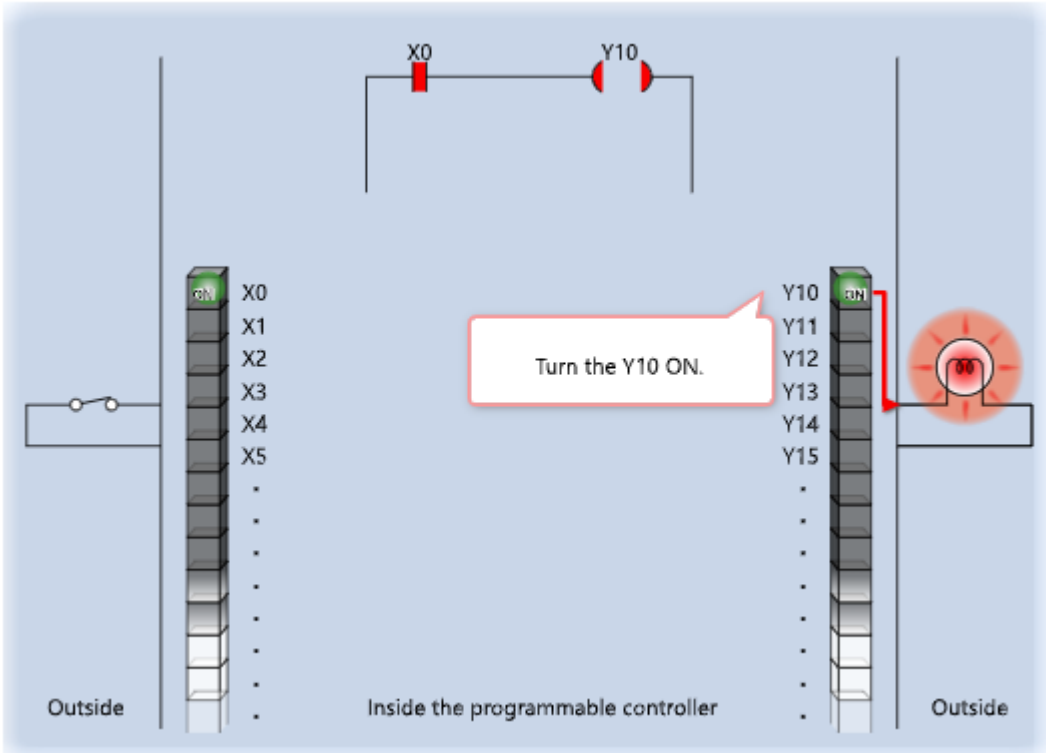


1.4.3 I/O numbers and devices

Statuses of equipment connected to the unit are saved in the programmable controller memory area called "devices". Likewise for the output, output equipment operates in accordance with the statuses of the devices. As explained above, control programs are often executed based on the device statuses.

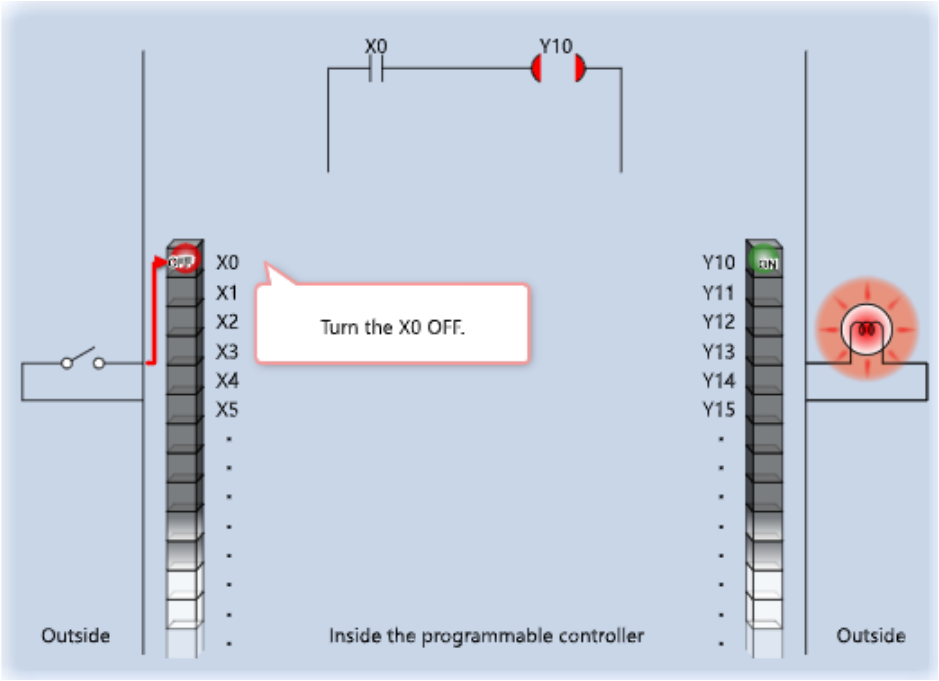
Devices that store bit information (ON/OFF) such as input (X) and output (Y) are called "bit devices".

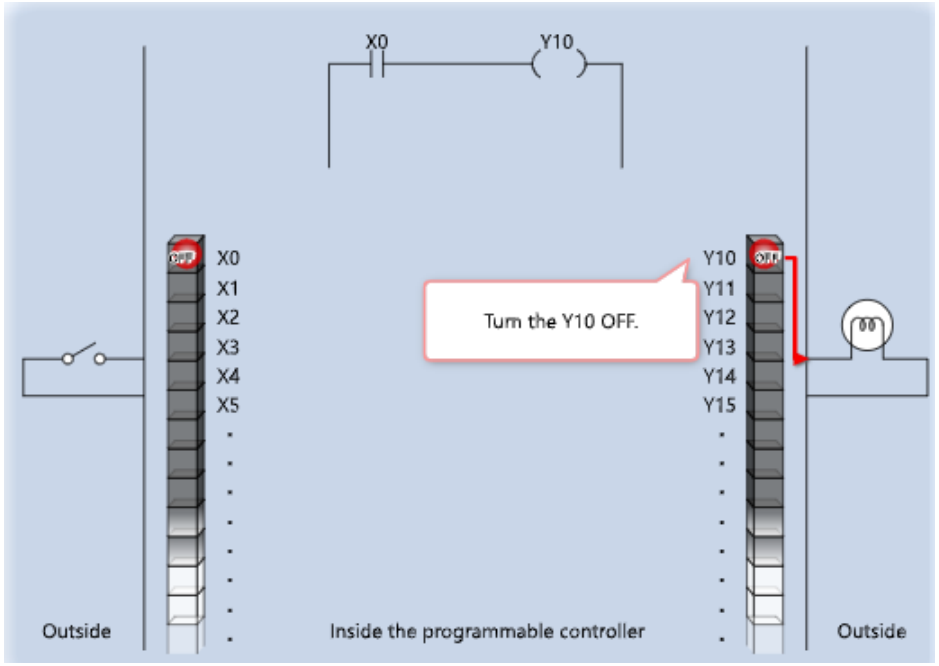
The device numbers correspond with I/O numbers.
For example, the status of the terminal assigned with I/O number X0 is saved in the device X0.
Likewise, the status of the device Y10 corresponds with the terminal assigned with the I/O number Y10.



Ch4: PLC Programming– PLC Programming Basic (Ladder)

PLC
Programmable Logic Controller





1.4.4 Internal relays

We have learned that bit devices, such as X (input) and Y (output), correspond with the numbers assigned to physical module I/O terminals.
There is another group of bit devices, which have no relation with module I/O terminals, and one of them is called "internal relays (M)".

Internal relays (M) are expressed in decimal format, although input (X) and output (Y) devices are expressed in hexadecimal for the MELSEC iQ-R/Q/L Series, and are in octal for the MELSEC iQ-F/F Series.

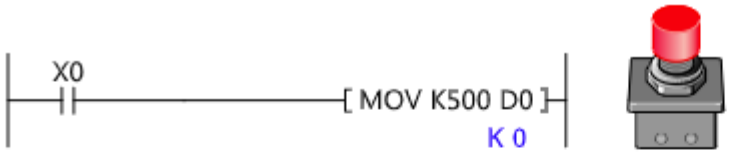
Internal relays (M) are mainly used to store temporary bit data.
For example, internal relays (M) can be used to store the computational result of an operation so that it can be used in another ladder rung.

1.4.5 Word devices

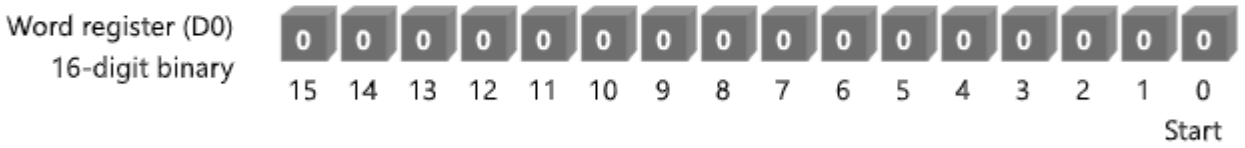
We have learned that devices that store bit information (ON/OFF) are called "bit devices", and the devices that store word devices are called "word devices".

"Data registers" (D) are one of the word devices typically used. One data register (D) can store 1 word (16 bits) of data.

The animation below shows how to use data registers (D).
In the program, turning ON X0 saves "K500" (decimal value) to D0.
A MOV instruction copies a value to a specified device. (Further details will be explained in Chapter 3.1.)
In this section, numbers are saved in data registers.



Decimal K0



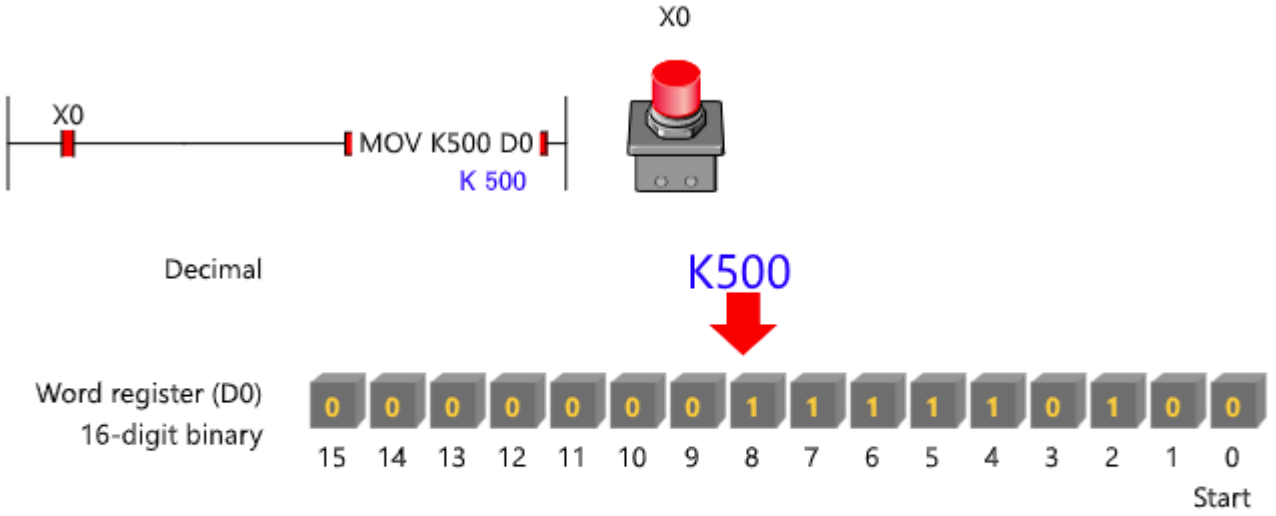
Pressing X0 copies the decimal value "K500" to the data register D0.
(The initial value of D0 is "K0".)

1.4.5 Word devices

We have learned that devices that store bit information (ON/OFF) are called "bit devices", and the devices that store word devices are called "word devices".

"Data registers" (D) are one of the word devices typically used. One data register (D) can store 1 word (16 bits) of data.

The animation below shows how to use data registers (D).
In the program, turning ON X0 saves "K500" (decimal value) to D0.
A MOV instruction copies a value to a specified device. (Further details will be explained in Chapter 3.1.)
In this section, numbers are saved in data registers.



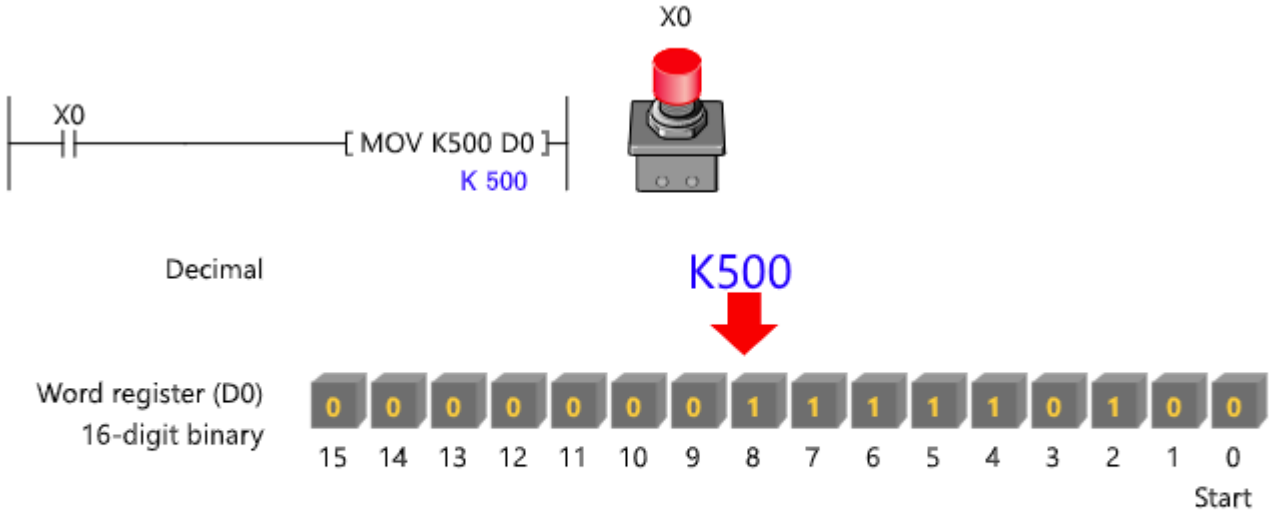
Pressing X0 copies the decimal value "K500" to the data register D0.
(The initial value of D0 is "K0".)

1.4.5 Word devices

We have learned that devices that store bit information (ON/OFF) are called "bit devices", and the devices that store word devices are called "word devices".

"Data registers" (D) are one of the word devices typically used. One data register (D) can store 1 word (16 bits) of data.

The animation below shows how to use data registers (D).
In the program, turning ON X0 saves "K500" (decimal value) to D0.
A MOV instruction copies a value to a specified device. (Further details will be explained in Chapter 3.1.)
In this section, numbers are saved in data registers.

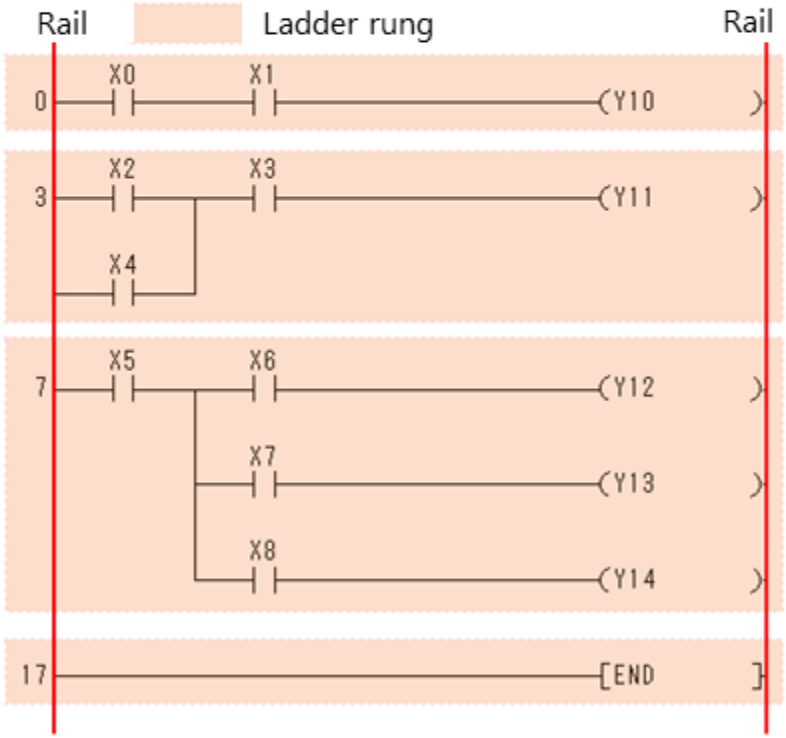


Even if the button is released and X0 turns OFF, "K500"saved in the data register D0 is held.

1.5 Creating control programs

Control programs consist of rails on the left and right edges and instruction symbols connected with lines.

An area from the -| |- instruction connected the left rail to the -( )- or -[ ]- instruction connected to the right rail is called a ladder rung.
Several of these ladder rungs constitute a control program, which ends with -[END]- or -[FEND]- instruction.



■ **Difference between engineering software**

-( )- and -[ ]- instructions are different between the engineering software, GX Works2 and GX Works3. In this course, the description uses the instructions of GX Works2.

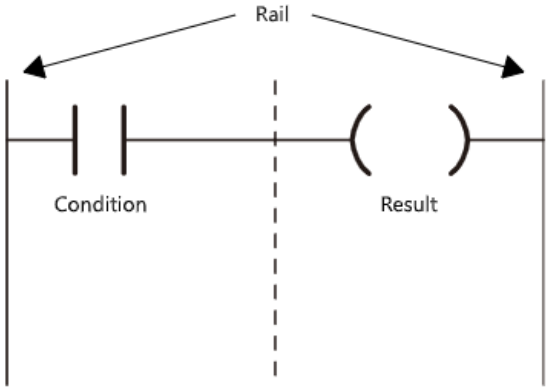
	GX Works2	GX Works3
-( )-	(Y10)	Y10
-[ ]-	{MOV K500 D0 }	MOV K500 D0

1.5.1 Instruction symbols

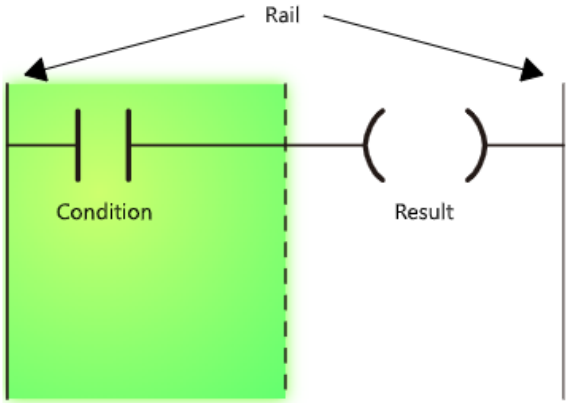
As explained in Chapter 1.3, a programmable controller must to be instructed to perform certain operations when a pre-specified input condition is satisfied. For such instructions, instruction symbols are used to express input conditions and output details.

An instruction symbol often contains a device number.
A device number specifies an area (device) storing a status, which is used for condition judgment, or as an output result.

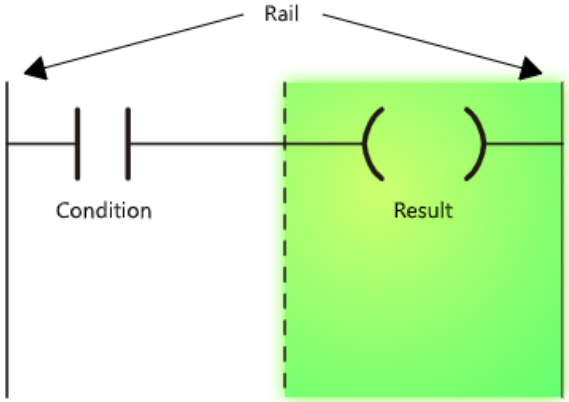
A ladder rung contains conditions and output results. Conditions are placed to the left, and output results are placed to the right.
Output results could be a simple ON/OFF signal or a dedicated instruction such as a calculation or copying operation.



In a ladder diagram, there are two parallel rails.



In a ladder diagram, there are two parallel rails.
On the left, conditions are written.



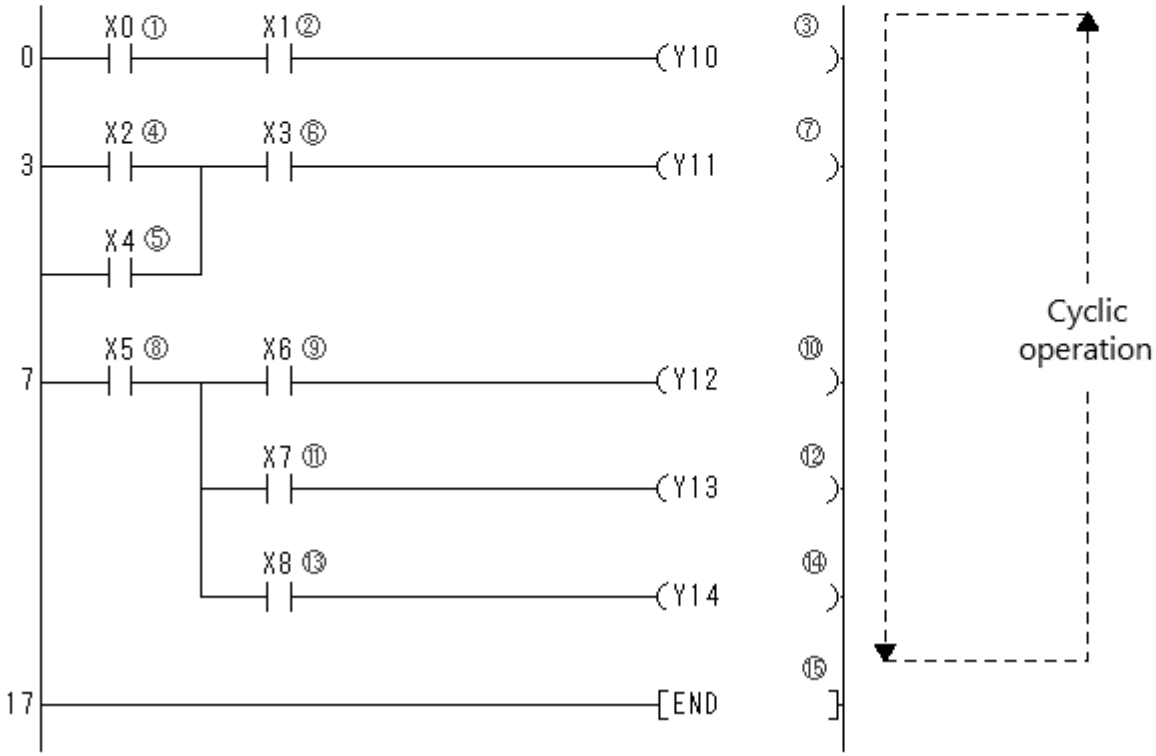
In a ladder diagram, there are two parallel rails.
On the left, conditions are written.
On the right, results are written.

1.6 Program execution procedure

A program starts from the left start instruction and ends with the -[END]- instruction.
After reaching to the -[END]- instruction, the program execution starts again from the start instruction.
This repetitive execution is called "cyclic operation".

One cycle of this cyclic operation is called a "scan", and the time period that takes to process one scan is called the "scan time".

The figure below shows instruction execution procedure.
Instructions are executed from left to right on each ladder rung, and then from the top to the bottom ladder rung (No. 1, 2,...15 -> 1...).

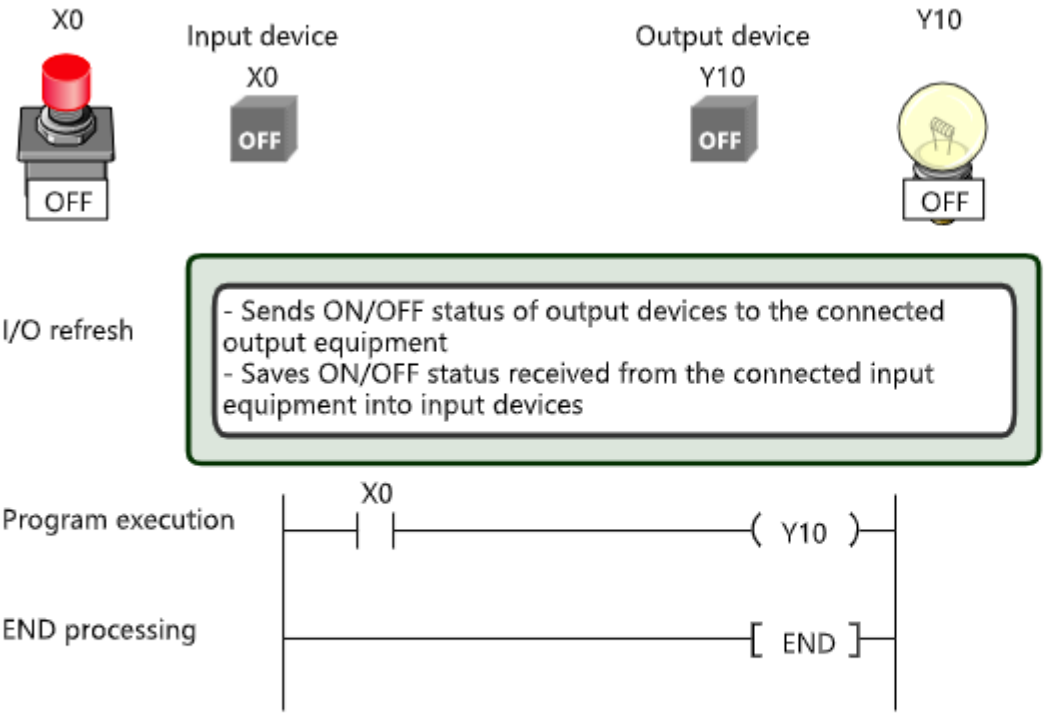


1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



The programmable controller CPU repeats I/O refresh -> program execution -> END processing.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).

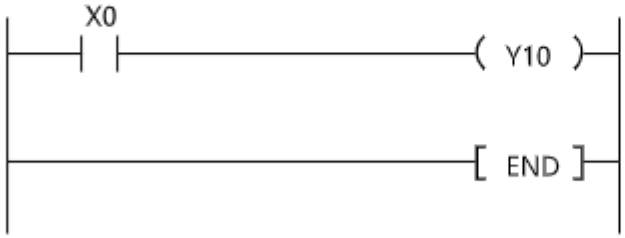


(1) I/O refresh

- Sends ON/OFF status of output devices to the connected output equipment
- Saves ON/OFF status received from the connected input equipment into input devices

(2) Program execution

(3) END processing



I/O refresh
- OFF status of the output device Y10 turns off the lamp (output equipment)
- OFF status of the push-button switch (input equipment) saves the OFF status into the input device X0.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

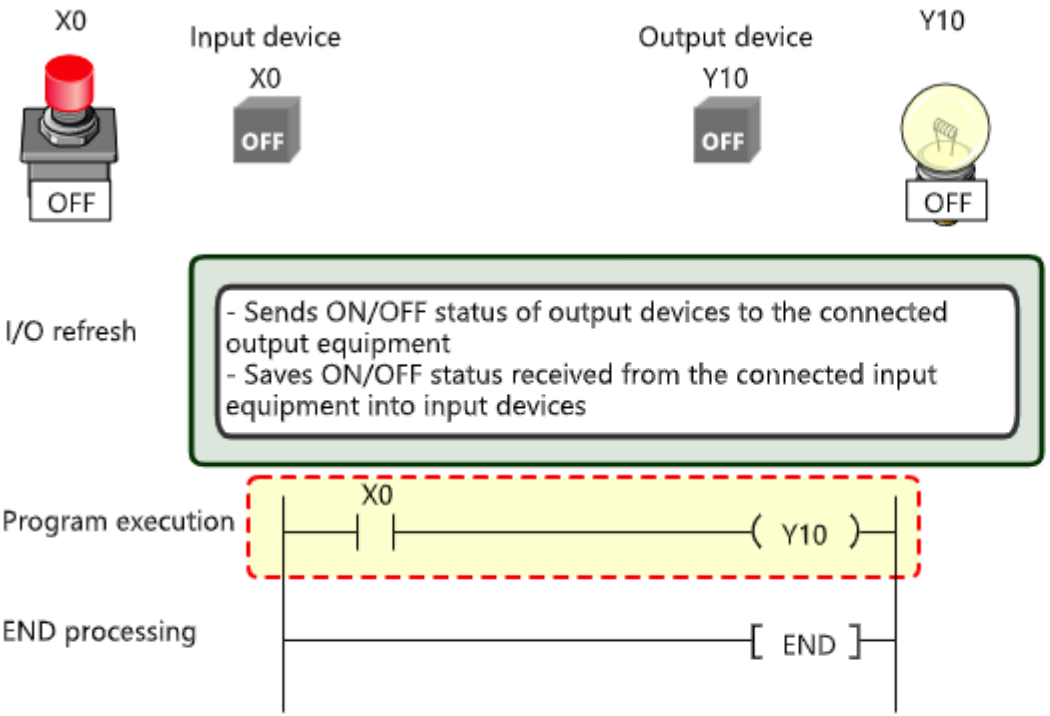
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



Program execution
OFF status of the input device X0 does not satisfy the input condition, so the output device Y10 is OFF.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

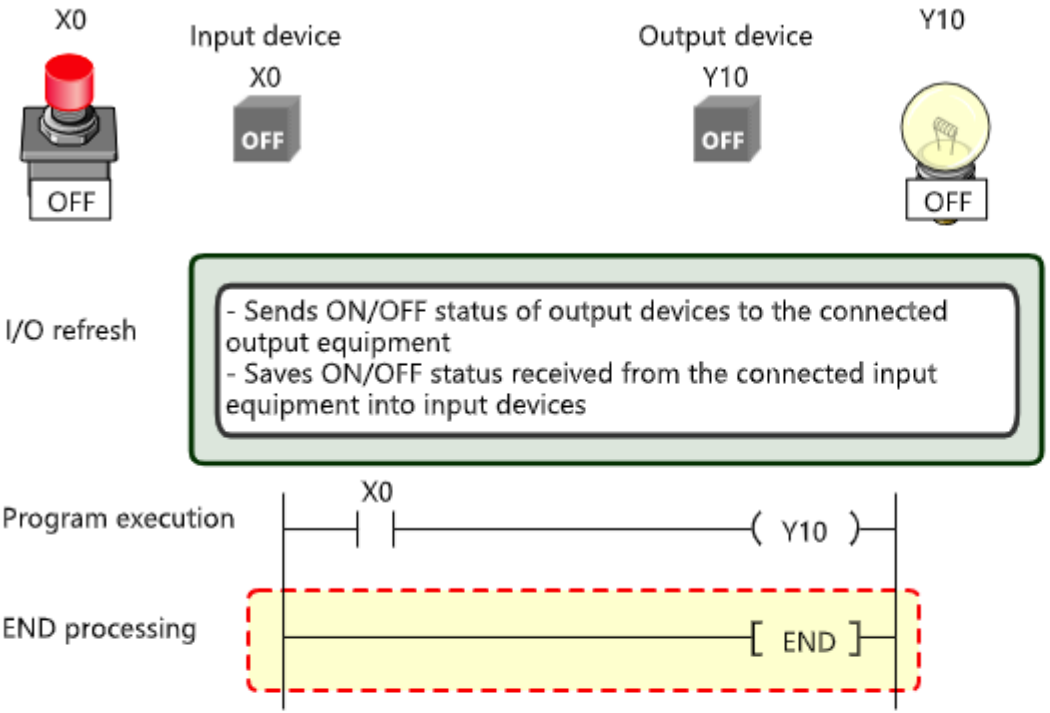
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

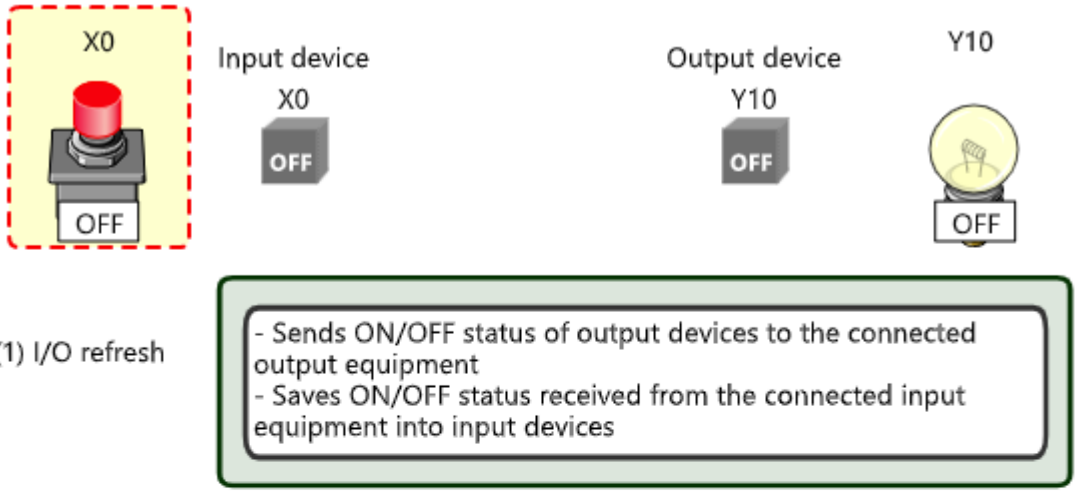
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

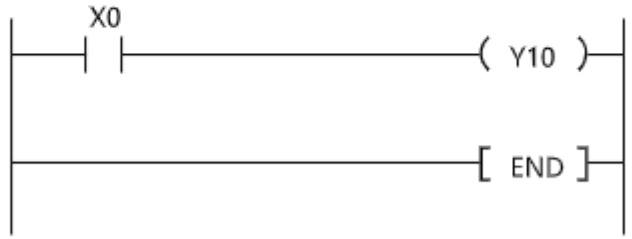
Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



(1) I/O refresh

- Sends ON/OFF status of output devices to the connected output equipment
- Saves ON/OFF status received from the connected input equipment into input devices



(2) Program execution

(3) END processing

Pressing the button on the input equipment sends an ON signal to the programmable controller input module.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).

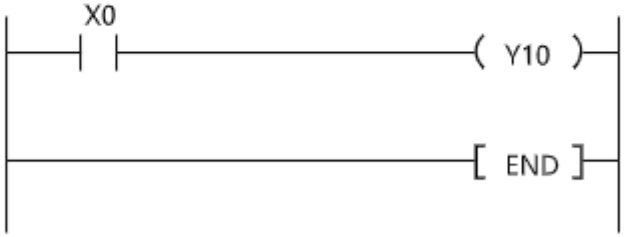


(1) I/O refresh

- Sends ON/OFF status of output devices to the connected output equipment
- Saves ON/OFF status received from the connected input equipment into input devices

(2) Program execution

(3) END processing



I/O refresh
- OFF status of the output device Y10 turns off the lamp (output equipment)
- ON status of the push-button switch (input equipment) saves the ON status into the input device X0.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

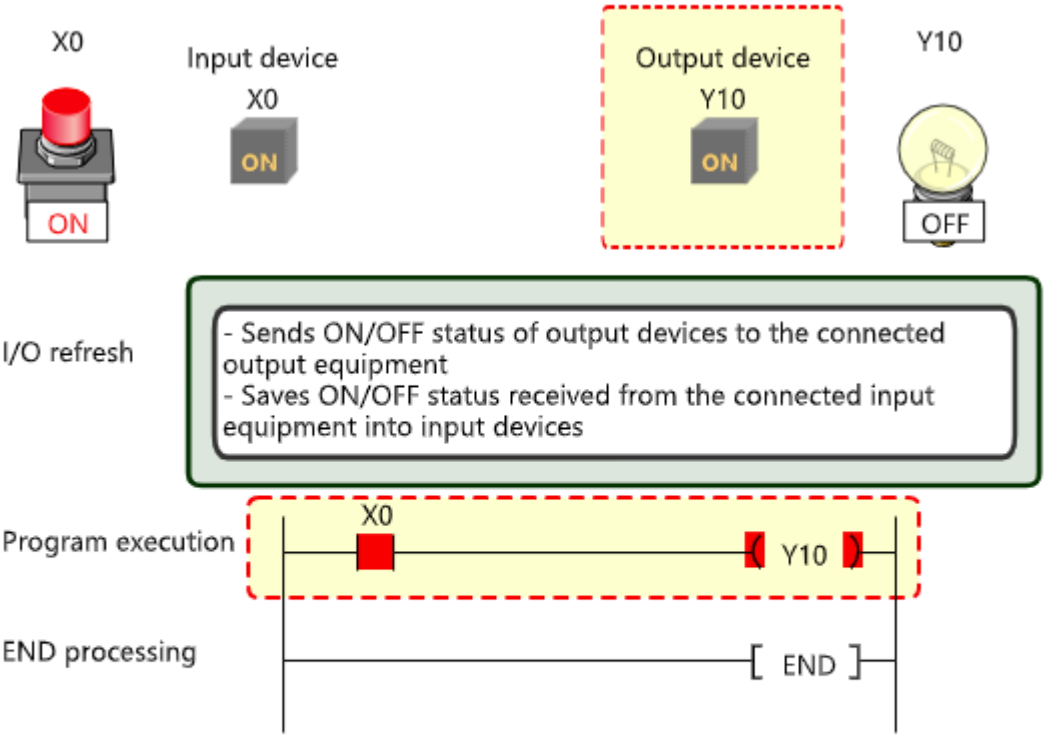
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



Program execution
ON status of the input device X0 satisfies the input condition and turns ON the output device Y10.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

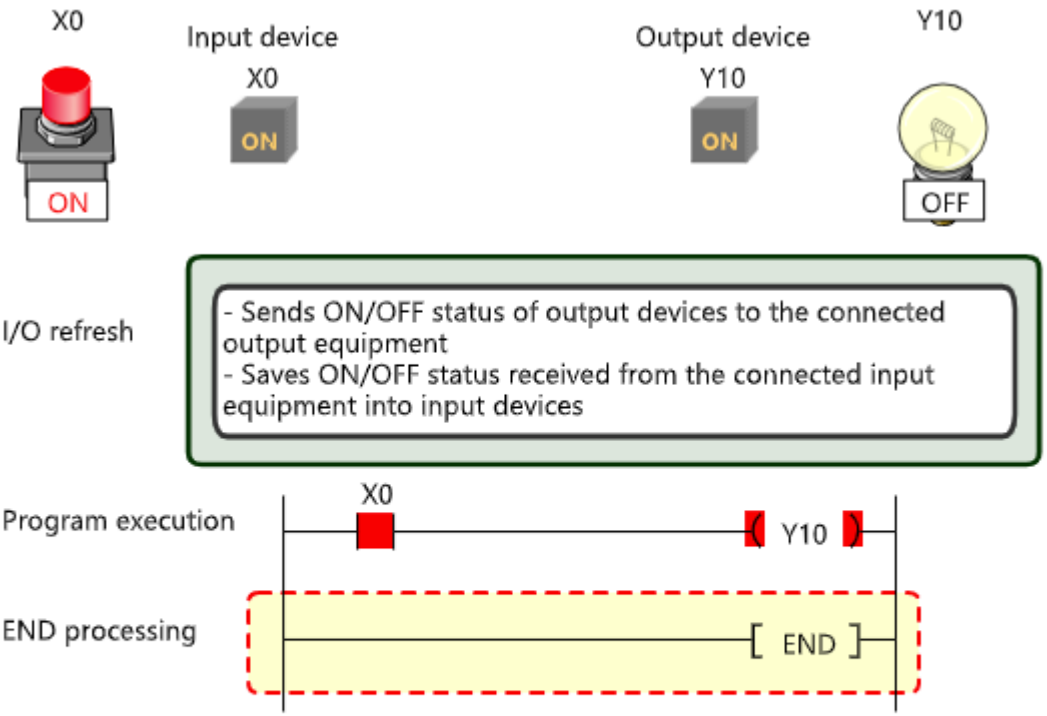
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



END processing by an END instruction
(Details are omitted here.)

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).

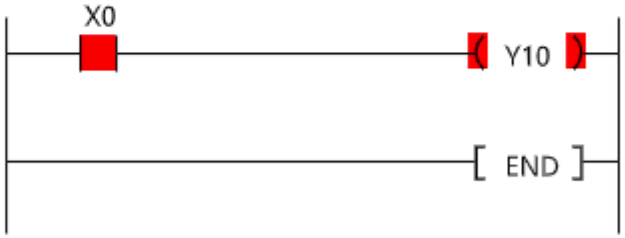


(1) I/O refresh

- Sends ON/OFF status of output devices to the connected output equipment
- Saves ON/OFF status received from the connected input equipment into input devices

(2) Program execution

(3) END processing



I/O refresh
- ON status of the output device Y10 turns ON the lamp (output equipment)
- ON status of the push-button switch (input equipment) saves the ON status into the input device X0.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

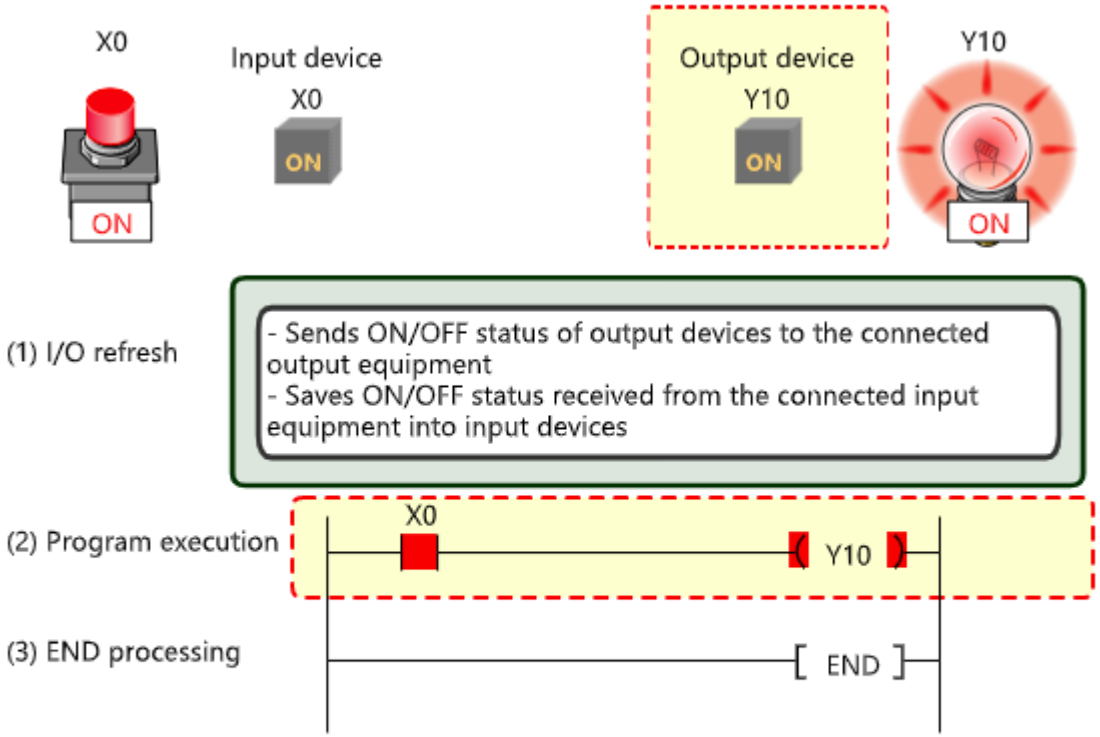
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



Program execution
ON status of the input device X0 satisfies the input condition and turns ON the output device Y10.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

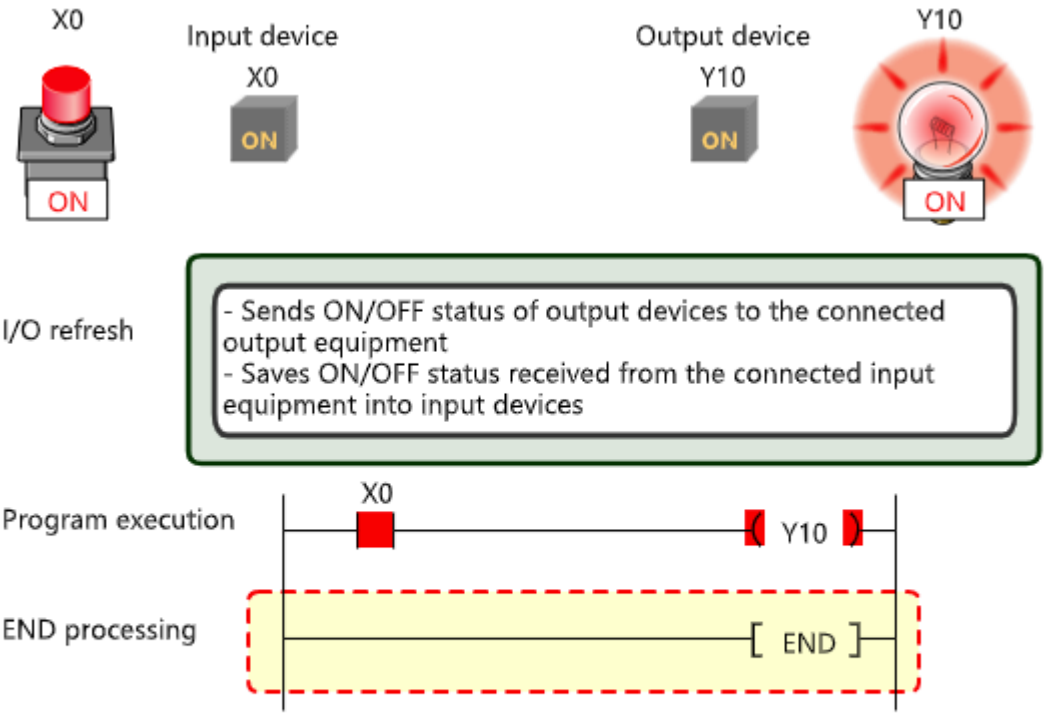
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



END processing by an END instruction
(Details are omitted here.)

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

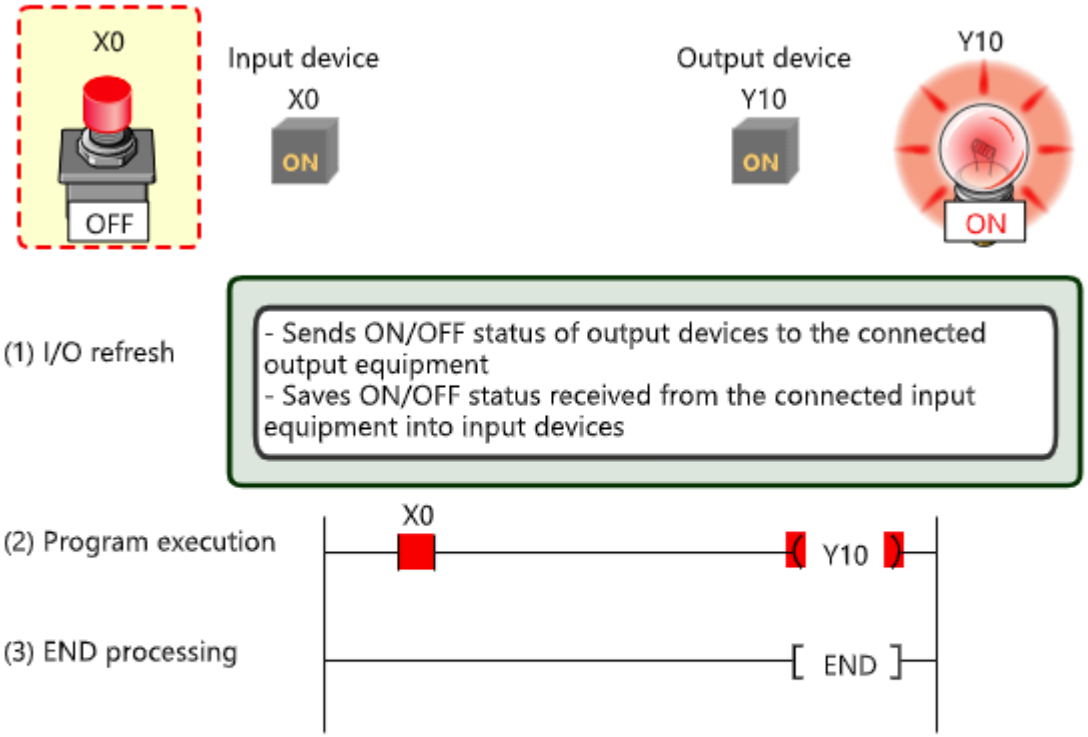
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



Releasing the button on the input equipment sends an OFF signal to the programmable controller input module.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).

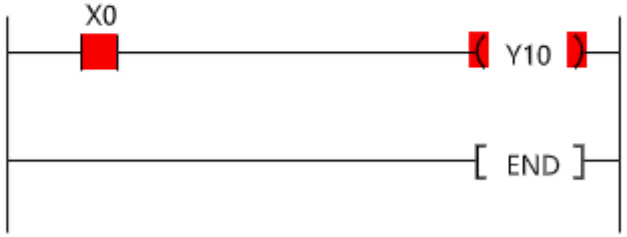


(1) I/O refresh

- Sends ON/OFF status of output devices to the connected output equipment
- Saves ON/OFF status received from the connected input equipment into input devices

(2) Program execution

(3) END processing



I/O refresh
- ON status of the output device Y10 turns ON the lamp (output equipment)
- OFF status of the push-button switch (input status) saves the OFF status into the input device X0.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

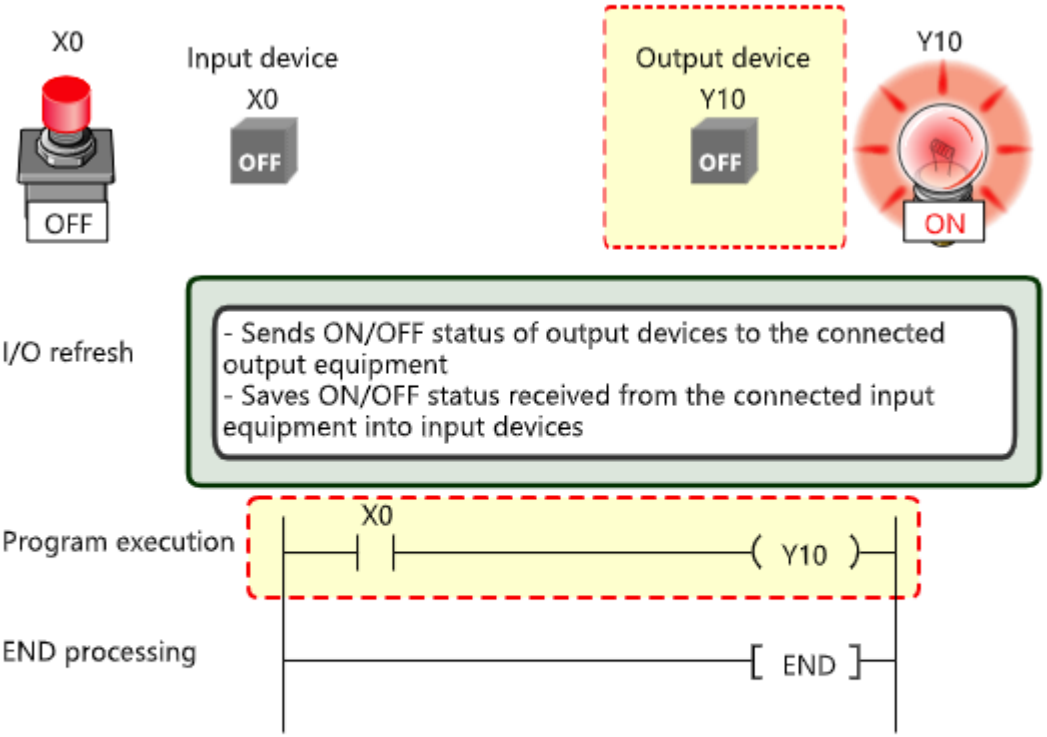
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



Program execution
OFF status of the input device X0 does not satisfy the input condition, so the output device Y10 is OFF.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

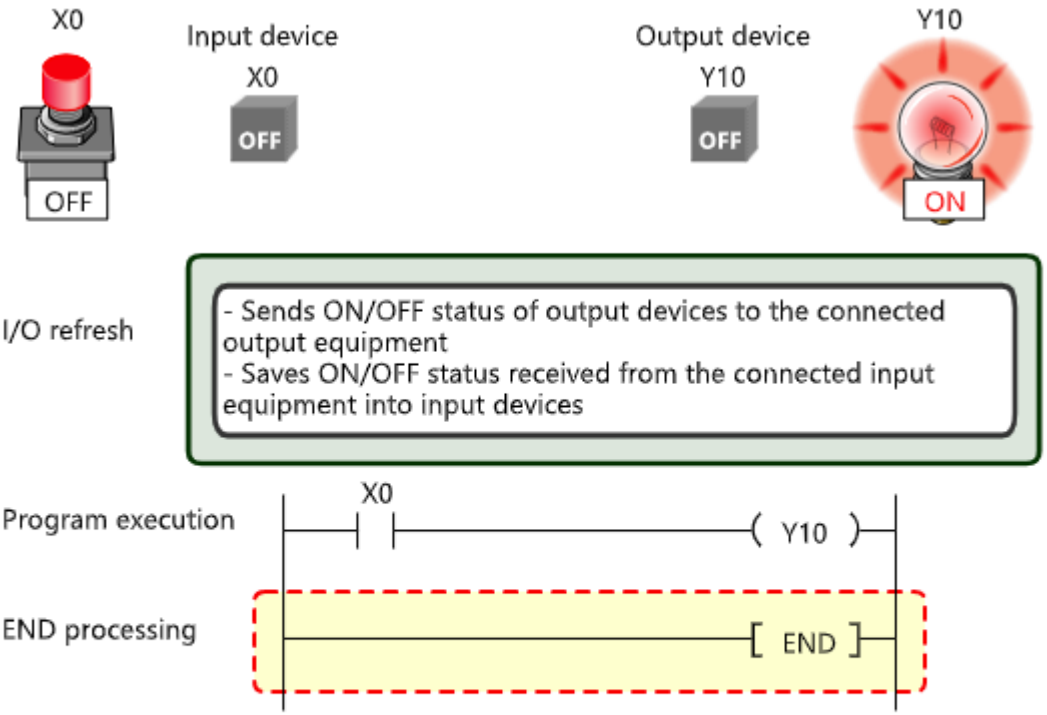
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).

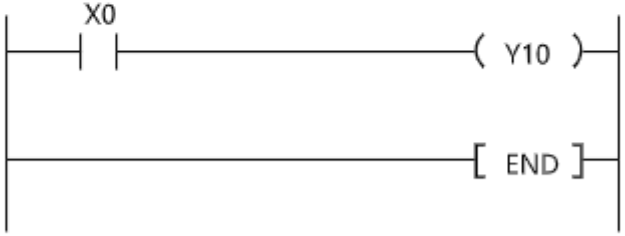


(1) I/O refresh

- Sends ON/OFF status of output devices to the connected output equipment
- Saves ON/OFF status received from the connected input equipment into input devices

(2) Program execution

(3) END processing



I/O refresh
- OFF status of the output device Y10 turns off the lamp (output equipment)
- OFF status of the push-button switch (input status) saves the OFF status into the input device X0.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

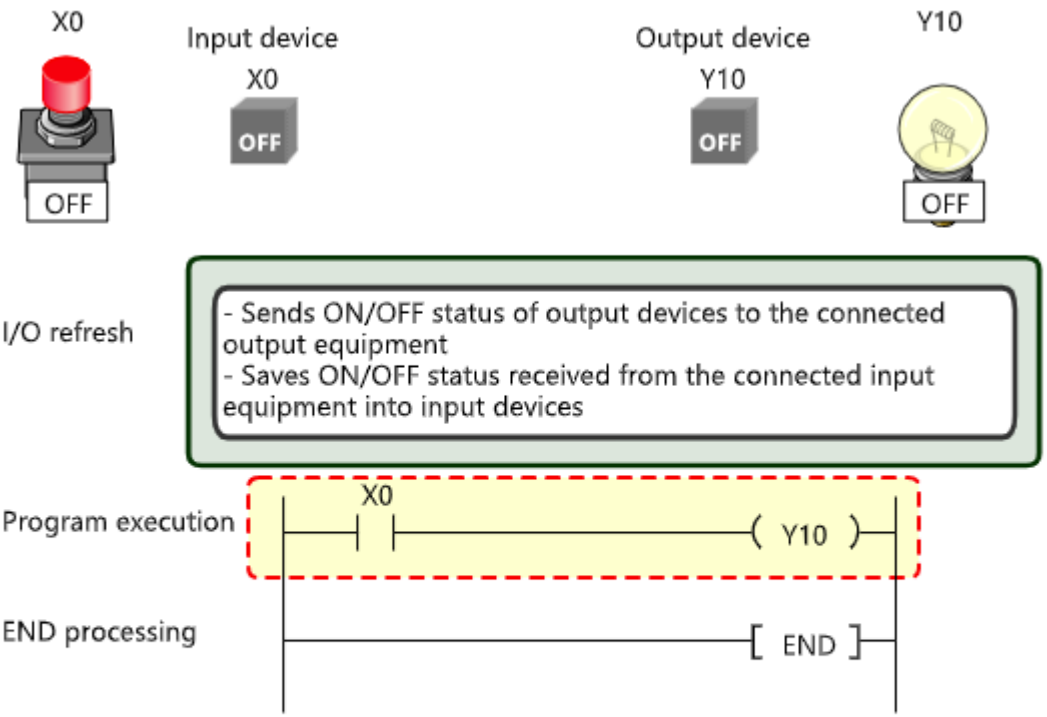
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



Program execution
OFF status of the input device X0 does not satisfy the input condition, so the output device Y10 is OFF.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

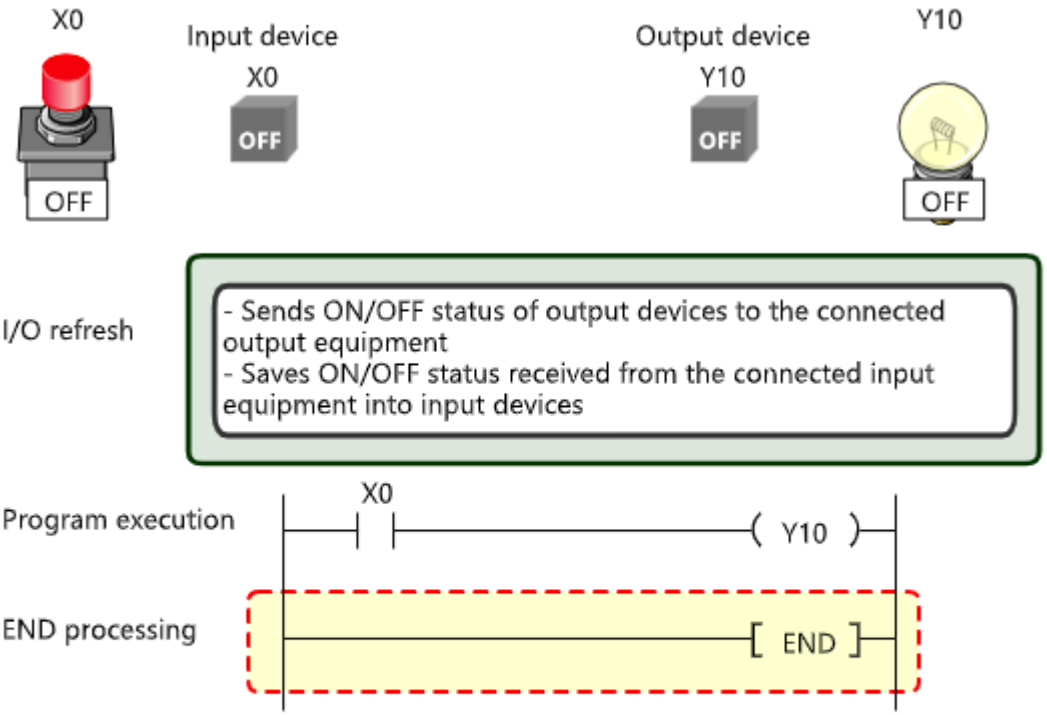
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

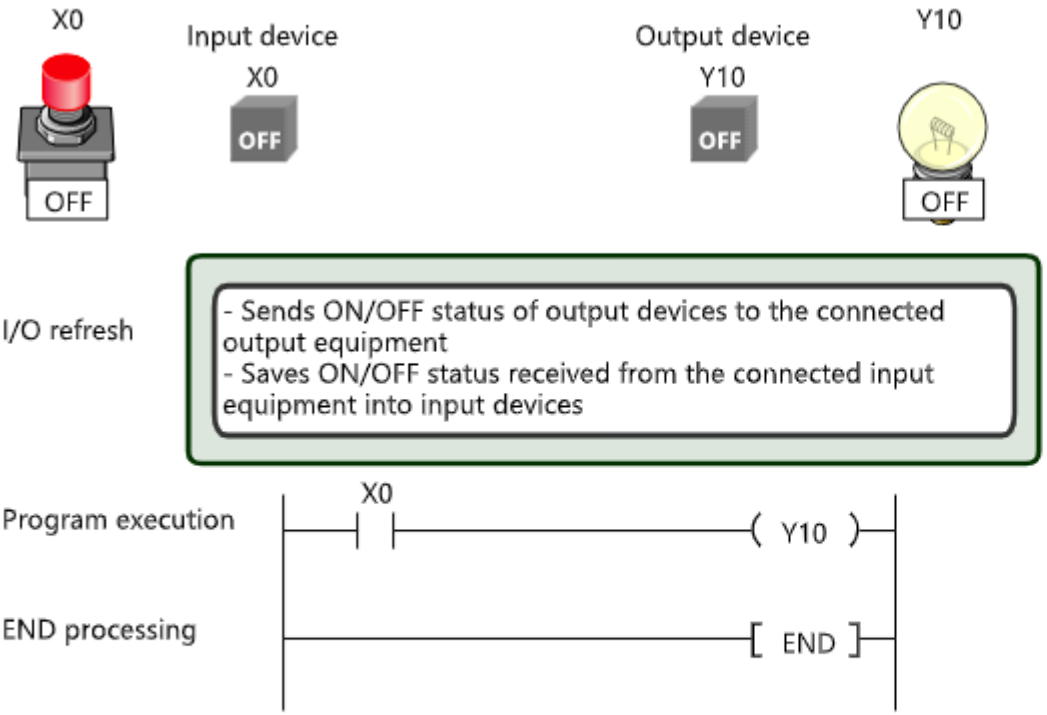
This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

1.7 Refresh time

As explained earlier, scan time is the time period that takes to execute a series of programs for once.
Scan time can also be expressed as:

Scan time = refresh time + program execution time + END processing time

Refresh time is the time period that takes to read data from the input module into the input devices (X) plus the time period that takes to write data to the output module from the output devices (Y).



The programmable controller CPU repeats I/O refresh -> program execution -> END processing.

Note that ON/OFF statuses of an actual switch are read at once and saved to the input devices (X), and the new data overwrite the existing values during refresh.
Likewise, data in the output devices (Y) are written at once to the output module at an instruction execution.

This means that if a signal turns from OFF to ON, then to OFF again, the signal is never recognized as ON. However, the scan time is very short compared to signal length. It is rare for the programmable controller to miss a signal status change.

Chapter 2 Bit device instructions

This chapter explains about instructions using bit devices (ON/OFF).

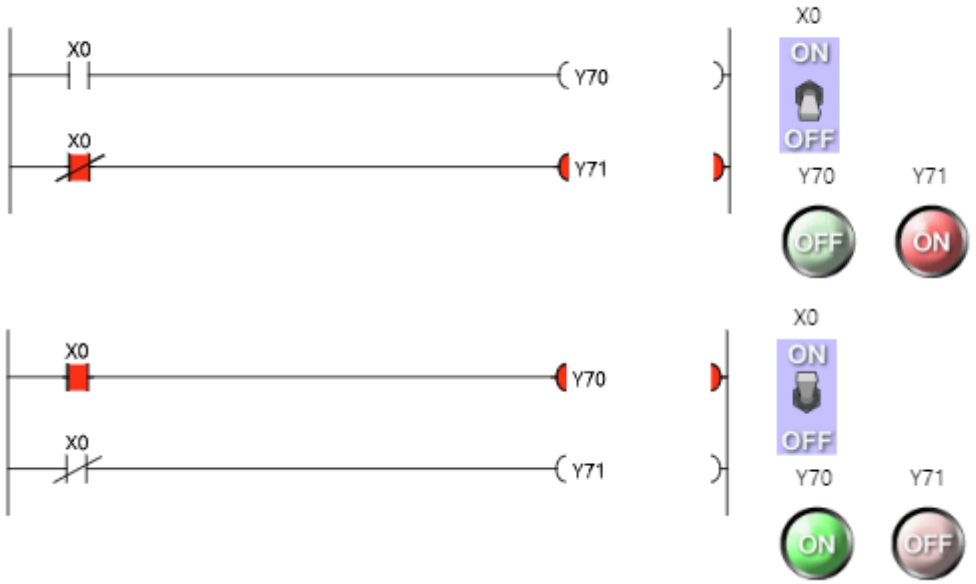
Operations that use bit devices are the most basic operations in control programs.
Inputs from input equipment are used as conditions to control output equipment.

2.1 Input conditions and outputs

Normally open (NO) and normally closed (NC) contacts are used as input conditions.
When input conditions are satisfied, a coil output instruction (OUT instruction) is output.
When the input conditions are not satisfied, the coil output instruction is not output.
NO/NC contact instruction and OUT instruction are the main instruction combination used in control programs.

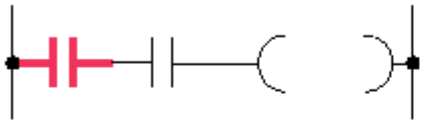
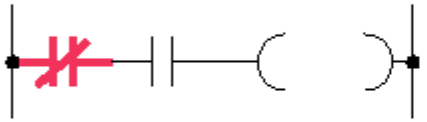
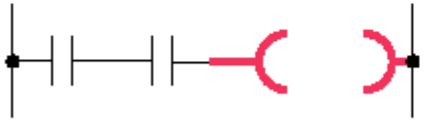
■ **Ladder program and operation**

Simulate the operation of the NO, NC, and OUT instructions by clicking the input switch shown on the right.



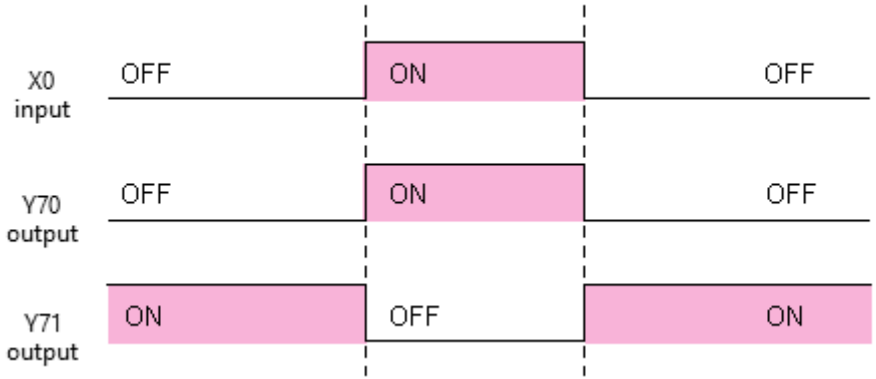
Ch4: PLC Programming– PLC Programming Basic (Ladder)

■ Instruction codes and functions

Symbol	Function
	NO contact Conducted when the device status is ON.
	NC contact Conducted when the device status is OFF (opposite of the NO contact).
	Coil output (OUT) When the preceding input condition is satisfied, the data in the preset device is output.
	End instruction (END) Indicates the end of a program. A program requires an END instruction.



■ Timing chart



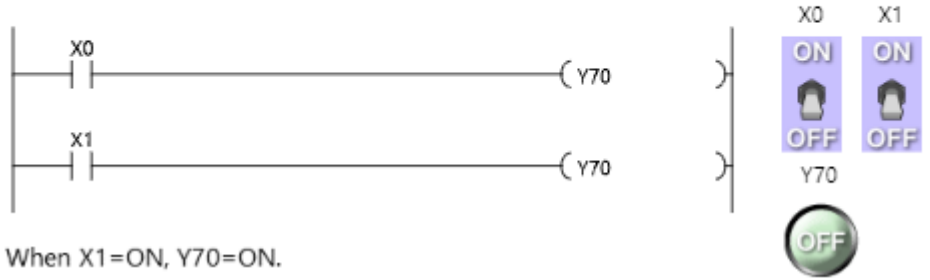
2.1.1 Using the same device number for instructions

On a ladder rung, only one OUT instruction can be used with one device number. If more than one OUT instruction are used with the same device number, only the latter OUT instruction becomes valid, making the first OUT instruction invalid.

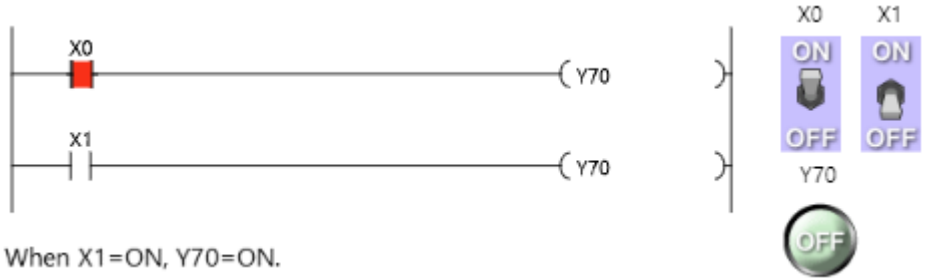
■ Ladder program

Simulate the operation of two instructions having the same device number by clicking the input switch shown on the right.
This type of usage (using OUT Y70 for two instructions) is called "duplicate coil".

■ Ladder program
Simulate the operation of two instructions having the same device number by clicking the input switch shown on the right.
This type of usage (using OUT Y70 for two instructions) is called "duplicate coil".



When X1=ON, Y70=ON.



When X1=ON, Y70=ON.

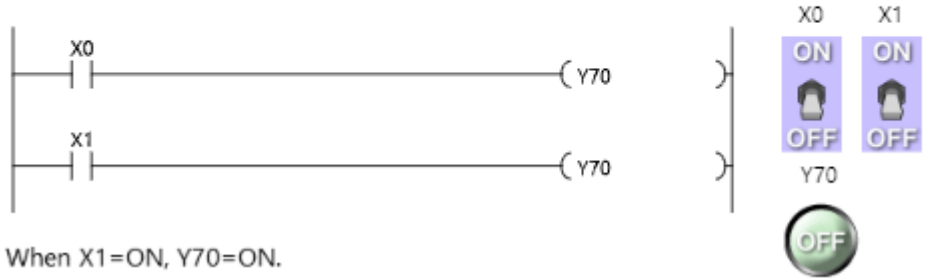
2.1.1 Using the same device number for instructions

On a ladder rung, only one OUT instruction can be used with one device number. If more than one OUT instruction are used with the same device number, only the latter OUT instruction becomes valid, making the first OUT instruction invalid.

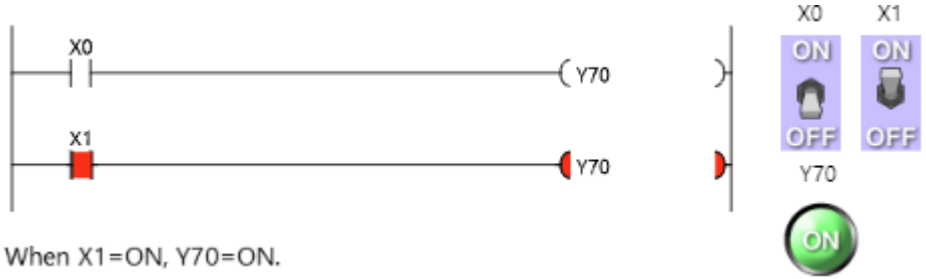
■ Ladder program

Simulate the operation of two instructions having the same device number by clicking the input switch shown on the right.
This type of usage (using OUT Y70 for two instructions) is called "duplicate coil".

■ Ladder program
Simulate the operation of two instructions having the same device number by clicking the input switch shown on the right.
This type of usage (using OUT Y70 for two instructions) is called "duplicate coil".



When X1=ON, Y70=ON.



When X1=ON, Y70=ON.

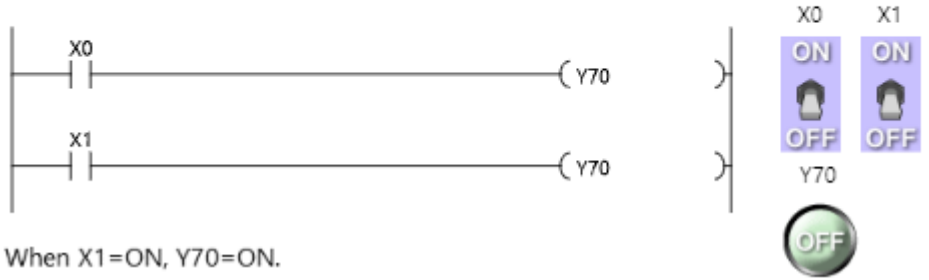
2.1.1 Using the same device number for instructions

On a ladder rung, only one OUT instruction can be used with one device number. If more than one OUT instruction are used with the same device number, only the latter OUT instruction becomes valid, making the first OUT instruction invalid.

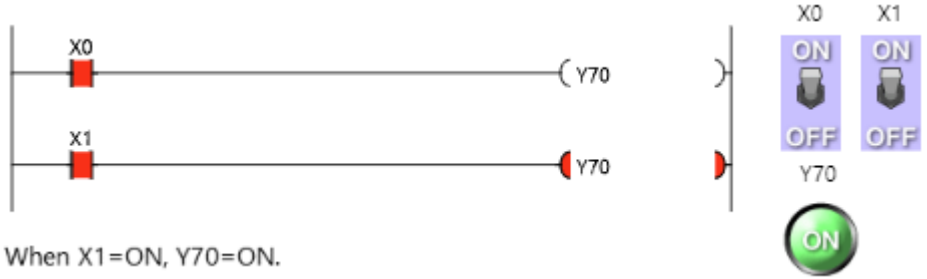
■ Ladder program

Simulate the operation of two instructions having the same device number by clicking the input switch shown on the right.
This type of usage (using OUT Y70 for two instructions) is called "duplicate coil".

■ Ladder program
Simulate the operation of two instructions having the same device number by clicking the input switch shown on the right.
This type of usage (using OUT Y70 for two instructions) is called "duplicate coil".



When X1=ON, Y70=ON.



When X1=ON, Y70=ON.

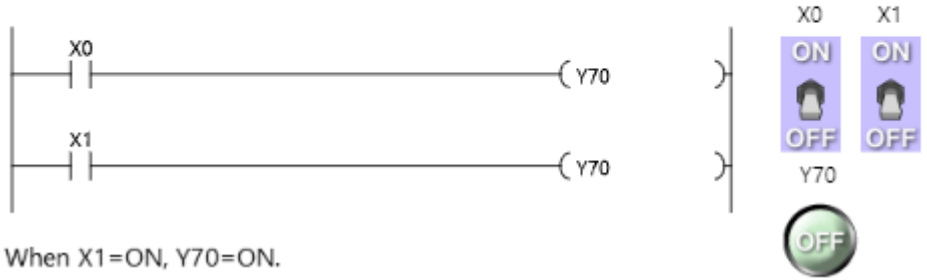
2.1.1 Using the same device number for instructions

On a ladder rung, only one OUT instruction can be used with one device number. If more than one OUT instruction are used with the same device number, only the latter OUT instruction becomes valid, making the first OUT instruction invalid.

■ Ladder program

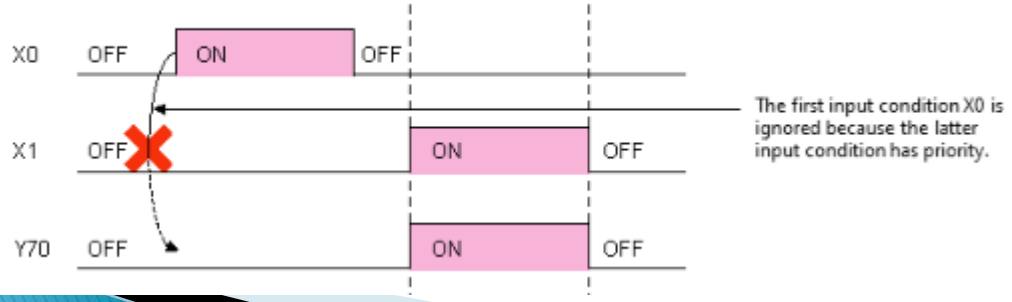
Simulate the operation of two instructions having the same device number by clicking the input switch shown on the right.
This type of usage (using OUT Y70 for two instructions) is called "duplicate coil".

■ Ladder program
Simulate the operation of two instructions having the same device number by clicking the input switch shown on the right.
This type of usage (using OUT Y70 for two instructions) is called "duplicate coil".



When X1=ON, Y70=ON.

■ Timing chart

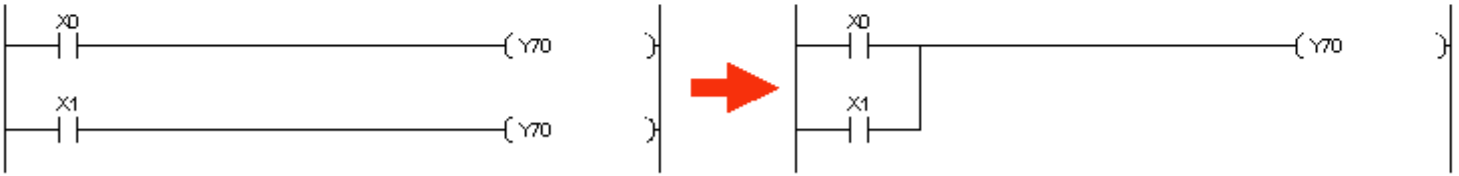


2.1.1 Using the same device number for instructions

■ Correction example

In this example, the input condition "X1" has a higher priority, and "X0" is ignored.

By correcting the ladder rung to the one in Fig. B, device Y70 is turned ON when either one of the two input conditions is satisfied, avoiding a conflict between the two OUT instructions.



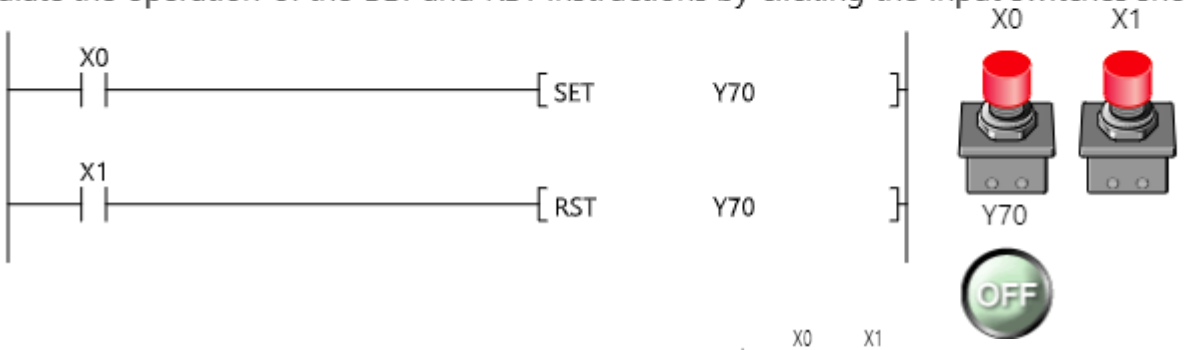
(Fig. A) (Fig. B)

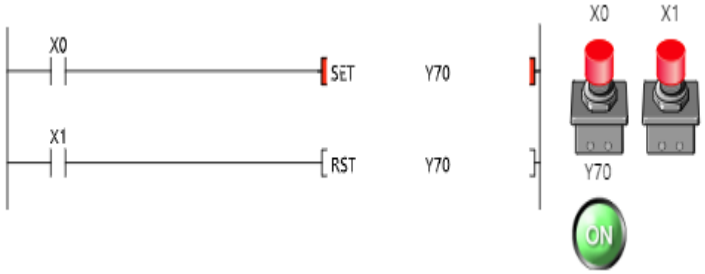
2.2 Retaining/cancelling outputs

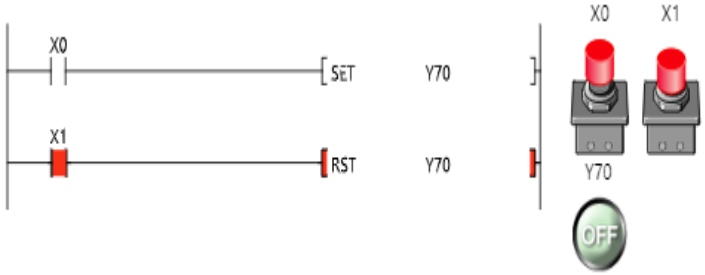
Unlike the OUT instruction, the operation retention instruction (SET instruction) holds an output status even if the input condition becomes unsatisfied.
To cancel the output (OFF), an operation retention cancel instruction (RST instruction) can be executed.

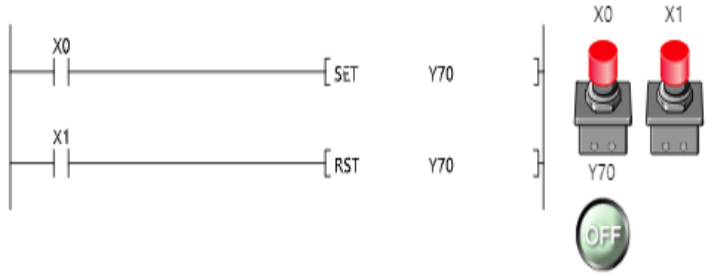
■ Ladder program and operation

Simulate the operation of the SET and RST instructions by clicking the input switches shown on the right.







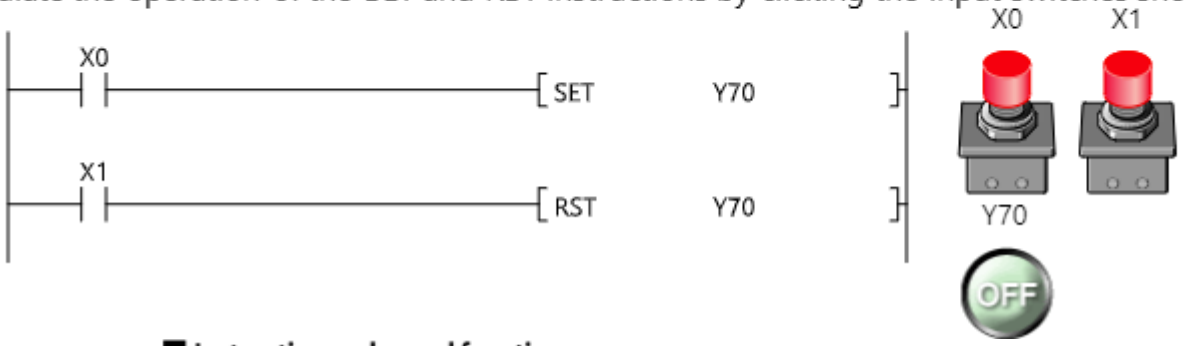


2.2 Retaining/cancelling outputs

Unlike the OUT instruction, the operation retention instruction (SET instruction) holds an output status even if the input condition becomes unsatisfied.
To cancel the output (OFF), an operation retention cancel instruction (RST instruction) can be executed.

■ Ladder program and operation

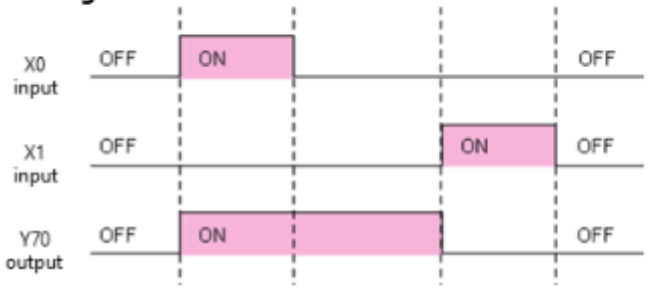
Simulate the operation of the SET and RST instructions by clicking the input switches shown on the right.



■ Instruction codes and functions

Symbol	Function
SET	Operation retention instruction (SET) This turns ON a device, and holds the ON (output) status. The output is held even if the input condition becomes unsatisfied.
RST	Operation retention cancel instruction (RSET) Cancels the ON status, and cancel the output to the specified device.

■ Timing chart



2.2.1 Differences between OUT and SET instructions

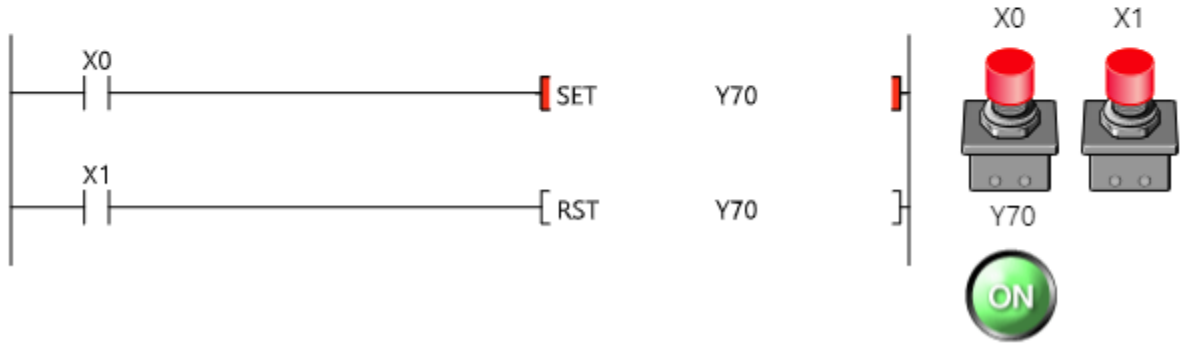
Simulate the operational differences between the OUT and SET instructions by clicking the input switches shown on the right.

■ OUT instruction



Y70 is ON when the input condition is satisfied.

■ SET/RST instructions

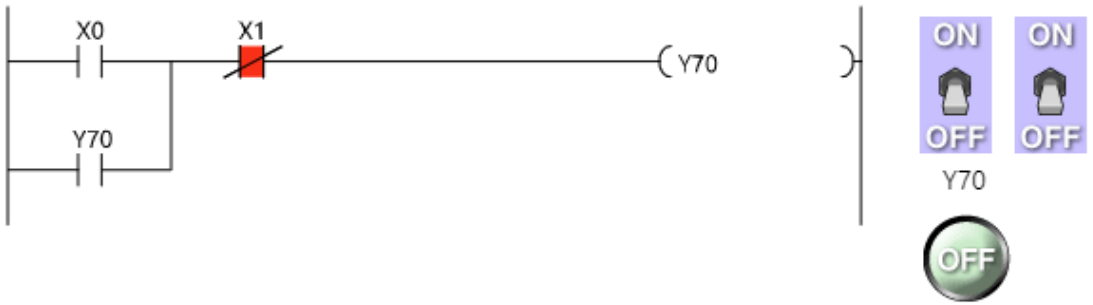


Once the input condition is satisfied, Y70 is ON until the RST instruction is executed.

2.2.2 Replacing retention ladders with SET instruction

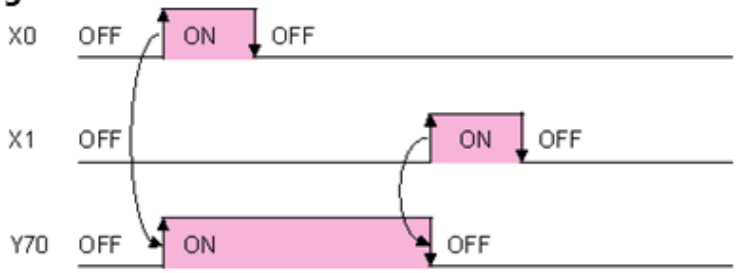
Ladder program and operation

Simulate the operation of the retention ladder by clicking the input switches shown on the right.



When X0=ON and X1=OFF, Y70=ON.
Y70 = ON (retention) until X1=ON.

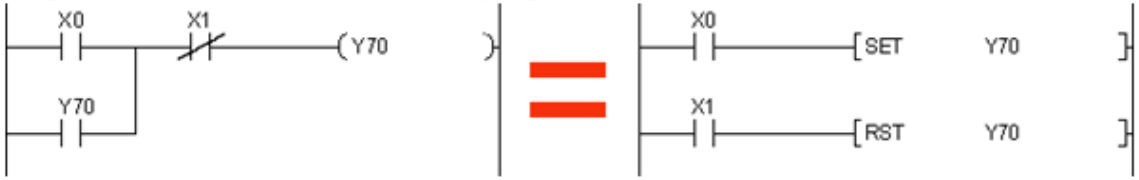
■ Timing chart



Even after X10 turns OFF, Y70 (coil) stays ON (retention)

■ Replacing with SET instruction

A retention ladder program can be rewritten as a ladder program with SET instruction.
By using the SET instruction, the ladder program can be simplified.

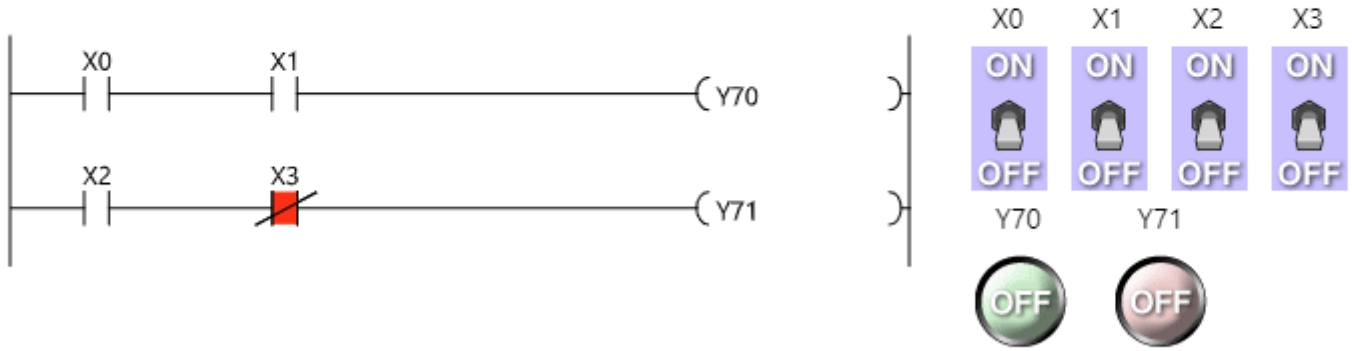


2.3 Adding conditions (AND logic)

To have an AND logic, NO/NC contacts are placed in series
In an AND logic, the condition is satisfied when more than NO/NC contacts, which are connected in series, are ON.

■ Ladder program and operation

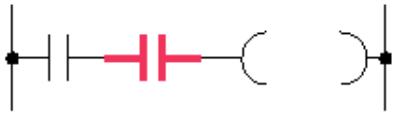
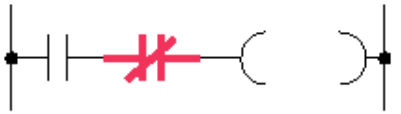
Simulate the operation of the AND logic by clicking the input switches shown on the right.



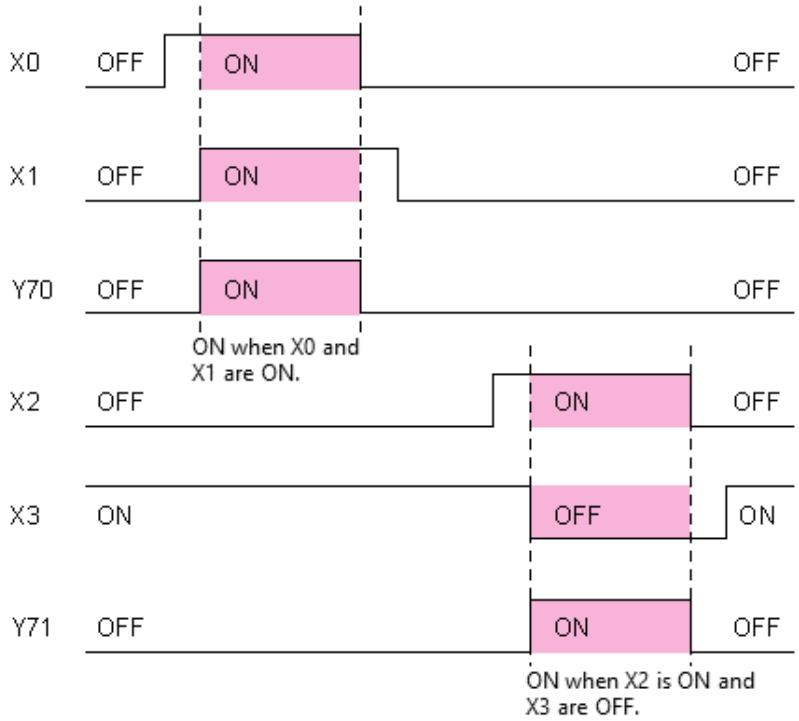
When X0 and X1 are ON, Y70 is ON.
When X2 is ON and X3 is OFF, Y71 is ON.

2.3 Adding conditions (AND logic)

■ Instruction codes and functions

Symbol	Function
	Serial connection of NO contact NO contact is connected in series (horizontally).
	Serial connection of NC contact NC contact is connected in series (horizontally).

■ Timing chart



2.4 Adding conditions (OR logic)

To have an OR logic, NO/NC contacts are placed in parallel.
In an OR logic, the condition is satisfied when one of NO/NC contacts, which are connected in parallel, is ON.

■ Ladder program and operation

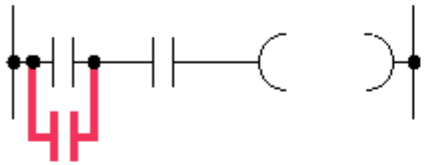
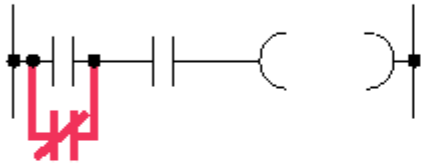
Simulate the operation of the OR logic by clicking the input switches shown on the right.



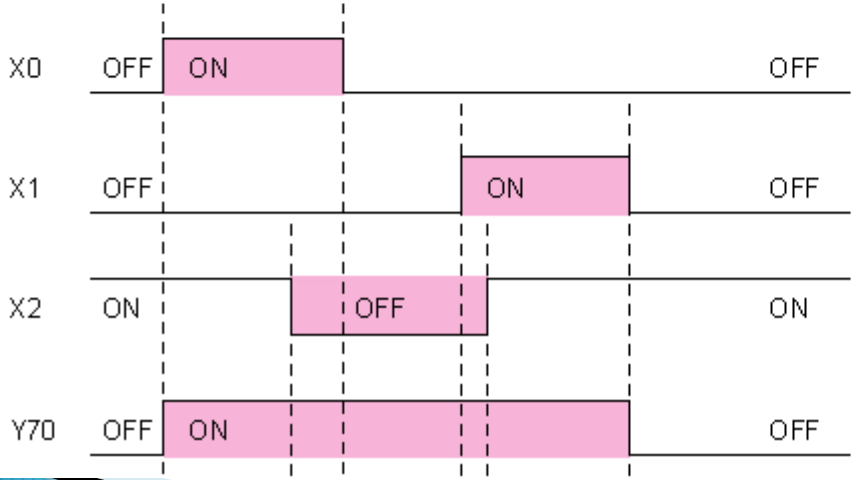
Y70 is ON when any of the following conditions are satisfied: X0 is ON, X1 is ON, or X2 is OFF.

2.4 Adding conditions (OR logic)

■ Instruction codes and functions

Symbol	Function
	Parallel connection of NO contacts NO contact is connected in parallel (vertically).
	Parallel connection of NC contact NC contact is connected in parallel (vertically).

■ Timing chart



2.5 Output as pulses

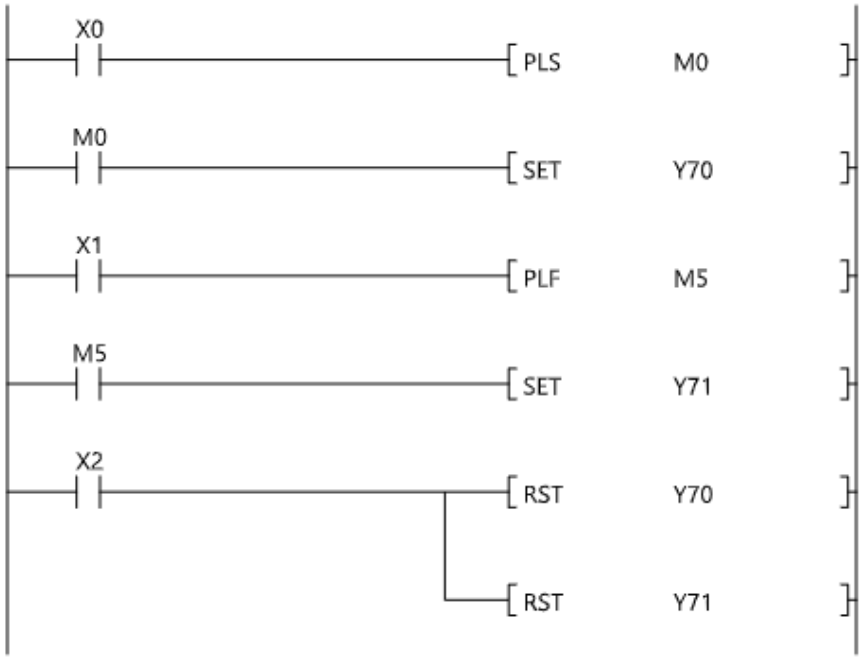
Unlike the OUT instruction, a rising edge instruction (PLS instruction) turns the coil ON for one scan after the input condition is satisfied.

Contrary to the PLS instruction, a falling edge instruction (PLF instruction) turns the coil ON for one scan after the input condition becomes unsatisfied.

The coil turned ON by PLS/PLF instruction returns to OFF after one scan.

■ Ladder program and operation

Simulate the operation of the PLS and PLF instructions by clicking the input switches shown on the right.





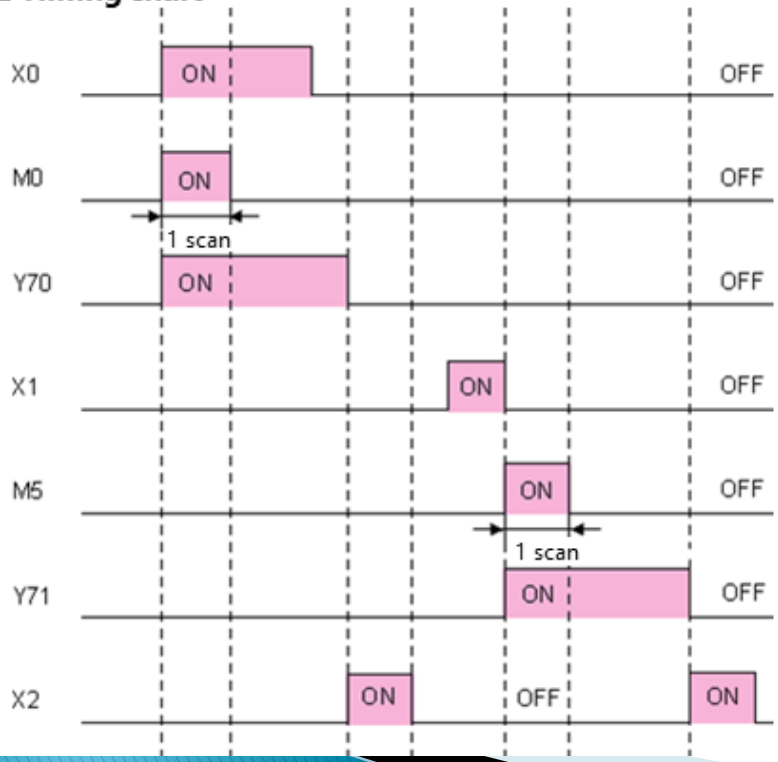
At the rising edge of X0 (OFF to ON), M0 turns ON for 1 scan
At the falling edge of X0 (ON to OFF), M5 turns ON for 1 scan

2.5 Output as pulses

Instruction codes and functions

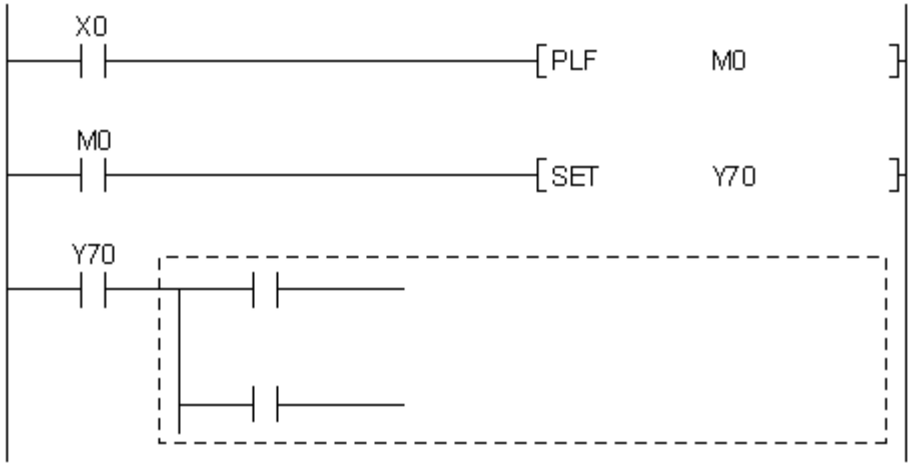
Symbol	Function
	Output at rising edge (PLS) Data is output to a specified device at the 1st scan after satisfying the input condition.
	Output at falling edge (PLF) Data is output to a specified device at the 1st scan after the input condition becomes unsatisfied.

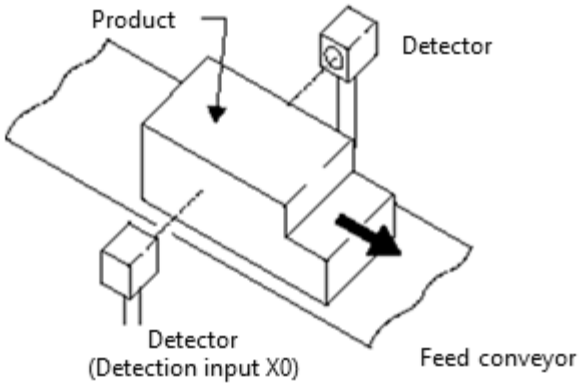
Timing chart



2.5.1 Application example of pulse outputs

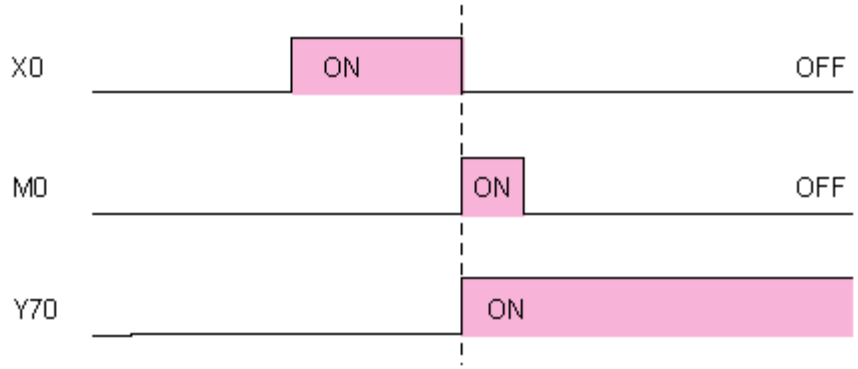
■ **Ladder program**





Pulse output is used to detect the passage of moving objects.
When the passage of a product is detected, the subsequent process is initiated.

■ **Timing chart**



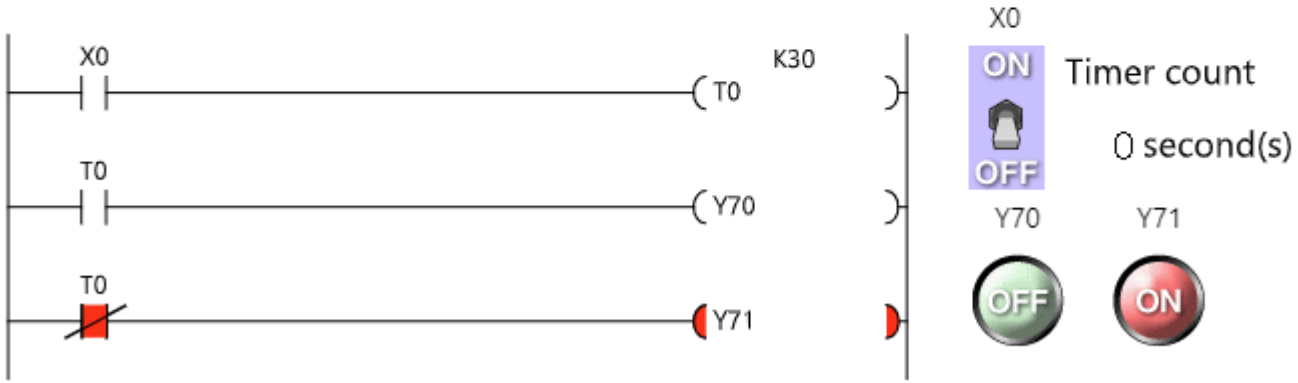
2.6 Time measurement

An OUT instruction and a timer device (T) are used for time measurement.
When the input condition is satisfied (ON), time measurement starts. When the time period reaches a specified value, a timer device (T) is turned ON.
If the input condition is not satisfied (OFF) or the timer device (T) is reset with the RST instruction, the elapsed time and output are initialized.

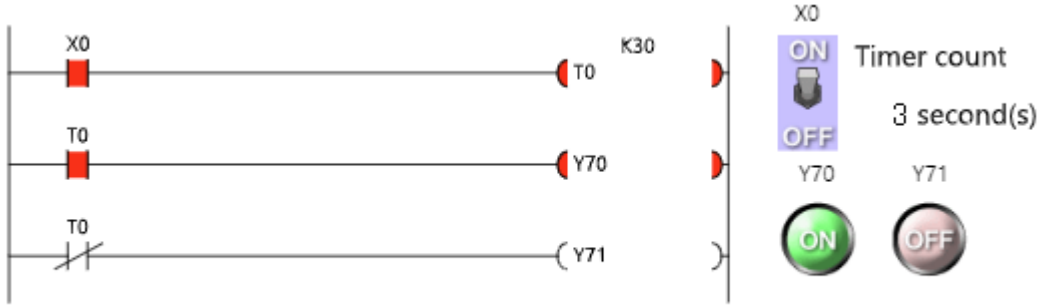
The status of the timer device (T) can be used as an input condition in other parts of the program.

■ Ladder program and operation

Simulate the timer operation by clicking the input switch shown on the right.



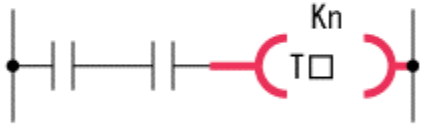
X0 turns ON, then after 3 seconds, Y70 turns ON and Y71 turns OFF.



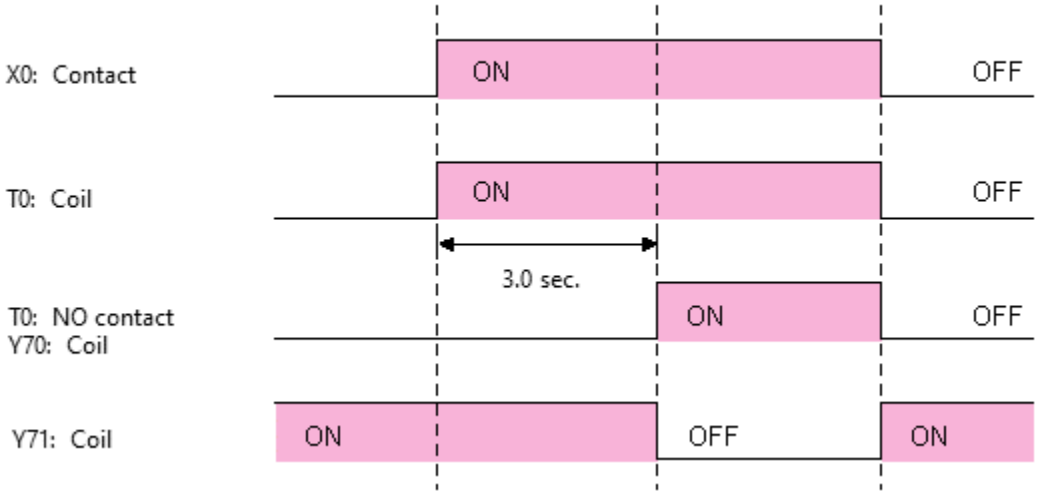
X0 turns ON, then after 3 seconds, Y70 turns ON and Y71 turns OFF.

2.6 Time measurement

■Instruction code and function

Symbol	Function
 □ : Device number	Timer operation A timer device (T) is used with a coil output (OUT) to measure how long the condition is being satisfied (being ON). A timeout occurs after a specified time period. Concurrently with the timeout, the timer (T0) turns ON. The timer set value is indicated by "Kn" (n: decimal). Timers are often used as an on-delay timer, which specifies time after a certain condition is satisfied.

■Timing chart



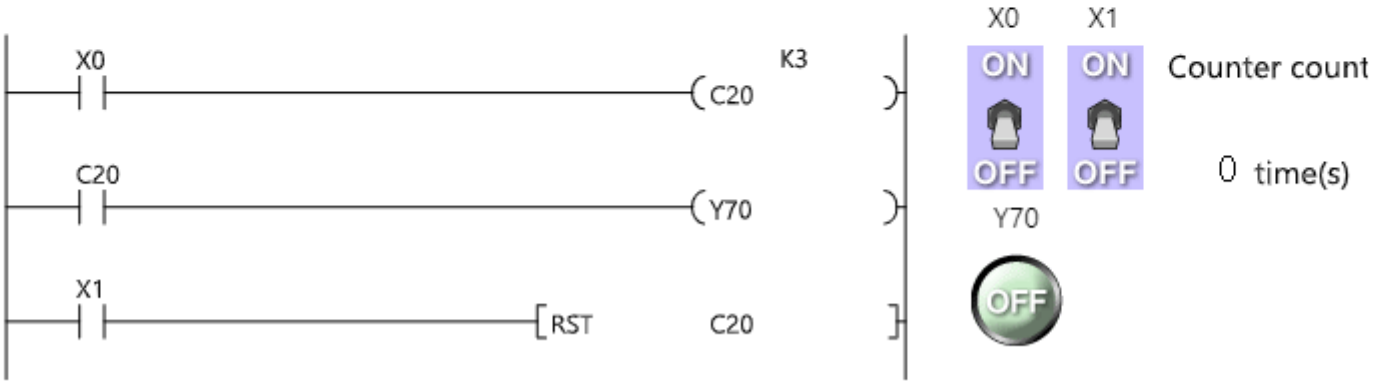
2.7 Counting

An OUT instruction and a counter device (C) are used for counting.
When an input condition is satisfied, the count increases, and when the count reaches to a specified value, the specified counter device (C) is turned ON.
If the counter device (C) is reset with the RST instruction, the count and the device status are initialized.

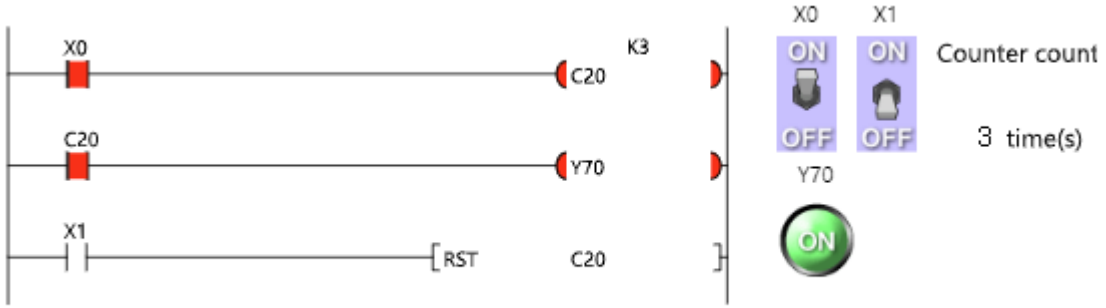
The status of the counter device (C) can be used as an input condition in other parts of the program.

■ Ladder program and operation

Simulate a counter operation by clicking the input switch shown on the right.



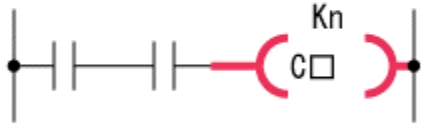
The value in C20 is incremented every time X0 turns ON. When the count reaches 3 (count-out), Y70 turns ON.



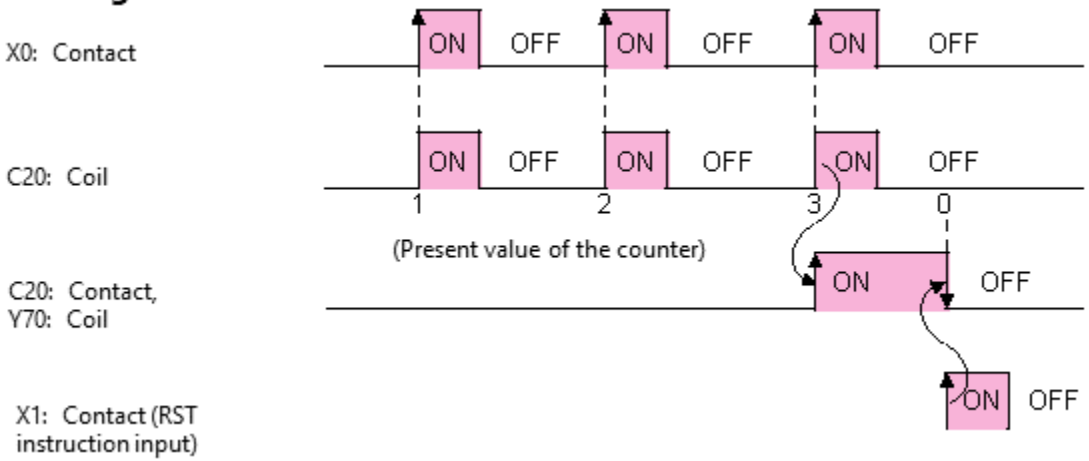
The value in C20 is incremented every time X0 turns ON. When the count reaches 3 (count-out), Y70 turns ON.

2.7 Counting

■ Instruction code and function

Symbol	Function
 □ : Device number	Counter Combined with a coil output (OUT), the counter counts up (one by one) the number of times the condition is satisfied. A count-out occurs when the count reaches the specified number, and the counter contact turns ON. The counter set value is indicated by "Kn" (n: decimal).

■ Timing chart



Chapter 3 Word device instructions

This chapter explains about instructions using word devices.

Word devices are useful in controlling time, count, and values input from external equipment.
Word devices can make control programs more responsive to the real operation.

- Simulate basic program operations to understand operation of main instructions
- From the simulation, understand the roles of instructions and the processing executed in the programmable controller

3.1 Moving data to a word device

3.1.2 Moving data to multiple word devices at once

3.1.3 Bit device digit

3.2 Comparing numeric values

3.3 Arithmetic operations

This section explains basic arithmetic operations of word (numerical) devices.

■ **Addition and subtraction**

Arithmetic operations using addition (+) and subtraction (-) symbols.

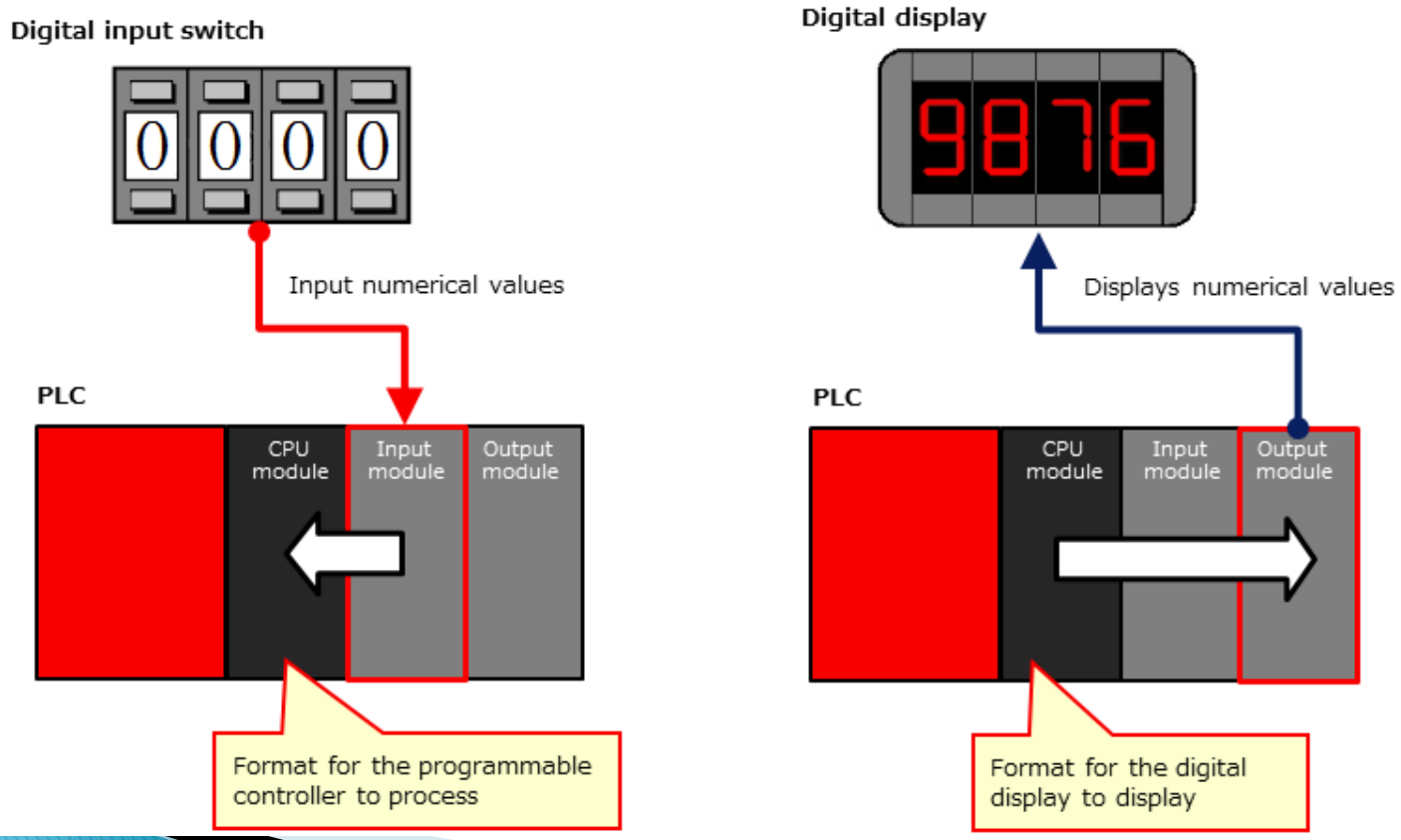
■ **Multiplication and division**

Arithmetic operations using multiplication (*) and division (/) symbols.

3.4 Data transmission/receipt between the PLC and I/O devices

A digital input switch is an input device that inputs data to a programmable controller in numerical values. A digital display is an output device that displays data received from a programmable controller in numerical values.

Data received from a digital input switch must be formatted so that it can be processed by a programmable controller. Likewise, the data output to a digital display must be formatted into a readable format for the digital display.



1.11 Development and maintenance of sequence programs

GX Works3 is an engineering tool for creating and maintaining sequence programs for PLCs including the MELSEC iQ-F Series and MELSEC iQ-R Series.
By connecting a Windows® personal computer in which GX Works3 is installed and an Ethernet port or the like, built into the CPU module, you can develop programs, check program operations, write programs to the CPU module, and check the module status.



END OF CHAPTER 4

Dr. Yılmaz KALKAN