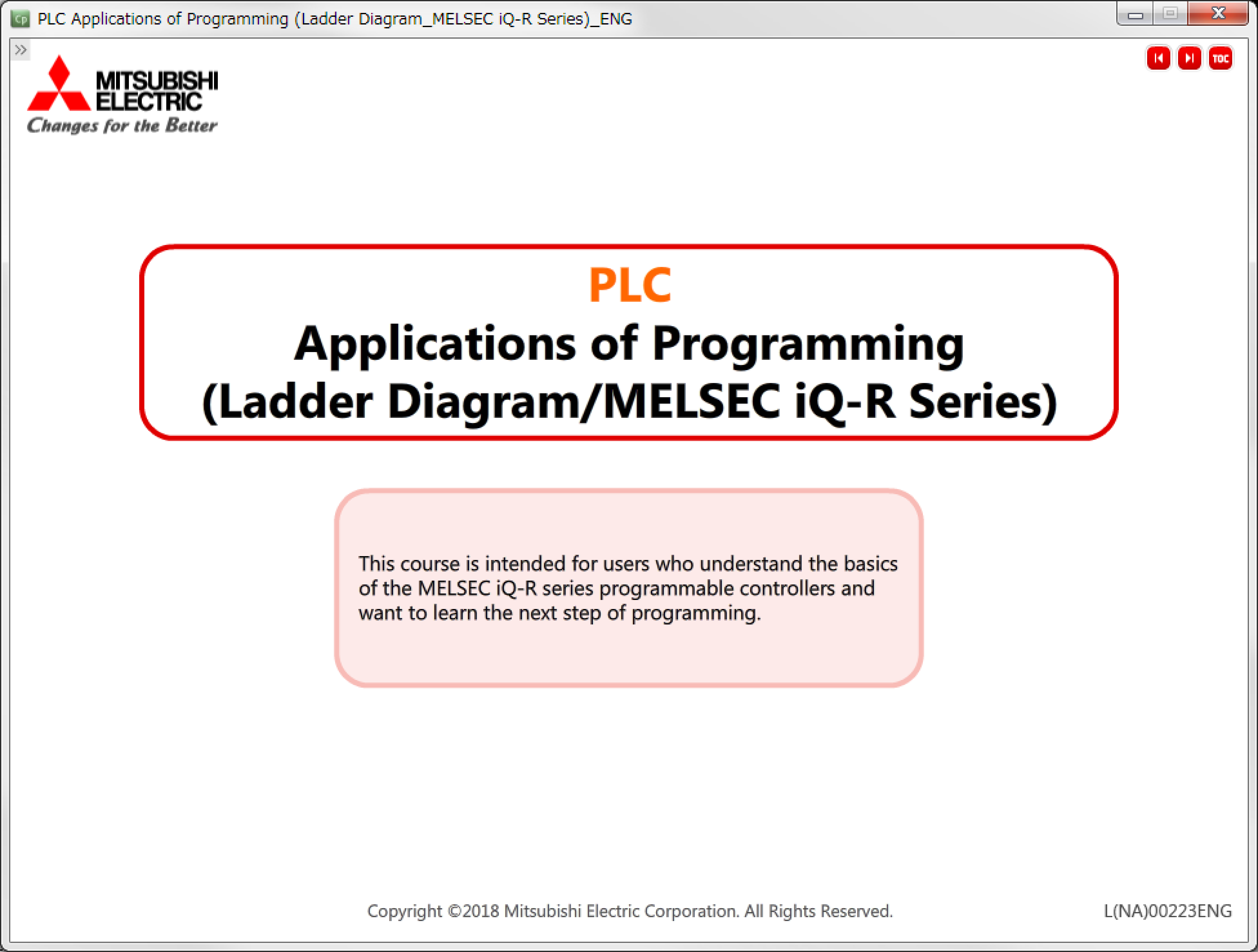
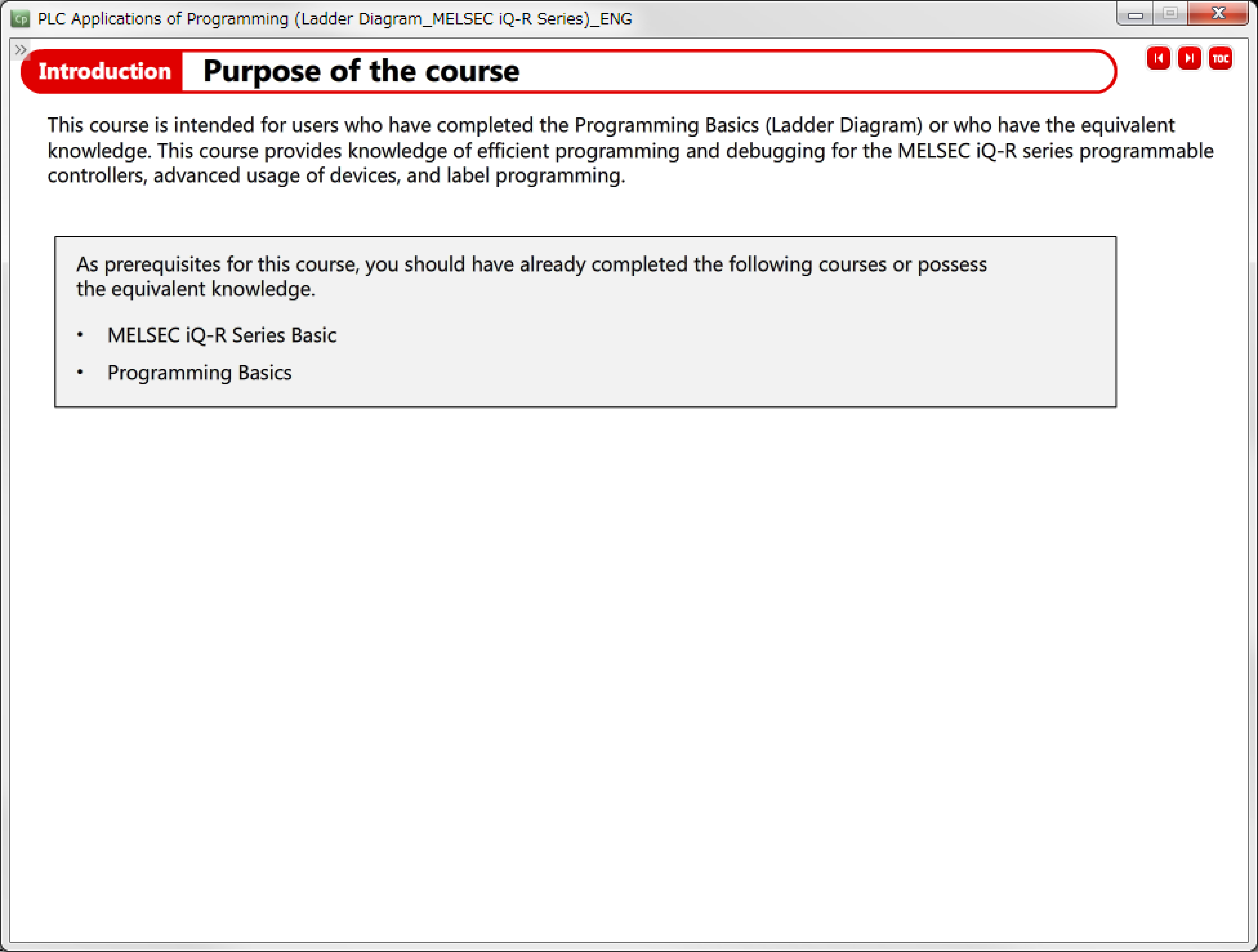

PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
MITSUBISHI ELECTRIC
Changes for the Better
TOC

PLC
Applications of Programming (Ladder Diagram/MELSEC iQ-R Series)

This course is intended for users who understand the basics of the MELSEC iQ-R series programmable controllers and want to learn the next step of programming.

L(NA)00223ENG



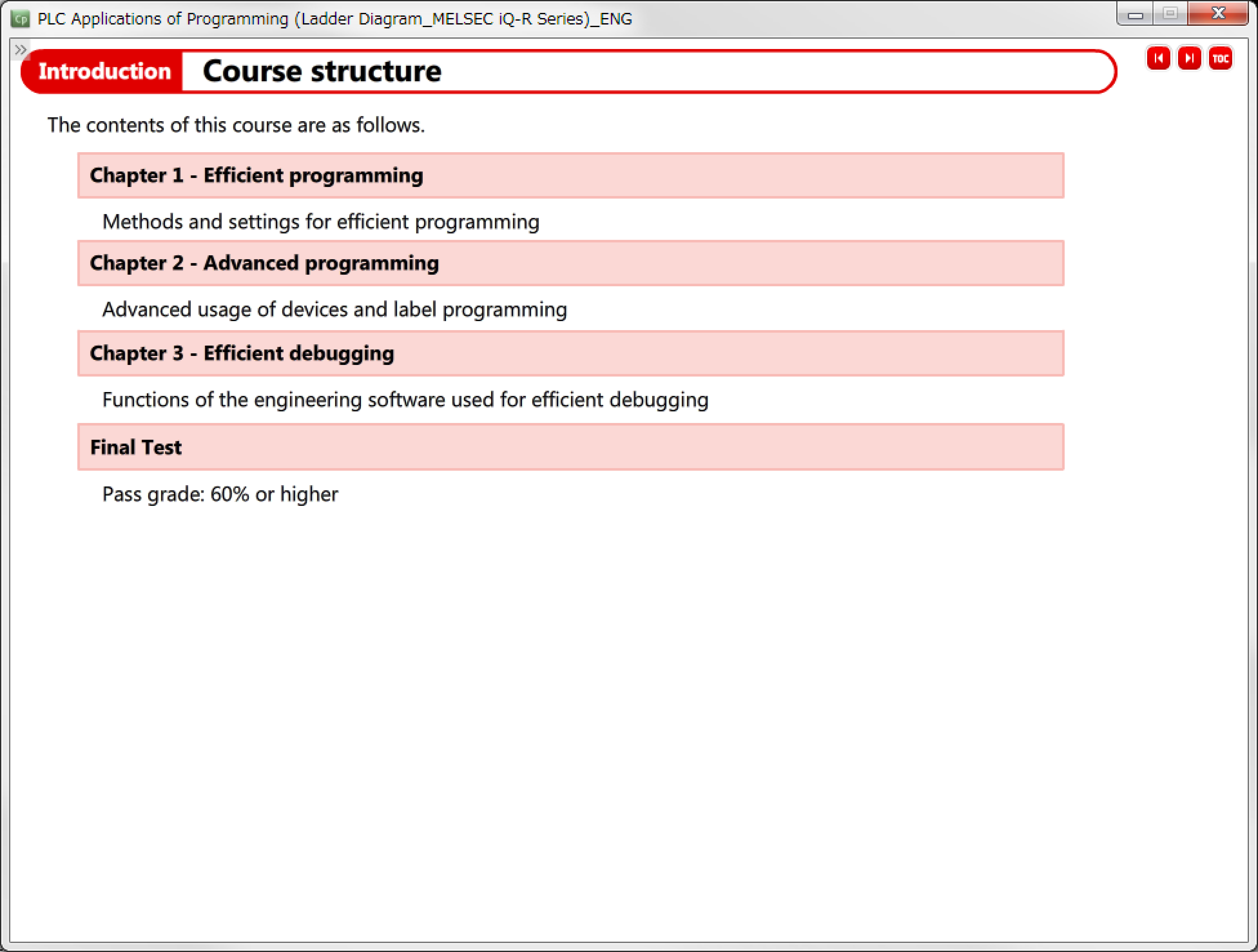
Introduction

Purpose of the course

This course is intended for users who have completed the Programming Basics (Ladder Diagram) or who have the equivalent knowledge. This course provides knowledge of efficient programming and debugging for the MELSEC iQ-R series programmable controllers, advanced usage of devices, and label programming.

As prerequisites for this course, you should have already completed the following courses or possess the equivalent knowledge.

- MELSEC iQ-R Series Basic
- Programming Basics



Introduction Course structure

The contents of this course are as follows.

Chapter 1 - Efficient programming

Methods and settings for efficient programming

Chapter 2 - Advanced programming

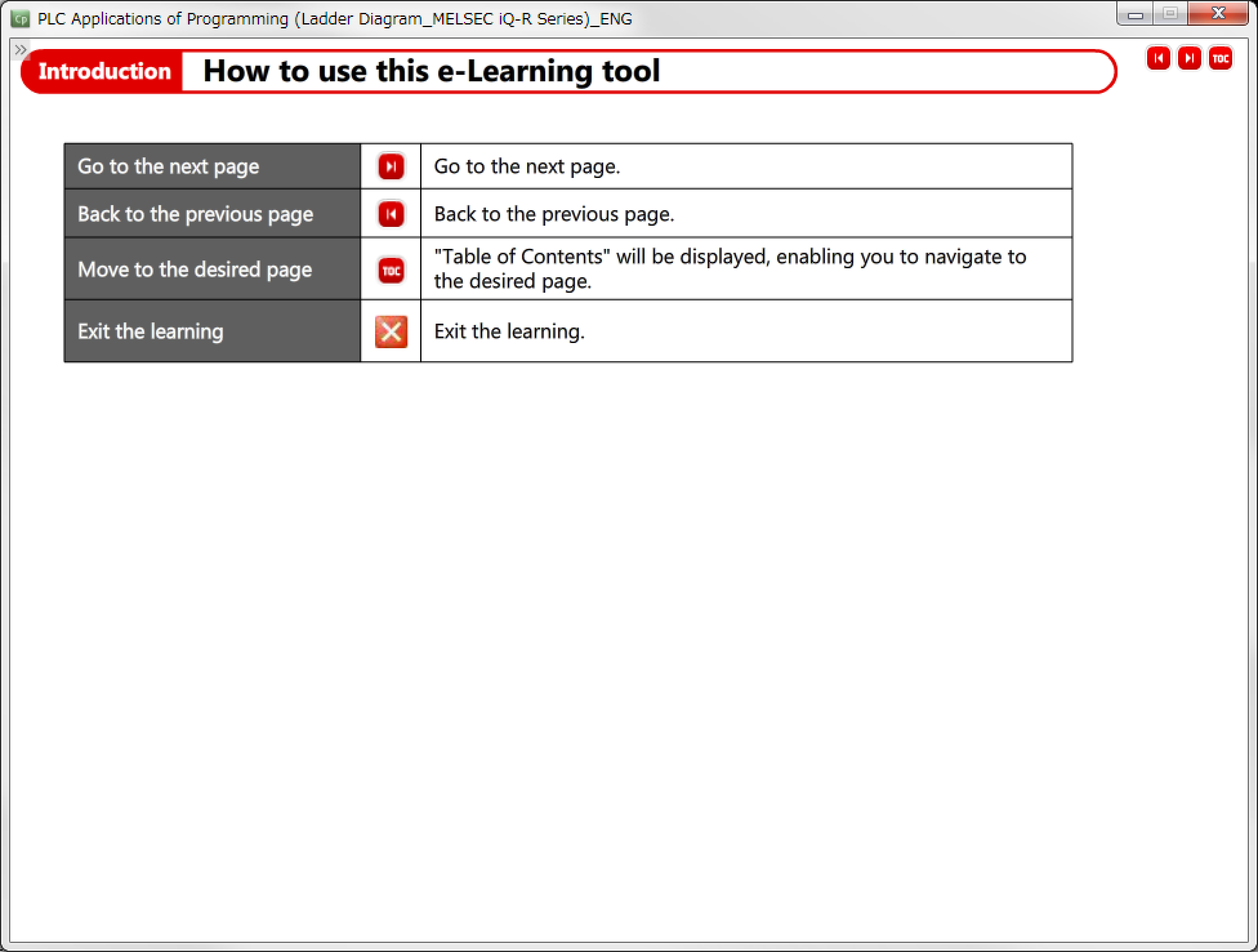
Advanced usage of devices and label programming

Chapter 3 - Efficient debugging

Functions of the engineering software used for efficient debugging

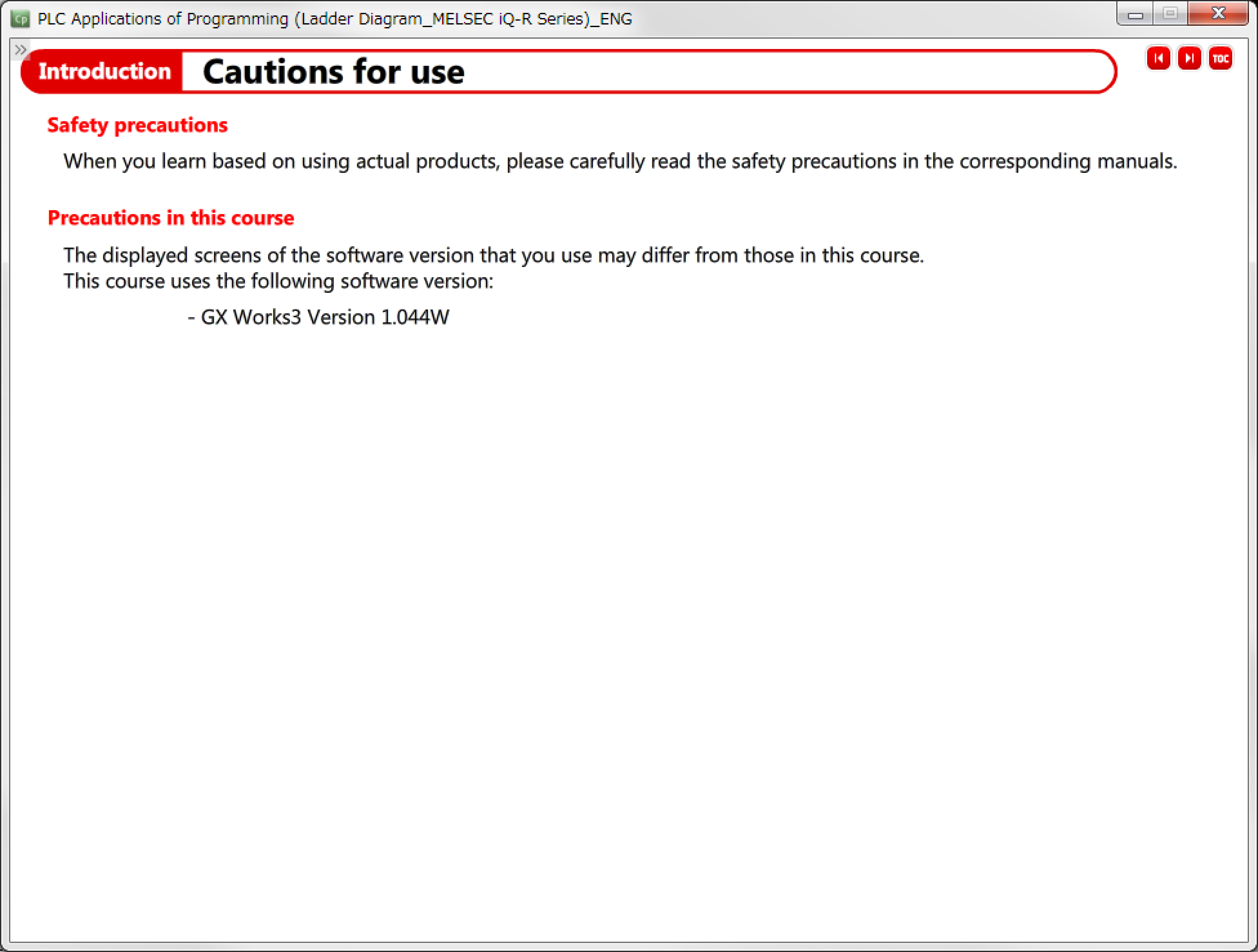
Final Test

Pass grade: 60% or higher


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
⏮
⏭
TOC

Introduction How to use this e-Learning tool

Go to the next page	⏭	Go to the next page.
Back to the previous page	⏮	Back to the previous page.
Move to the desired page	TOC	"Table of Contents" will be displayed, enabling you to navigate to the desired page.
Exit the learning	☒	Exit the learning.



Introduction Cautions for use

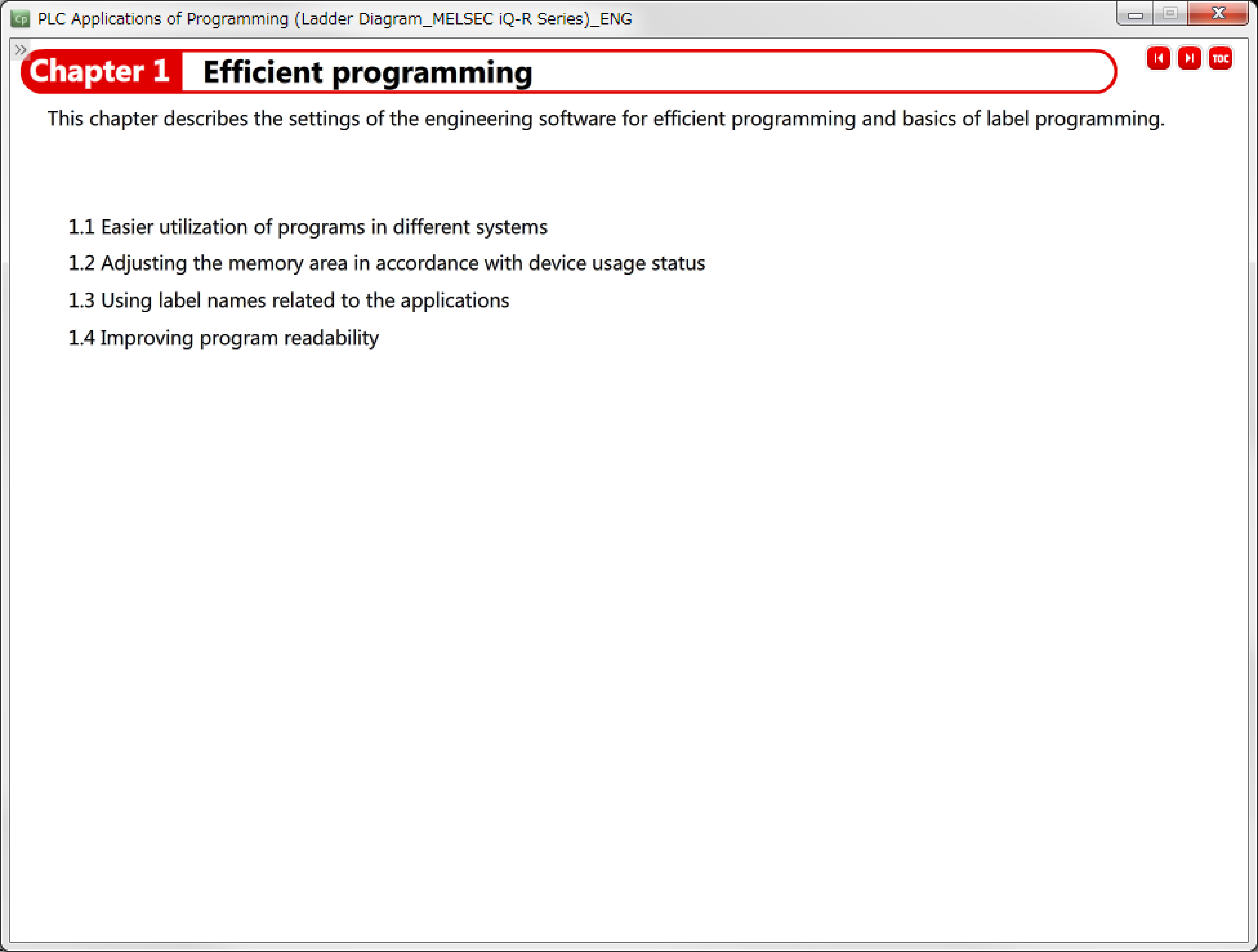
Safety precautions

When you learn based on using actual products, please carefully read the safety precautions in the corresponding manuals.

Precautions in this course

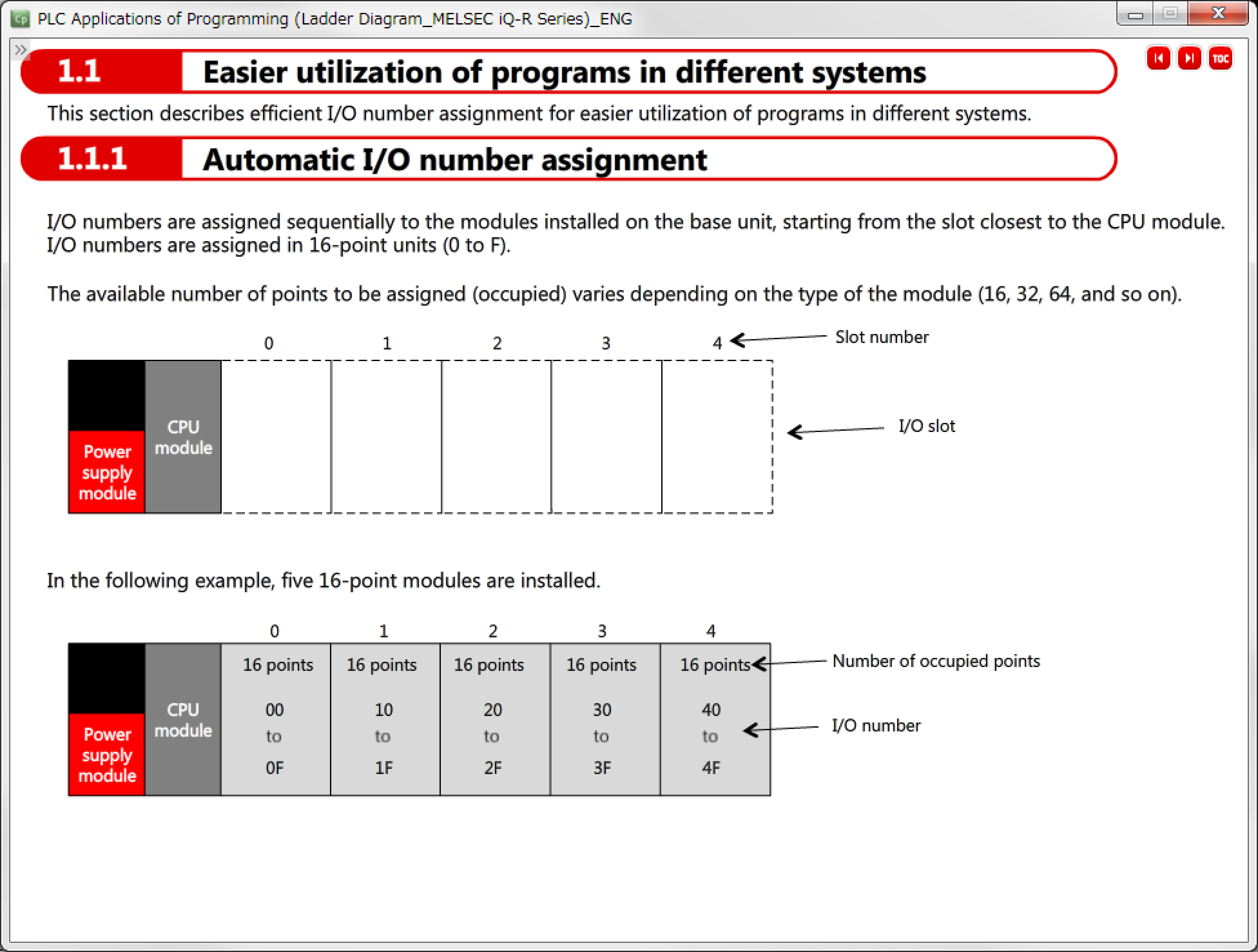
The displayed screens of the software version that you use may differ from those in this course.
This course uses the following software version:

- GX Works3 Version 1.044W



Chapter 1 Efficient programming

This chapter describes the settings of the engineering software for efficient programming and basics of label programming.



PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC

1.1 Easier utilization of programs in different systems

This section describes efficient I/O number assignment for easier utilization of programs in different systems.

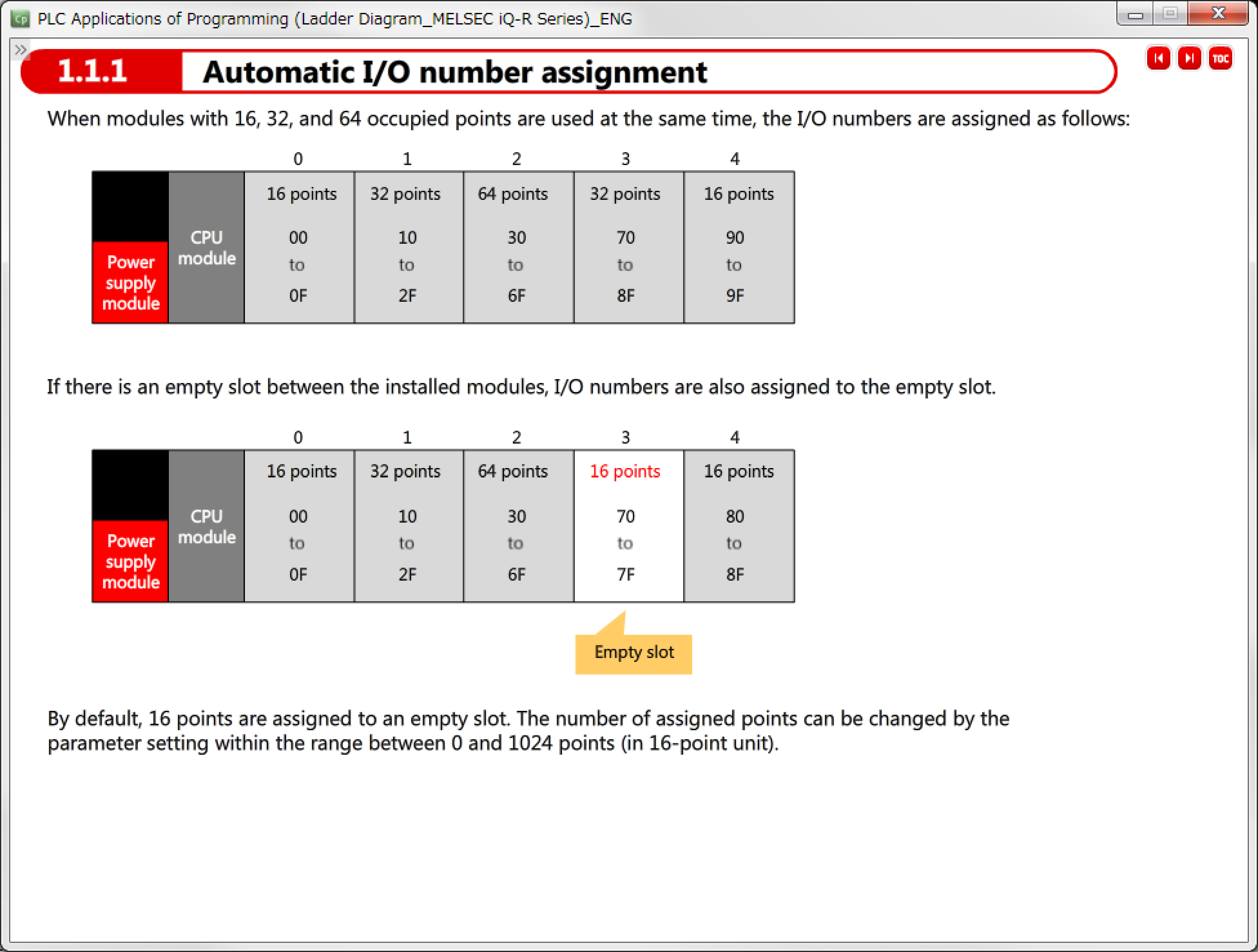
1.1.1 Automatic I/O number assignment

I/O numbers are assigned sequentially to the modules installed on the base unit, starting from the slot closest to the CPU module. I/O numbers are assigned in 16-point units (0 to F).

The available number of points to be assigned (occupied) varies depending on the type of the module (16, 32, 64, and so on).

In the following example, five 16-point modules are installed.

		0	1	2	3	4	
Power supply module	CPU module	16 points	16 points	16 points	16 points	16 points	Number of occupied points
		00 to 0F	10 to 1F	20 to 2F	30 to 3F	40 to 4F	I/O number



1.1.1 Automatic I/O number assignment

When modules with 16, 32, and 64 occupied points are used at the same time, the I/O numbers are assigned as follows:

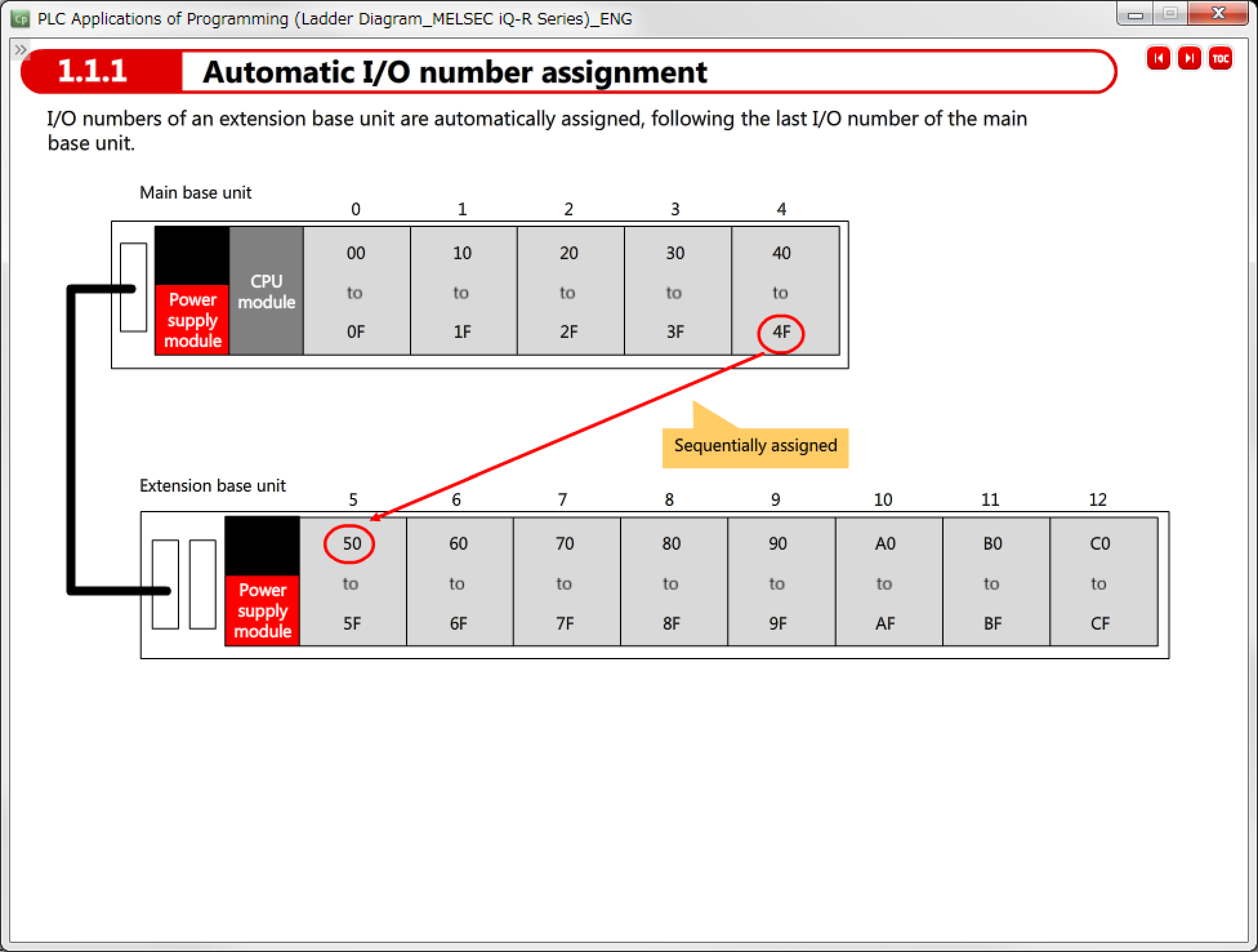
		0	1	2	3	4
Power supply module	CPU module	16 points 00 to 0F	32 points 10 to 2F	64 points 30 to 6F	32 points 70 to 8F	16 points 90 to 9F

If there is an empty slot between the installed modules, I/O numbers are also assigned to the empty slot.

		0	1	2	3	4
Power supply module	CPU module	16 points 00 to 0F	32 points 10 to 2F	64 points 30 to 6F	16 points 70 to 7F	16 points 80 to 8F

Empty slot

By default, 16 points are assigned to an empty slot. The number of assigned points can be changed by the parameter setting within the range between 0 and 1024 points (in 16-point unit).


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC

1.1.1 Automatic I/O number assignment

I/O numbers of an extension base unit are automatically assigned, following the last I/O number of the main base unit.

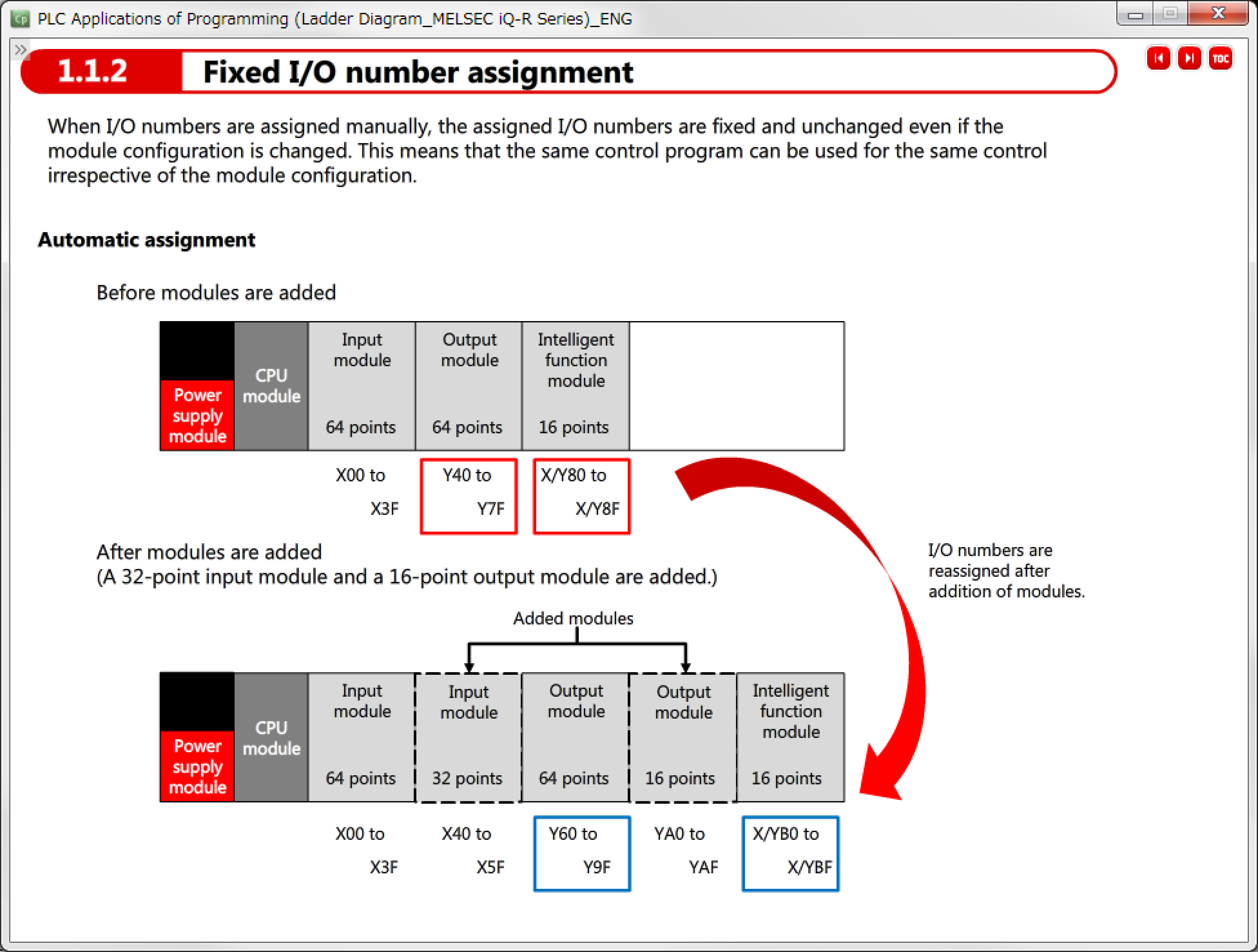
Main base unit

		0	1	2	3	4
Power supply module	CPU module	00 to 0F	10 to 1F	20 to 2F	30 to 3F	40 to 4F

Sequentially assigned

Extension base unit

	5	6	7	8	9	10	11	12
Power supply module	50 to 5F	60 to 6F	70 to 7F	80 to 8F	90 to 9F	A0 to AF	B0 to BF	C0 to CF


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC

1.1.2 Fixed I/O number assignment

When I/O numbers are assigned manually, the assigned I/O numbers are fixed and unchanged even if the module configuration is changed. This means that the same control program can be used for the same control irrespective of the module configuration.

Automatic assignment

Before modules are added

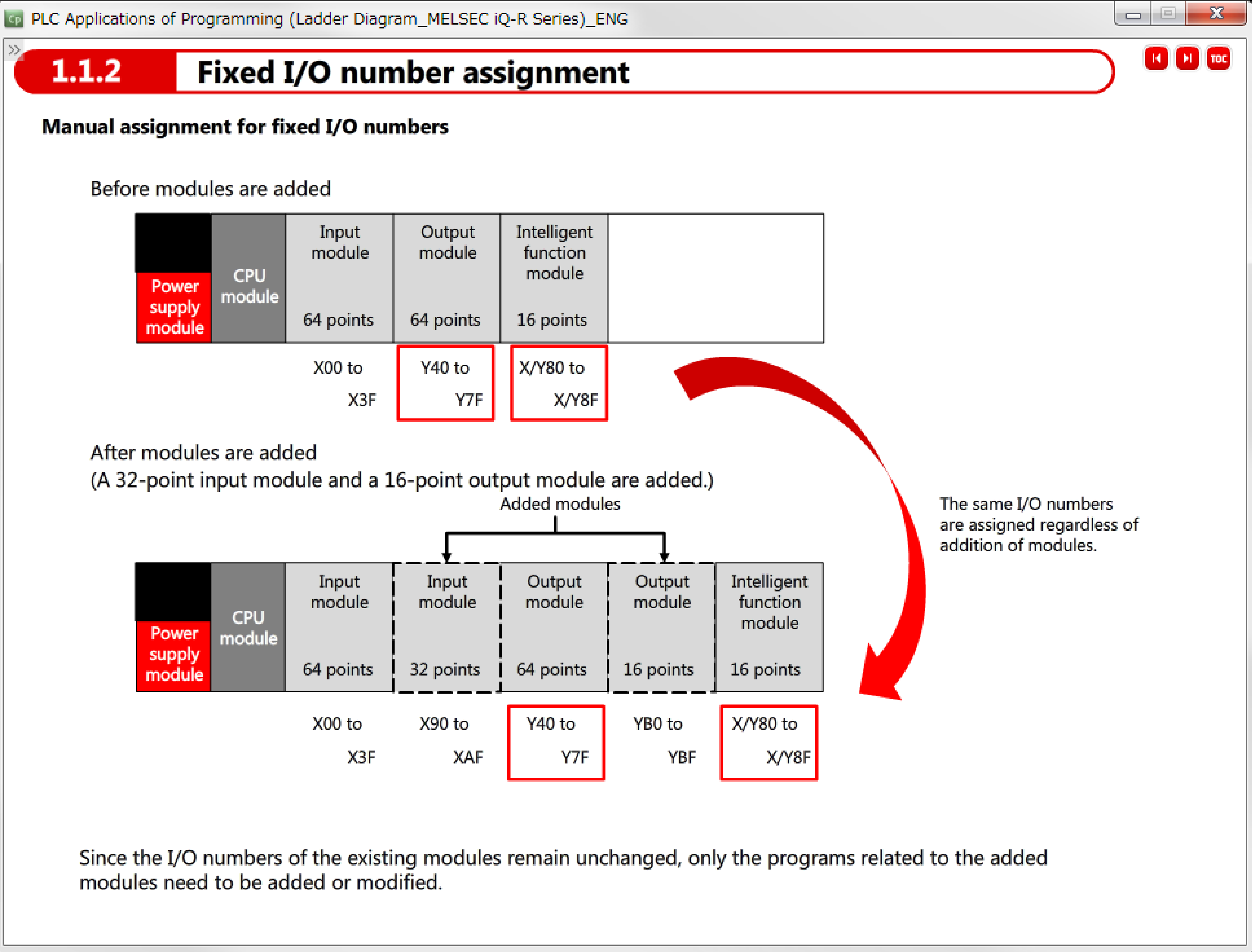
Power supply module	CPU module	Input module 64 points	Output module 64 points	Intelligent function module 16 points	
		X00 to X3F	Y40 to Y7F	X/Y80 to X/Y8F	

After modules are added
(A 32-point input module and a 16-point output module are added.)

Added modules

Power supply module	CPU module	Input module 64 points	Input module 32 points	Output module 64 points	Output module 16 points	Intelligent function module 16 points
		X00 to X3F	X40 to X5F	Y60 to Y9F	YA0 to YAF	X/YB0 to X/YBF

I/O numbers are reassigned after addition of modules.



1.1.2 Fixed I/O number assignment

Manual assignment for fixed I/O numbers

Before modules are added

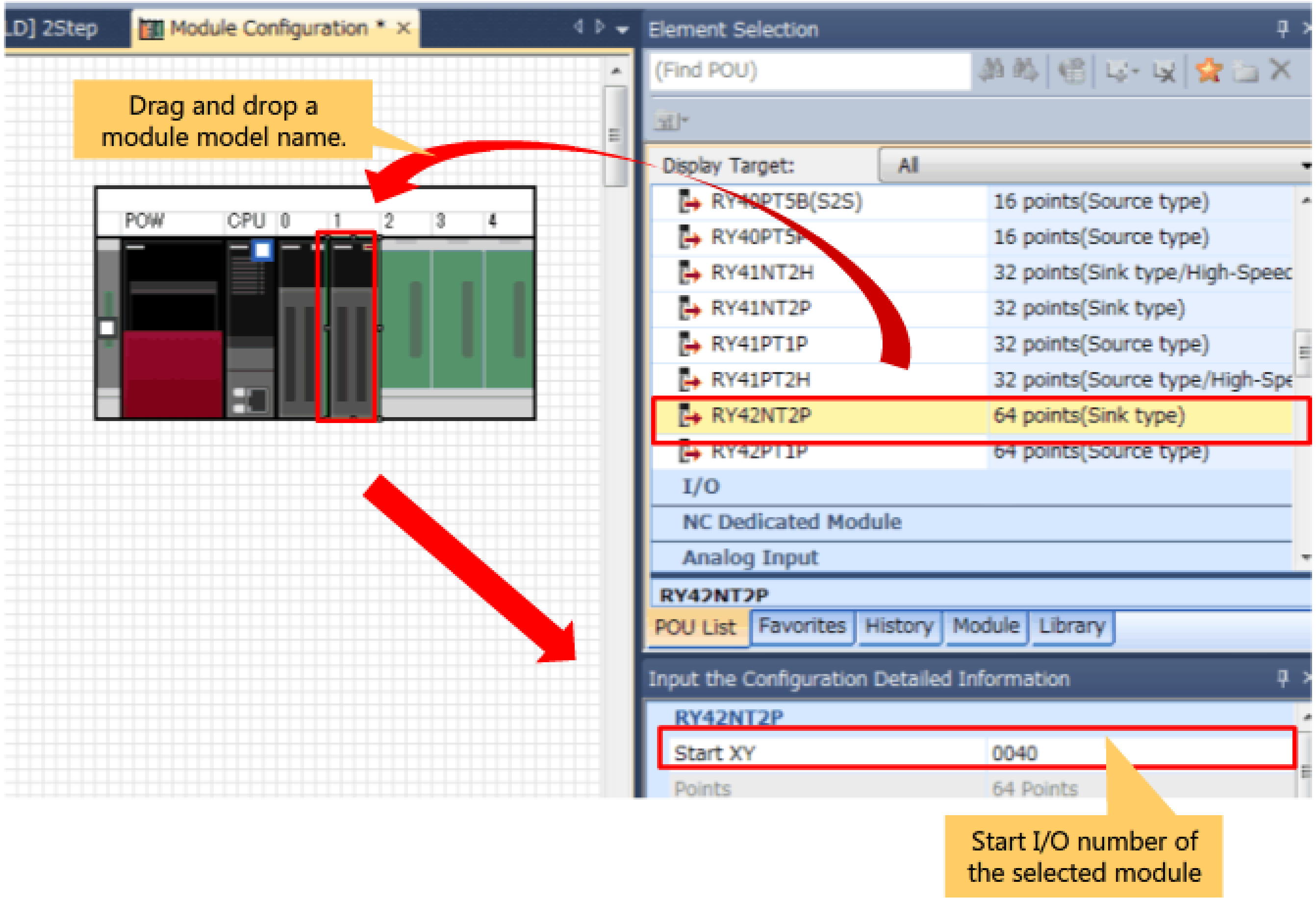
After modules are added
(A 32-point input module and a 16-point output module are added.)

The same I/O numbers are assigned regardless of addition of modules.

Since the I/O numbers of the existing modules remain unchanged, only the programs related to the added modules need to be added or modified.

1.1.3 Automatic I/O number assignment using the module configuration diagram

Module configuration can be set using the module configuration diagram of the engineering software, MELSOFT GX Works3.
Select a module model name, and drag and drop it on a slot to place the module there.
I/O numbers are sequentially assigned to modules, starting from the one closest to the CPU module as displayed in the the diagram.
Select a placed module to view the start I/O number of the module.



RY40PT5B(S2S)	16 points(Source type)
RY40PT5P	16 points(Source type)
RY41NT2H	32 points(Sink type/High-Speec
RY41NT2P	32 points(Sink type)
RY41PT1P	32 points(Source type)
RY41PT2H	32 points(Source type/High-Spe
RY42NT2P	64 points(Sink type)
RY42PT1P	64 points(Source type)

RY42NT2P

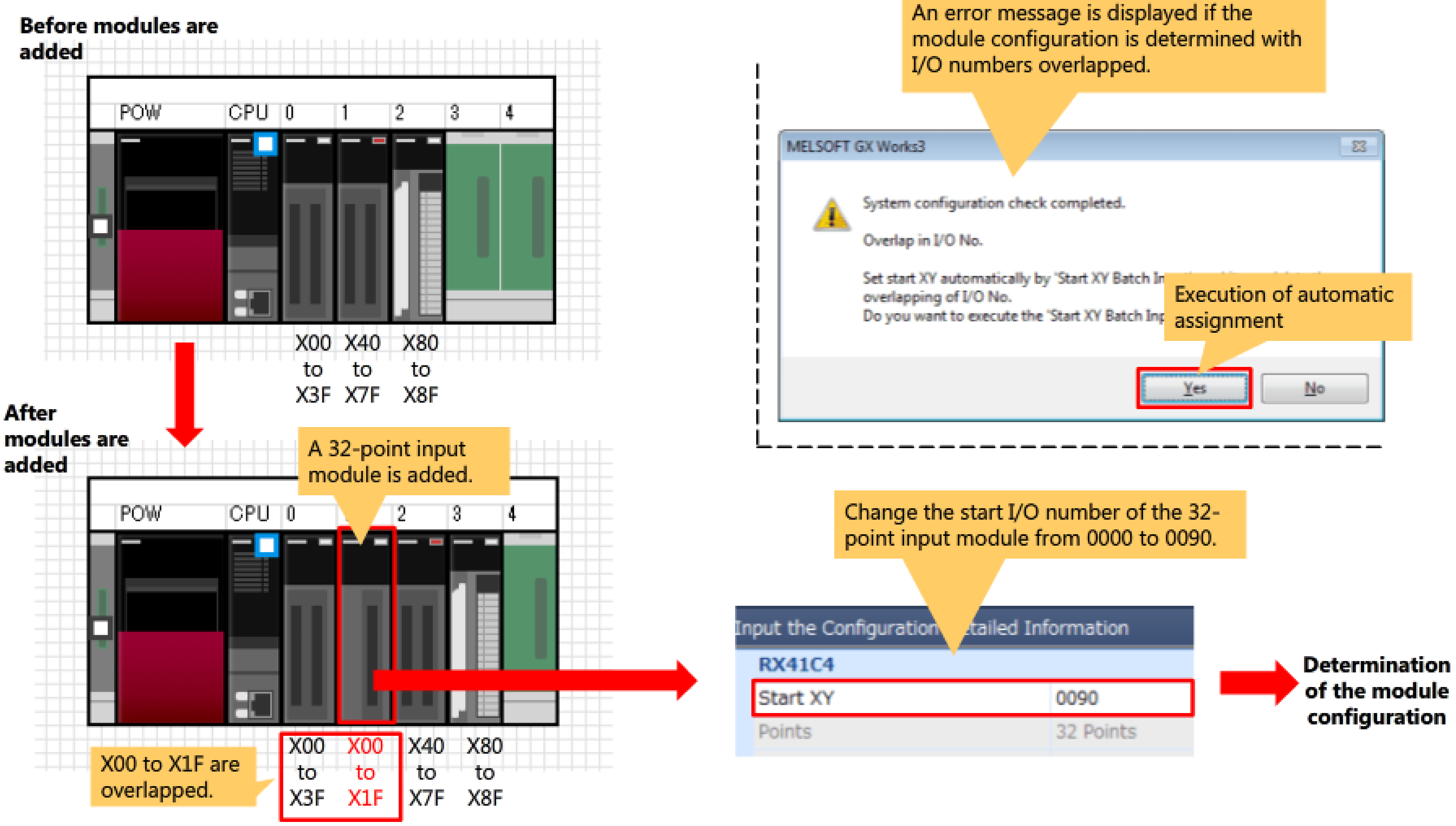
RY42NT2P

Start XY	0040
Points	64 Points

1.1.4 Manual I/O number assignment using the module configuration diagram

The following example describes how to assign I/O numbers manually using the module configuration diagram of GX Works3.

When the module configuration is changed by adding a new module to the module configuration diagram, the I/O number of the added module overlaps with existing numbers if the placement of the existing modules is changed. Edit the I/O numbers so that they do not overlap, and determine the module configuration.
An error message is displayed if the module configuration is determined with I/O numbers overlapped. At this time, automatic assignment can be executed from the displayed error message window.



A 32-point input module is added.

X00 to X1F are overlapped.

An error message is displayed if the module configuration is determined with I/O numbers overlapped.

System configuration check completed.

Overlap in I/O No.

Set start XY automatically by 'Start XY Batch In
overlapping of I/O No.
Do you want to execute the 'Start XY Batch Inp

Execution of automatic assignment

Change the start I/O number of the 32-point input module from 0000 to 0090.

RX41C4	
Start XY	0090
Points	32 Points

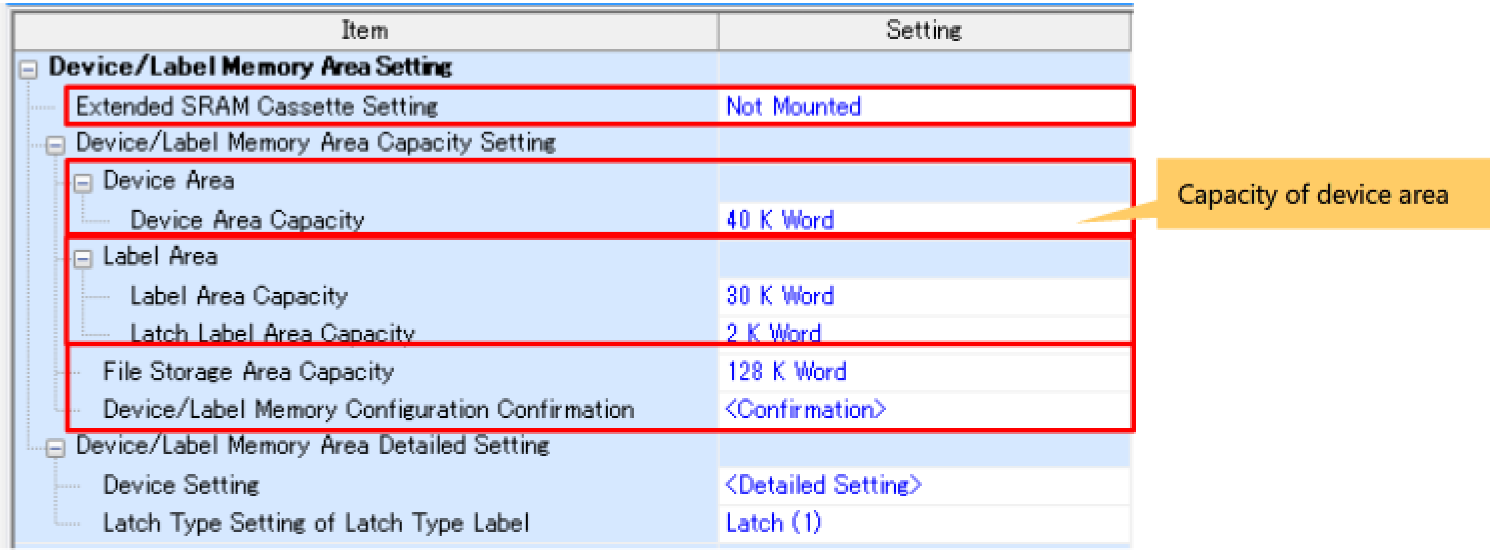
1.2 Adjusting the memory area in accordance with device usage status

1.2.1 Device/label memory area settings

The number of device points used for CPU modules varies with the type of CPU module. The initial number of device points is assigned based on the device area capacity of CPU module.

Capacity of the device area of the R04CPU is 40K words.
Reducing the unused areas increases the number of device points from the initial value.

The following figure shows the "Device/Label Memory Area Setting" window as parameters for adjusting memory area.
Capacity of the device area increases when the capacity of the label area or file storage area is reduced.
In addition, the capacity of the entire device/label memory area can be extended by using an extended SRAM cassette.

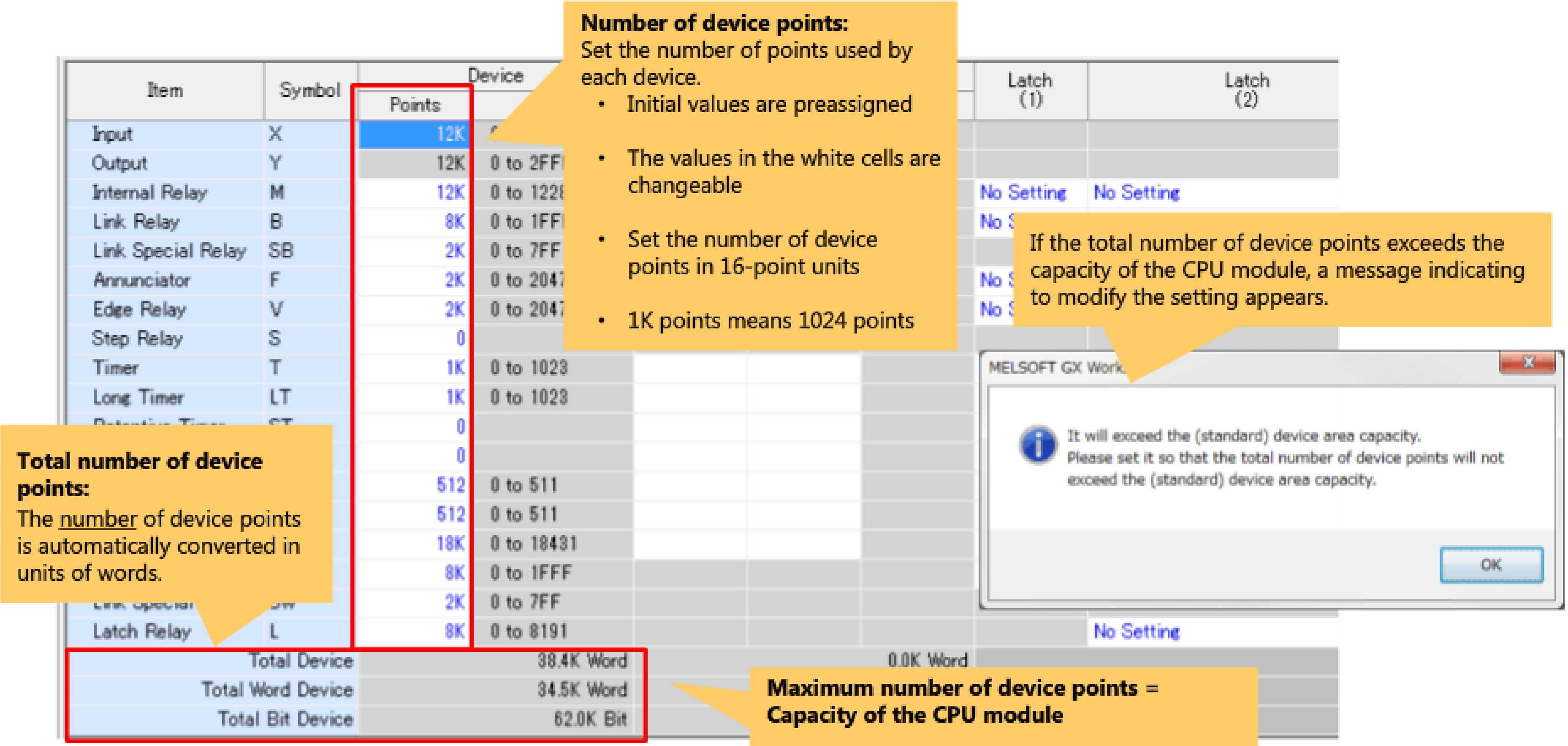


Item	Setting
Device/Label Memory Area Setting	
Extended SRAM Cassette Setting	Not Mounted
Device/Label Memory Area Capacity Setting	
Device Area	
Device Area Capacity	40 K Word
Label Area	
Label Area Capacity	30 K Word
Latch Label Area Capacity	2 K Word
File Storage Area Capacity	128 K Word
Device/Label Memory Configuration Confirmation	<Confirmation>
Device/Label Memory Area Detailed Setting	
Device Setting	<Detailed Setting>
Latch Type Setting of Latch Type Label	Latch (1)

Capacity of device area

1.2.2 Device settings

The number of device points assigned to each device can be changed in the "Device Setting" window.
Initial value of some devices is 0 point. Assign the number of points when using such devices.



Item	Symbol	Device Points		Latch (1)	Latch (2)
Input	X	12K			
Output	Y	12K	0 to 2FF		
Internal Relay	M	12K	0 to 122	No Setting	No Setting
Link Relay	B	8K	0 to 1FF	No S	
Link Special Relay	SB	2K	0 to 7FF		
Annunciator	F	2K	0 to 204	No S	
Edge Relay	V	2K	0 to 204	No S	
Step Relay	S	0			
Timer	T	1K	0 to 1023		
Long Timer	LT	1K	0 to 1023		
Retentive Timer	ST	0			
		0			
		512	0 to 511		
		512	0 to 511		
		18K	0 to 18431		
		8K	0 to 1FFF		
Link Special	SW	2K	0 to 7FF		
Latch Relay	L	8K	0 to 8191		No Setting
Total Device		38.4K Word	0.0K Word		
Total Word Device		34.5K Word			
Total Bit Device		62.0K Bit			

MELSOFT GX Work

It will exceed the (standard) device area capacity.
Please set it so that the total number of device points will not exceed the (standard) device area capacity.

Number of device points:
Set the number of points used by each device.

- Initial values are preassigned
- The values in the white cells are changeable
- Set the number of device points in 16-point units
- 1K points means 1024 points

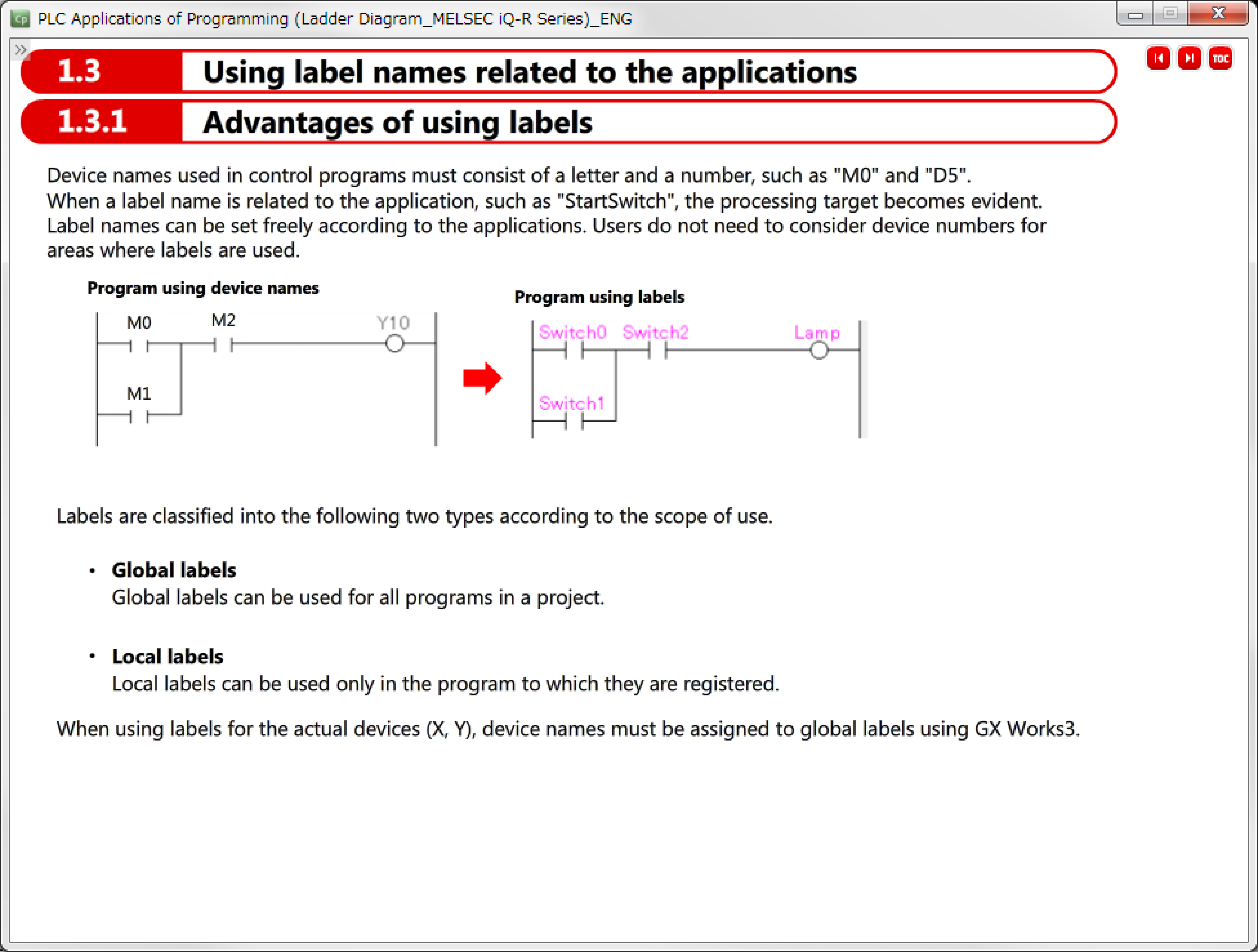
If the total number of device points exceeds the capacity of the CPU module, a message indicating to modify the setting appears.

Total number of device points:
The number of device points is automatically converted in units of words.

Maximum number of device points = Capacity of the CPU module

For example, the CPU module capacity of the R04CPU is 40K words.

"Device Setting" window



1.3 Using label names related to the applications

1.3.1 Advantages of using labels

Device names used in control programs must consist of a letter and a number, such as "M0" and "D5".
When a label name is related to the application, such as "StartSwitch", the processing target becomes evident.
Label names can be set freely according to the applications. Users do not need to consider device numbers for areas where labels are used.

Program using device names

Program using labels

Labels are classified into the following two types according to the scope of use.

- **Global labels**
  Global labels can be used for all programs in a project.

- **Local labels**
  Local labels can be used only in the program to which they are registered.

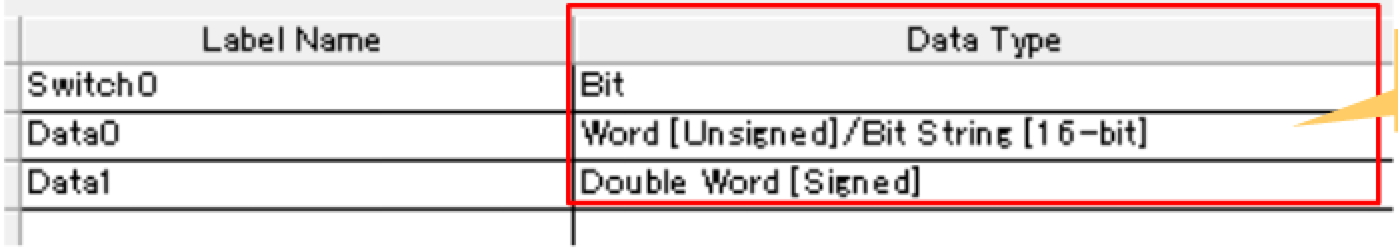
When using labels for the actual devices (X, Y), device names must be assigned to global labels using GX Works3.

1.3.2 Data type of labels

A data type must be specified for each label to define the range of the values to be handled.
Data types include bit and integer, as shown below.

Data type		Data range
Bit type		On/off state of bit devices and true/false state of execution results
Integer type	Word (unsigned)	0 to 65,535
	Word (signed)	-32,768 to 32,767
	Double word (unsigned)	0 to 4,294,967,295
	Double word (signed)	-2,147,483,648 to 2,147,483,647

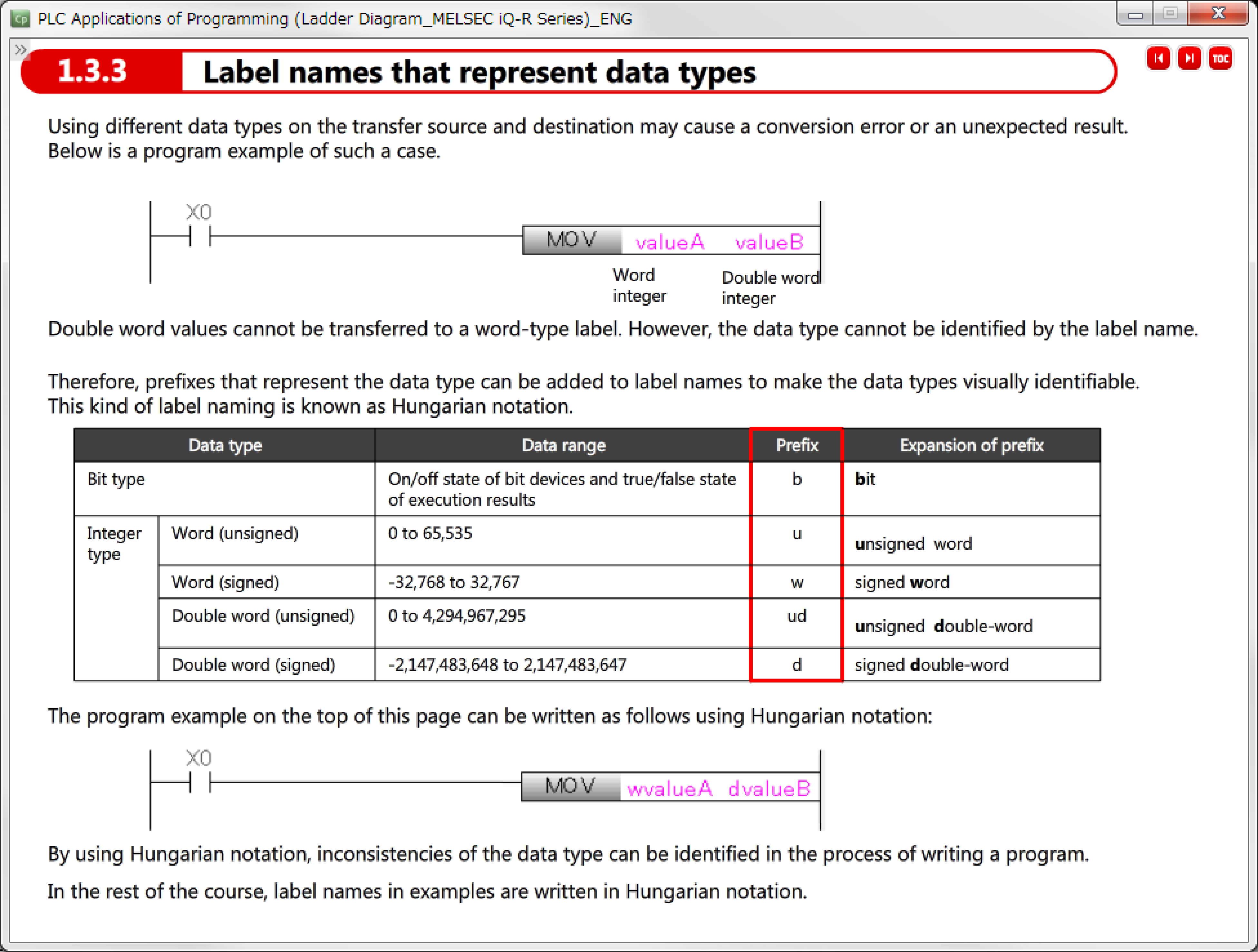
When using the integer type, select the word or double word type according to the data range, and select the signed or unsigned type according to the necessity to handle negative values.
Specify the data type of a label when setting a label name using GX Works3.



Label Name	Data Type
Switch0	Bit
Data0	Word [Unsigned]/Bit String [16-bit]
Data1	Double Word [Signed]

Specify the label value range.

Label setting window



1.3.3 Label names that represent data types

Using different data types on the transfer source and destination may cause a conversion error or an unexpected result.
Below is a program example of such a case.

Double word values cannot be transferred to a word-type label. However, the data type cannot be identified by the label name.

Therefore, prefixes that represent the data type can be added to label names to make the data types visually identifiable.
This kind of label naming is known as Hungarian notation.

Data type		Data range	Prefix	Expansion of prefix
Bit type		On/off state of bit devices and true/false state of execution results	b	**b**it
Integer type	Word (unsigned)	0 to 65,535	u	**u**nsigned word
	Word (signed)	-32,768 to 32,767	w	signed **w**ord
	Double word (unsigned)	0 to 4,294,967,295	ud	**u**nsigned **d**ouble-word
	Double word (signed)	-2,147,483,648 to 2,147,483,647	d	signed **d**ouble-word

The program example on the top of this page can be written as follows using Hungarian notation:

By using Hungarian notation, inconsistencies of the data type can be identified in the process of writing a program.

In the rest of the course, label names in examples are written in Hungarian notation.

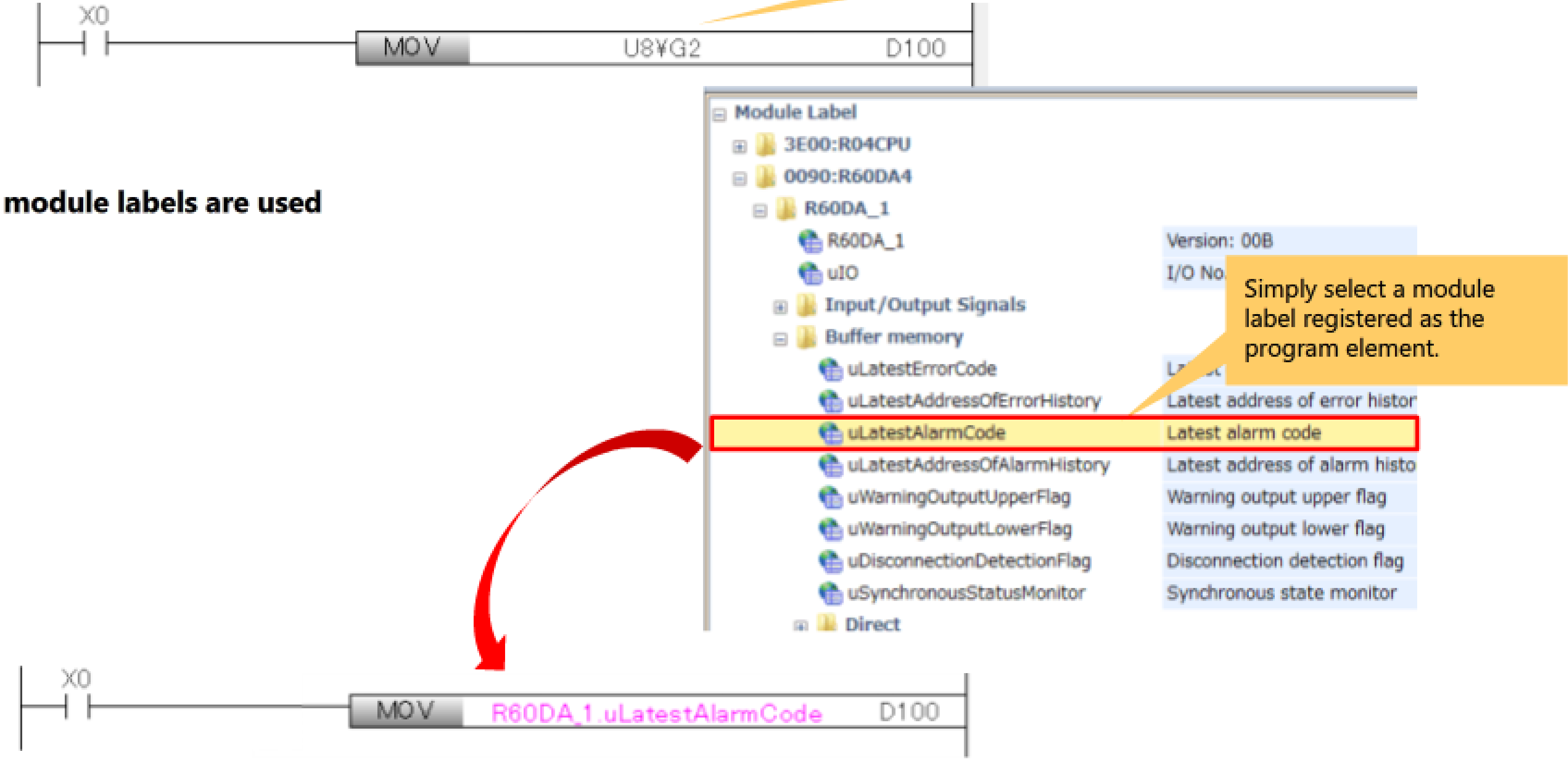
1.3.4 Using prepared labels

When the module configuration is set in the module configuration diagram, labels (module labels) representing module signals or setting values that correspond to the module installation positions are registered automatically.

For programming using device names, the device numbers and buffer memory addresses that correspond to the signals must be checked in manuals, which takes time. The programming time can be reduced by using module labels because users only need to select labels from the list.

When device names are used

When module labels are used



Check and describe installation position and buffer memory address.

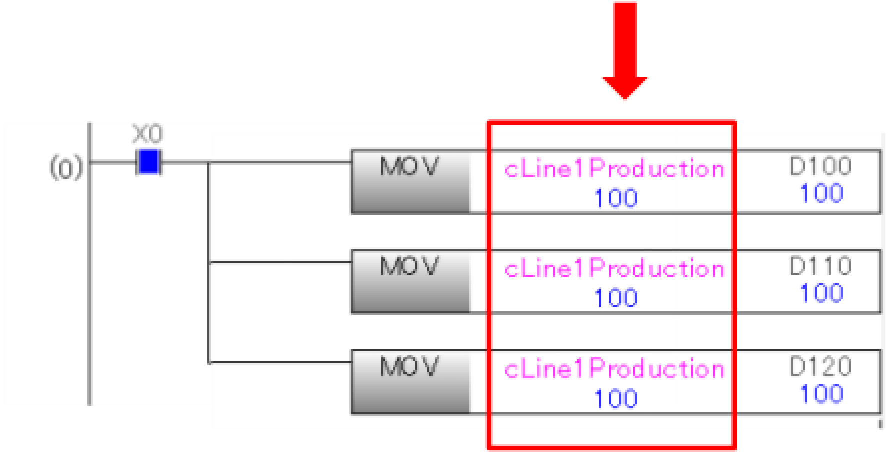
Simply select a module label registered as the program element.

Module Label
- 3E00:R04CPU
- 0090:R60DA4
  - R60DA_1
    - R60DA_1 — Version: 00B
    - uIO — I/O No.
    - Input/Output Signals
    - Buffer memory
      - uLatestErrorCode — L
      - uLatestAddressOfErrorHistory — Latest address of error histor
      - uLatestAlarmCode — Latest alarm code
      - uLatestAddressOfAlarmHistory — Latest address of alarm histo
      - uWarningOutputUpperFlag — Warning output upper flag
      - uWarningOutputLowerFlag — Warning output lower flag
      - uDisconnectionDetectionFlag — Disconnection detection flag
      - uSynchronousStatusMonitor — Synchronous state monitor
    - Direct

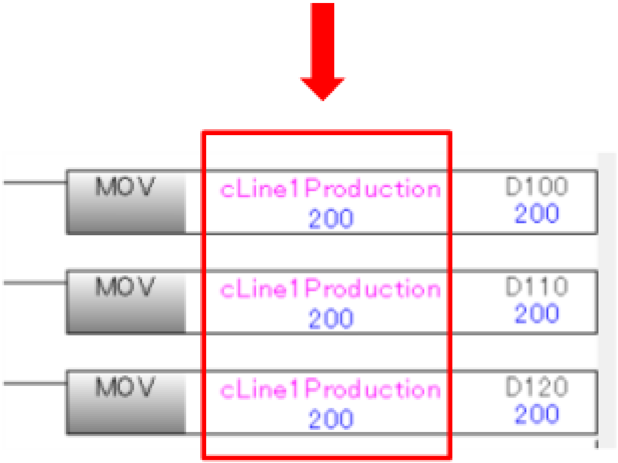
1.3.5 Assigning constants to labels

Constants can be assigned to labels.
When constants are assigned to labels, the values can be changed without modifying the program.
The same constant used for multiple labels can be changed collectively.

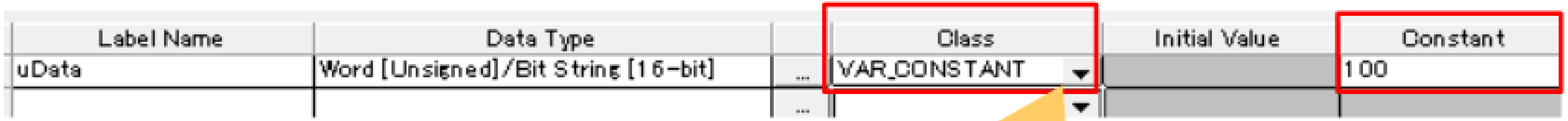
Assign the constant 100 to the label "cLine1Production".



Assign the constant 200.



To assign a constant to a label, change the class that specifies the application of the label on the window for setting labels.
For local labels, select "VAR_CONSTANT".

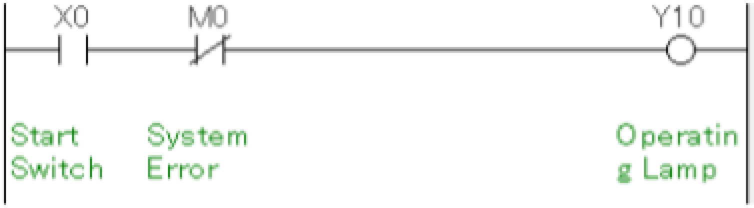


Label Name	Data Type		Class	Initial Value	Constant
uData	Word [Unsigned]/Bit String [16-bit]	...	VAR_CONSTANT		100
		...			

Specify the application of the label.

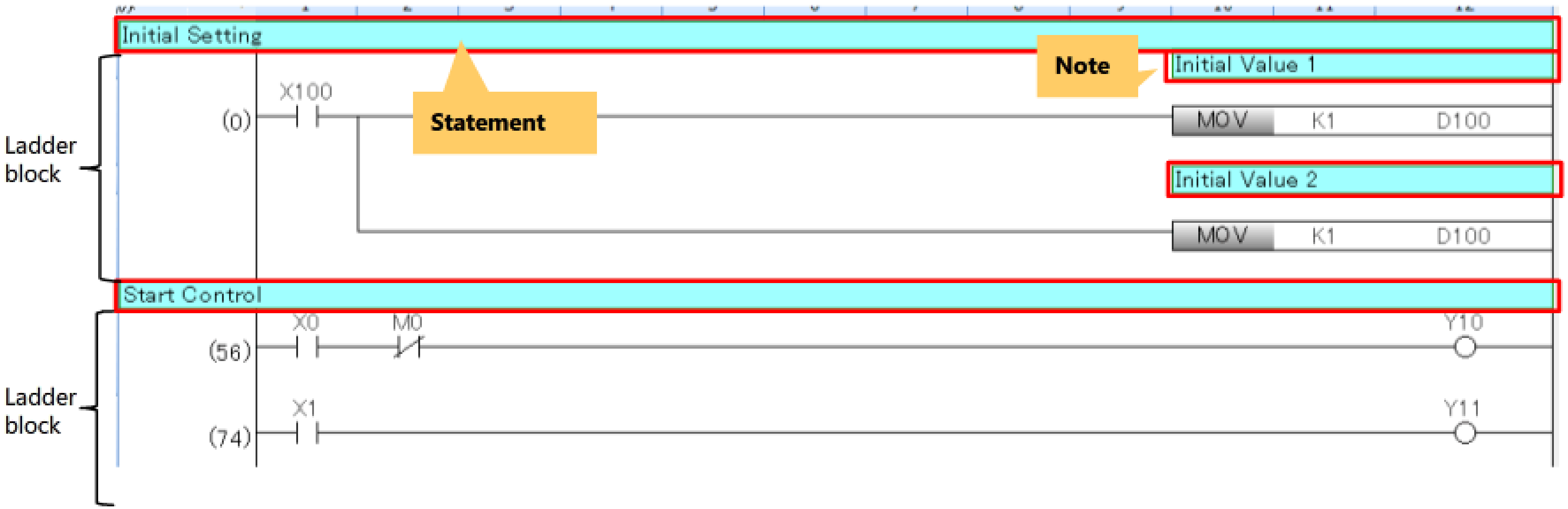
1.4 Improving program readability

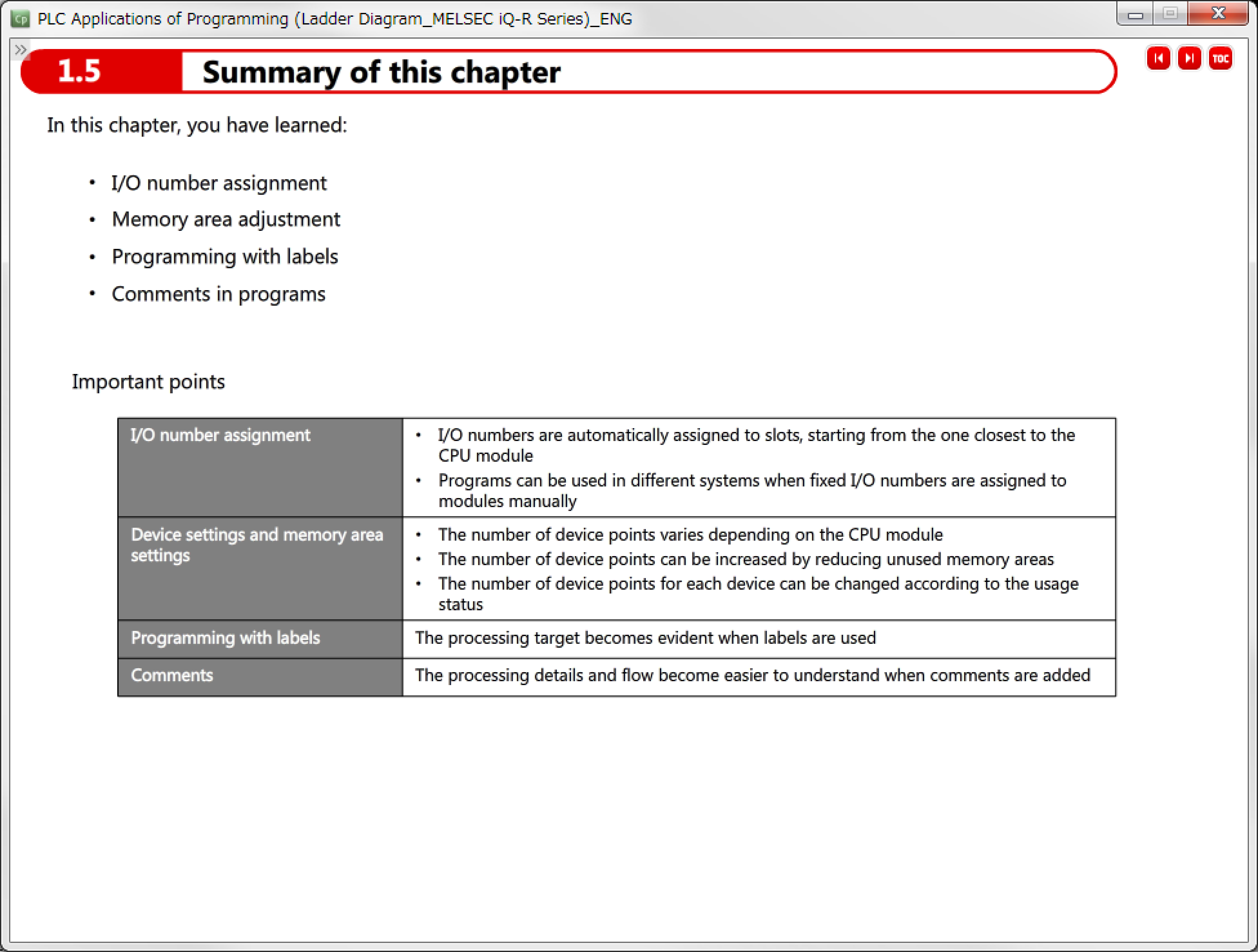
Comments, such as processing details and device names, can be added to a program.
Comments help clarify how the program runs.



Comment type can be selected according to the element or range of the program.
In the above program example, "device/label comments" have been added to clarify the applications of the device or the label and the types of connected I/O devices.

In addition, comment type includes a "statement" that is added to the ladder block to help clarify the processing flow and "notes" that help clarify the details of coils and application instructions.





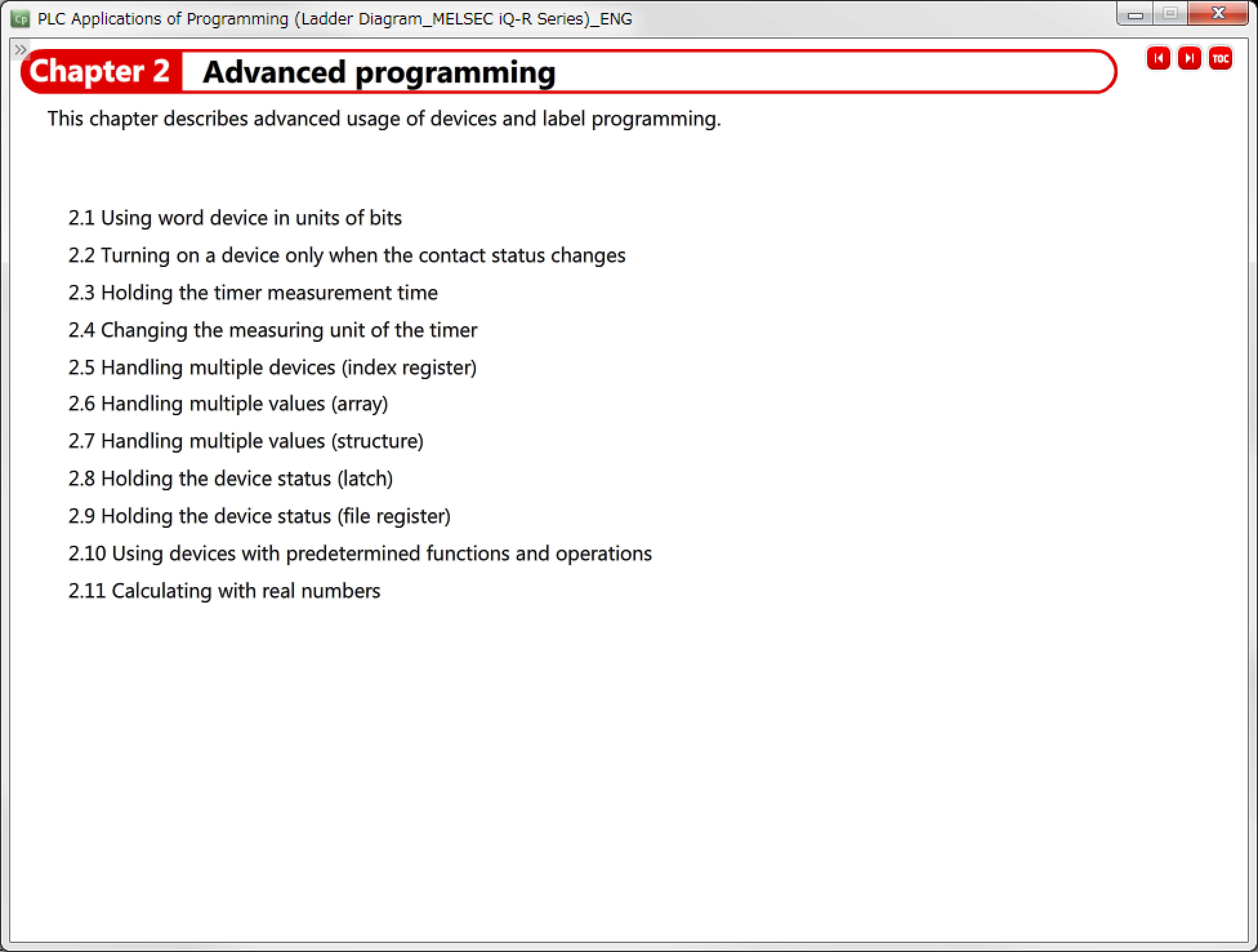
1.5 Summary of this chapter

In this chapter, you have learned:

- I/O number assignment
- Memory area adjustment
- Programming with labels
- Comments in programs

Important points

I/O number assignment	• I/O numbers are automatically assigned to slots, starting from the one closest to the CPU module • Programs can be used in different systems when fixed I/O numbers are assigned to modules manually
Device settings and memory area settings	• The number of device points varies depending on the CPU module • The number of device points can be increased by reducing unused memory areas • The number of device points for each device can be changed according to the usage status
Programming with labels	The processing target becomes evident when labels are used
Comments	The processing details and flow become easier to understand when comments are added



PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC

Chapter 2 Advanced programming

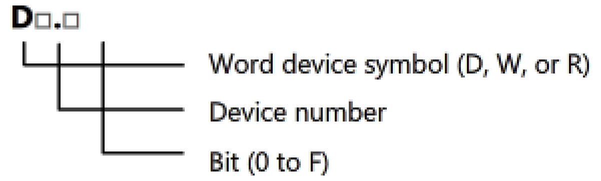
This chapter describes advanced usage of devices and label programming.

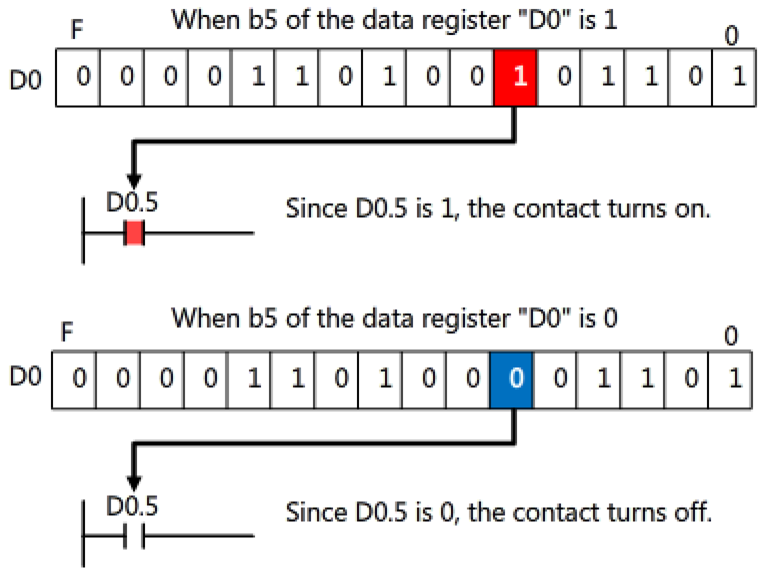
2.1 Using word device in units of bits

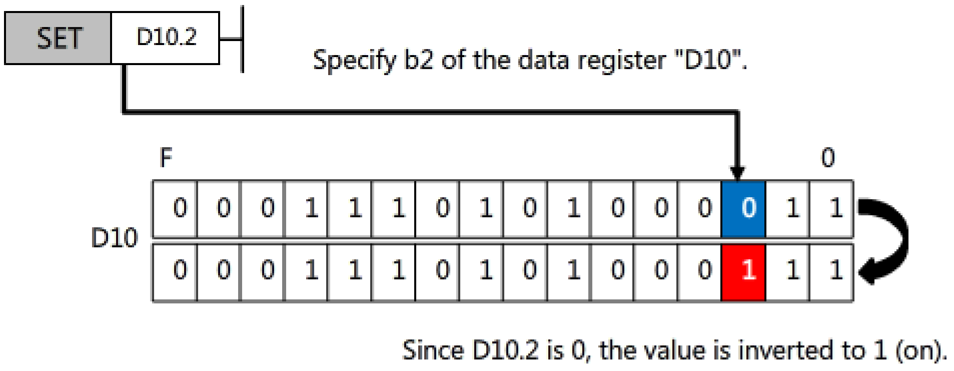
Word devices, such as data register, are normally used in units of words, but they can also be used in units of bits. The unit of bits is used to specify a particular bit in the data register (D).

Example) Data register (D)	0	0	1	0	0	1	1	0	1	1	0	0	0	1	0	1

Bit specification format




When b5 of the data register "D0" is 1
F
0
D0
0 0 0 0 1 1 0 1 0 0 1 0 1 1 0 1
D0.5
Since D0.5 is 1, the contact turns on.
When b5 of the data register "D0" is 0
F
0
D0
0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1
D0.5
Since D0.5 is 0, the contact turns off.

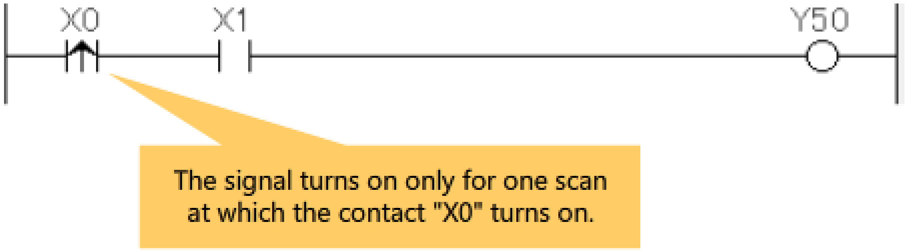

SET
D10.2
Specify b2 of the data register "D10".
F
0
D10
0 0 0 1 1 1 0 1 0 1 0 0 0 0 1 1
0 0 0 1 1 1 0 1 0 1 0 0 0 1 1 1
Since D10.2 is 0, the value is inverted to 1 (on).

To use labels, make description as "uData.2" and "uData.5".

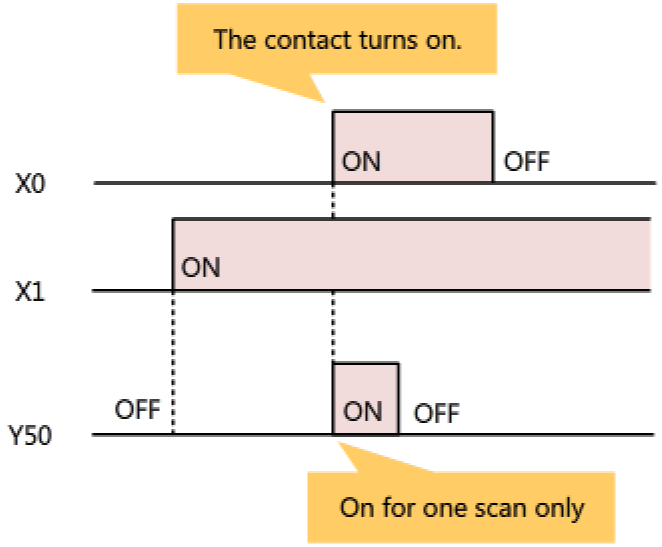
2.2 Turning on a device only when the contact status changes

A signal that turns on only for one scan on the rising edge or on the falling edge of the contact can be specified.
This function is useful for controlling rise and fall by the input condition.

Rising edge specification of contact



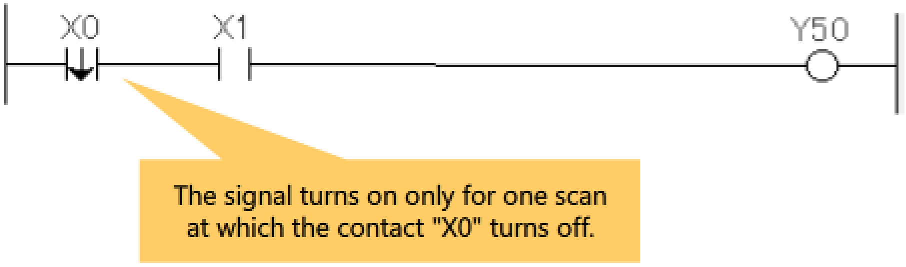
The signal turns on only for one scan at which the contact "X0" turns on.



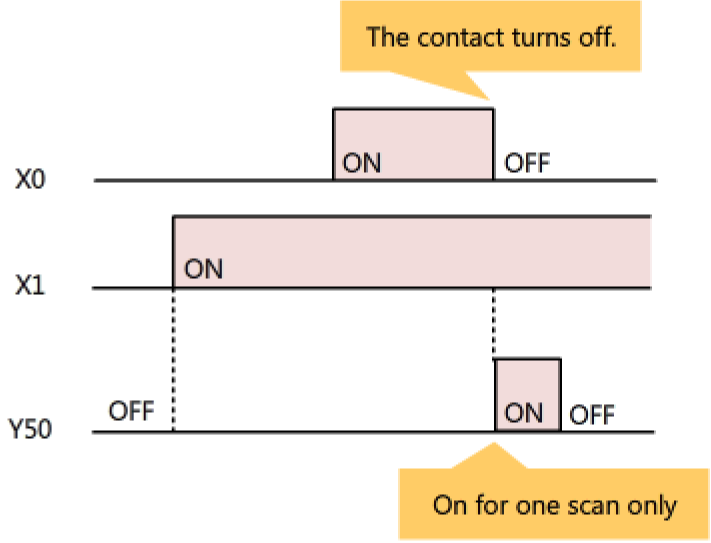
The contact turns on.

On for one scan only

Falling edge specification of contact

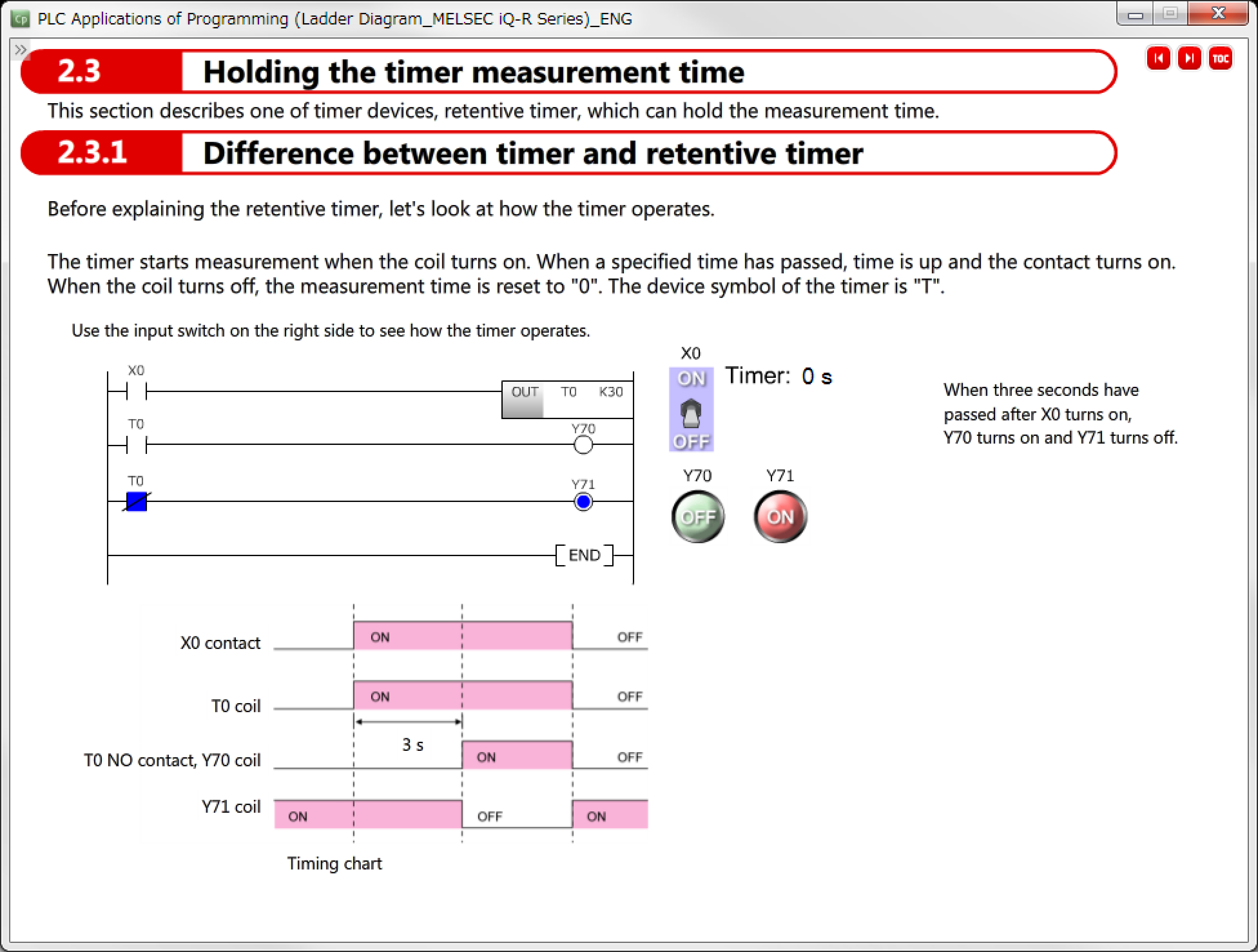


The signal turns on only for one scan at which the contact "X0" turns off.



The contact turns off.

On for one scan only



2.3 Holding the timer measurement time

This section describes one of timer devices, retentive timer, which can hold the measurement time.

2.3.1 Difference between timer and retentive timer

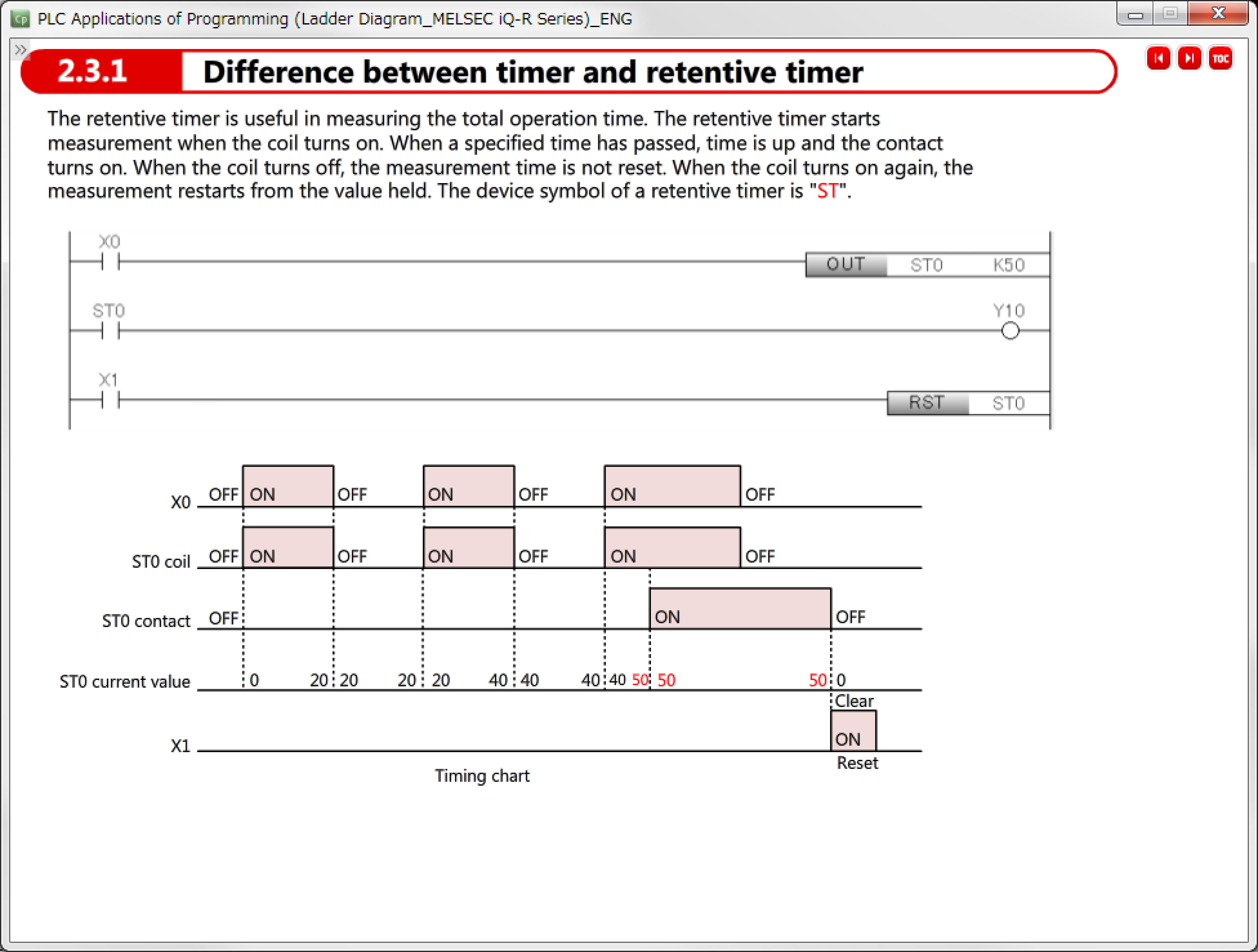
Before explaining the retentive timer, let's look at how the timer operates.

The timer starts measurement when the coil turns on. When a specified time has passed, time is up and the contact turns on. When the coil turns off, the measurement time is reset to "0". The device symbol of the timer is "T".

Use the input switch on the right side to see how the timer operates.

When three seconds have passed after X0 turns on, Y70 turns on and Y71 turns off.

Timing chart



2.3.1 Difference between timer and retentive timer

The retentive timer is useful in measuring the total operation time. The retentive timer starts measurement when the coil turns on. When a specified time has passed, time is up and the contact turns on. When the coil turns off, the measurement time is not reset. When the coil turns on again, the measurement restarts from the value held. The device symbol of a retentive timer is "ST".

Timing chart

2.3.2 Program example of retentive timer

Let's look at how the retentive timer operates by simulating a running machine using the input switches (X0 to X2).
*The retentive timer (ST0) is set in increments of 100 ms.

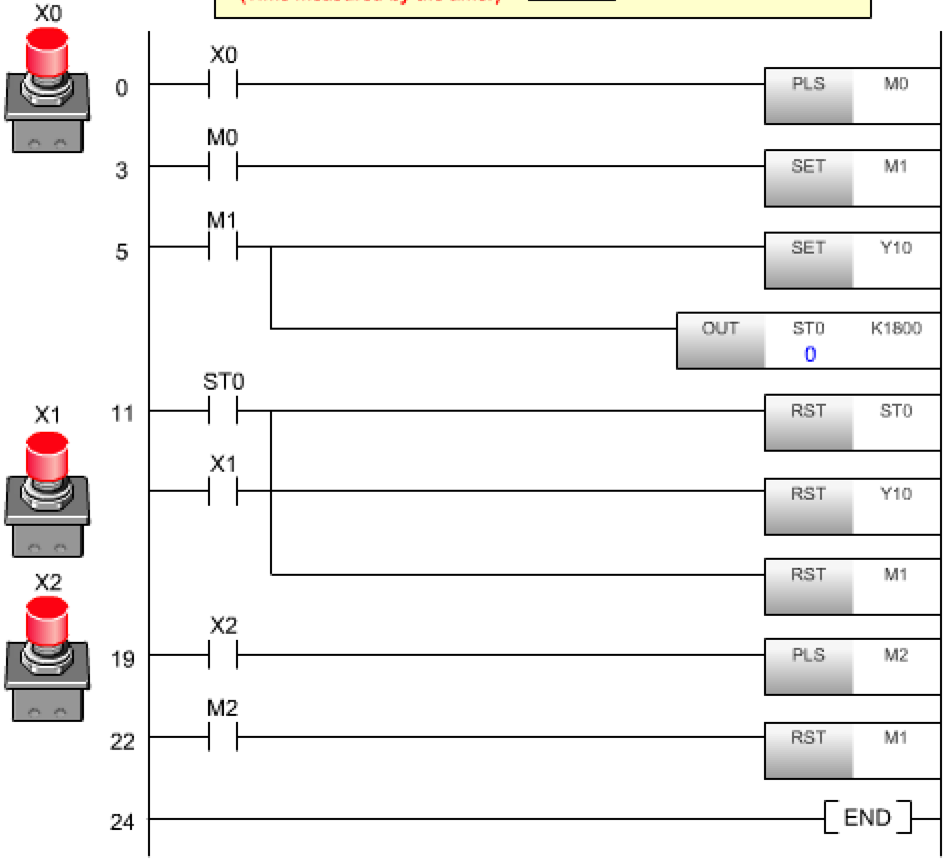
1. When X0 turns on, running starts.
2. When X2 turns on, running pauses and the current value is held.
3. When X0 turns on again, running starts again.
4. When X1 turns on, running ends and the current value is reset.

X0 to X2: Input switches

Y10: Start signal

Timer ST0 is set to K1800 = 180,000 ms (3 min.)/100 ms

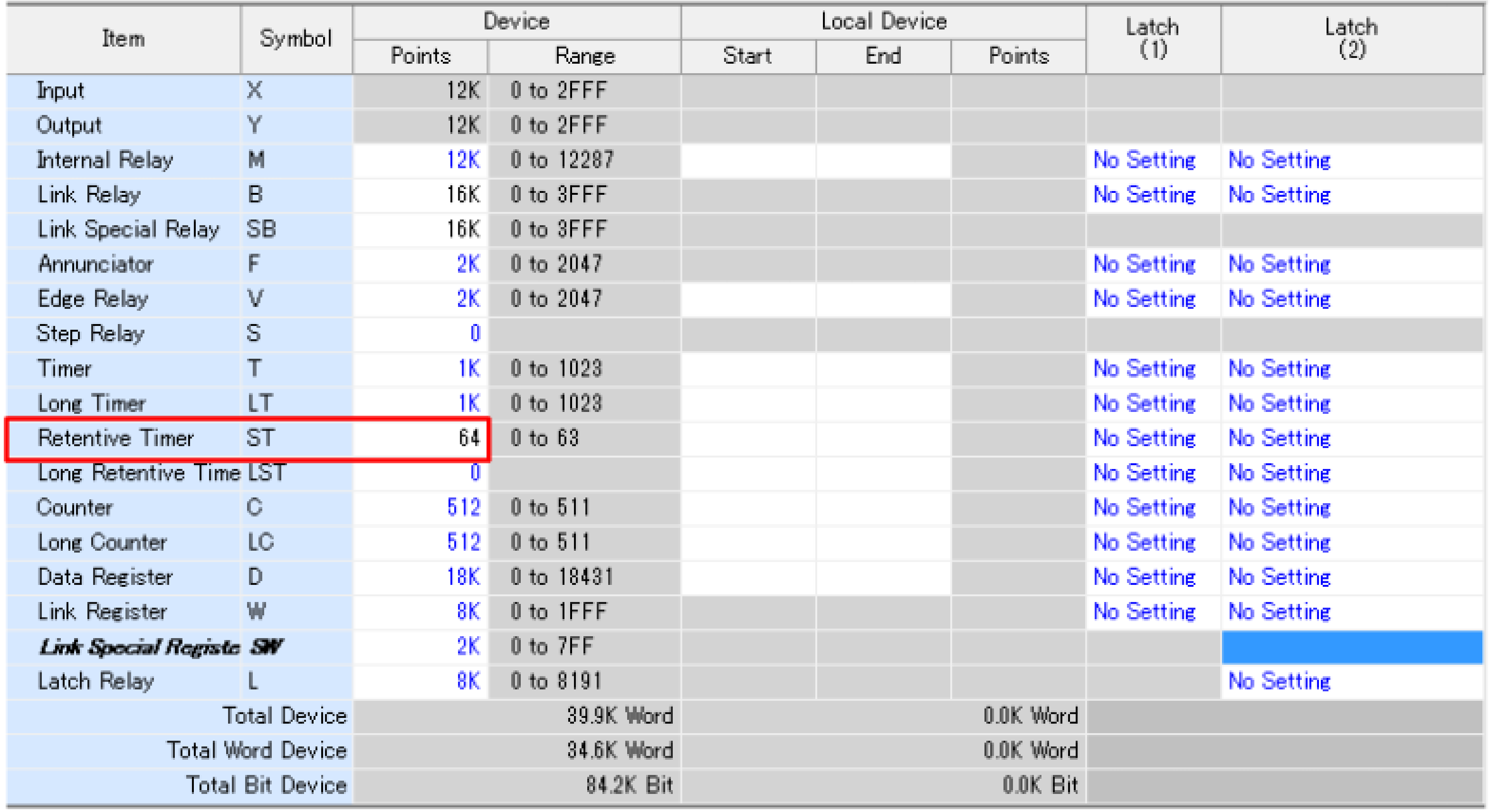
Running time
(Time measured by the timer) 0 sec.



2.3.3 Setting for the retentive timer

The number of points used by the retentive timer is "0" by default.
Before using the retentive timer, set the number of points in "Device Setting" of CPU parameter using GX Works3.

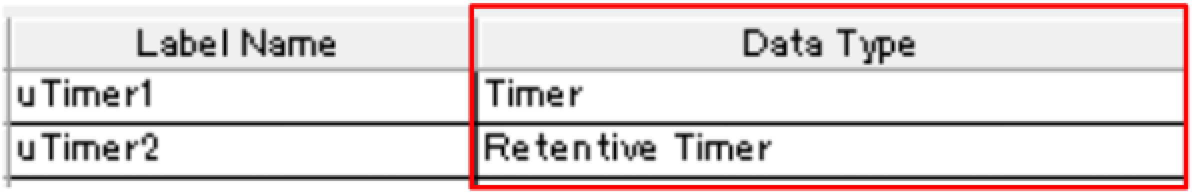
In the following example, 64 points (ST0 to ST63) are set for the retentive timer.



Item	Symbol	Device		Local Device			Latch (1)	Latch (2)
		Points	Range	Start	End	Points		
Input	X	12K	0 to 2FFF					
Output	Y	12K	0 to 2FFF					
Internal Relay	M	12K	0 to 12287				No Setting	No Setting
Link Relay	B	16K	0 to 3FFF				No Setting	No Setting
Link Special Relay	SB	16K	0 to 3FFF					
Annunciator	F	2K	0 to 2047				No Setting	No Setting
Edge Relay	V	2K	0 to 2047				No Setting	No Setting
Step Relay	S	0						
Timer	T	1K	0 to 1023				No Setting	No Setting
Long Timer	LT	1K	0 to 1023				No Setting	No Setting
Retentive Timer	ST	64	0 to 63				No Setting	No Setting
Long Retentive Time	LST	0					No Setting	No Setting
Counter	C	512	0 to 511				No Setting	No Setting
Long Counter	LC	512	0 to 511				No Setting	No Setting
Data Register	D	18K	0 to 18431				No Setting	No Setting
Link Register	W	8K	0 to 1FFF				No Setting	No Setting
Link Special Registe	***SW***	2K	0 to 7FF					
Latch Relay	L	8K	0 to 8191					No Setting
Total Device		39.9K Word		0.0K Word				
Total Word Device		34.6K Word		0.0K Word				
Total Bit Device		84.2K Bit		0.0K Bit				

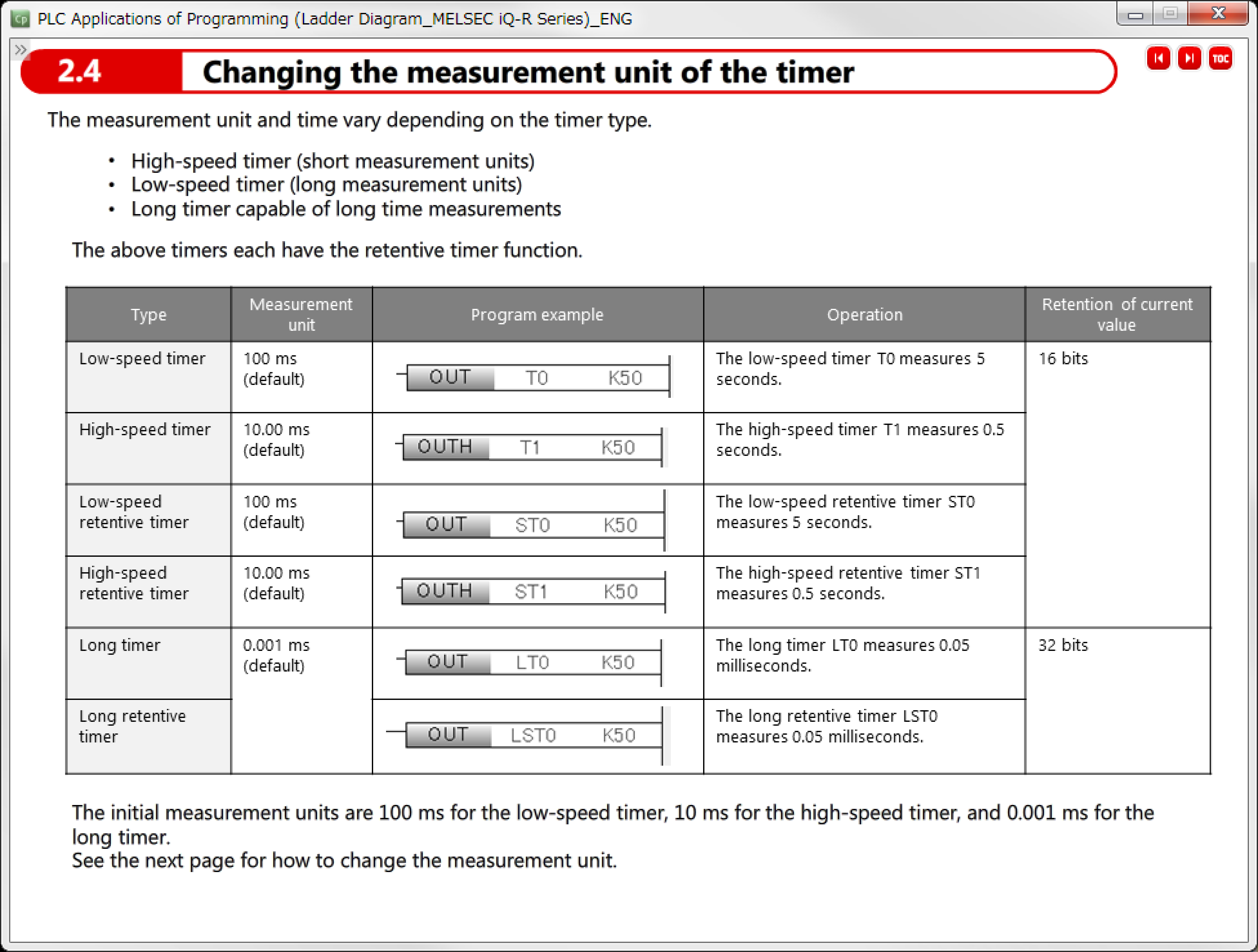
2.3.4 Using a label to specify a timer

Set "Timer" to the data type when a label uses the timer device.



Label Name	Data Type
uTimer1	Timer
uTimer2	Retentive Timer

The device setting is necessary to use the retentive timer as described in the previous section.
However, it is not necessary for labels.



2.4 Changing the measurement unit of the timer

The measurement unit and time vary depending on the timer type.

- High-speed timer (short measurement units)
- Low-speed timer (long measurement units)
- Long timer capable of long time measurements

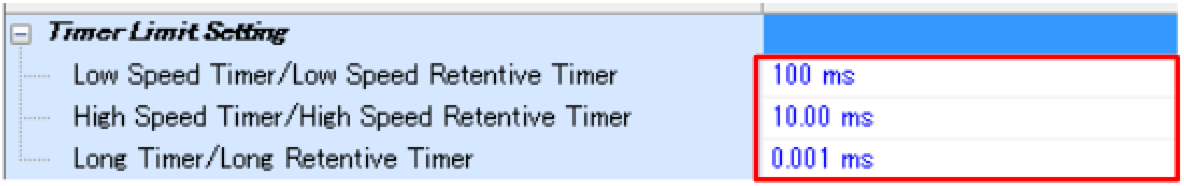
The above timers each have the retentive timer function.

Type	Measurement unit	Program example	Operation	Retention of current value
Low-speed timer	100 ms (default)	OUT T0 K50	The low-speed timer T0 measures 5 seconds.	16 bits
High-speed timer	10.00 ms (default)	OUTH T1 K50	The high-speed timer T1 measures 0.5 seconds.	
Low-speed retentive timer	100 ms (default)	OUT ST0 K50	The low-speed retentive timer ST0 measures 5 seconds.	
High-speed retentive timer	10.00 ms (default)	OUTH ST1 K50	The high-speed retentive timer ST1 measures 0.5 seconds.	
Long timer	0.001 ms (default)	OUT LT0 K50	The long timer LT0 measures 0.05 milliseconds.	32 bits
Long retentive timer		OUT LST0 K50	The long retentive timer LST0 measures 0.05 milliseconds.	

The initial measurement units are 100 ms for the low-speed timer, 10 ms for the high-speed timer, and 0.001 ms for the long timer.
See the next page for how to change the measurement unit.

2.4 Changing the measurement unit of the timer

The measurement unit of the timer can be changed in the "Timer Limit Setting" of CPU parameter.



Timer Limit Setting	
Low Speed Timer/Low Speed Retentive Timer	100 ms
High Speed Timer/High Speed Retentive Timer	10.00 ms
Long Timer/Long Retentive Timer	0.001 ms

2.5 Handling multiple devices (index register)

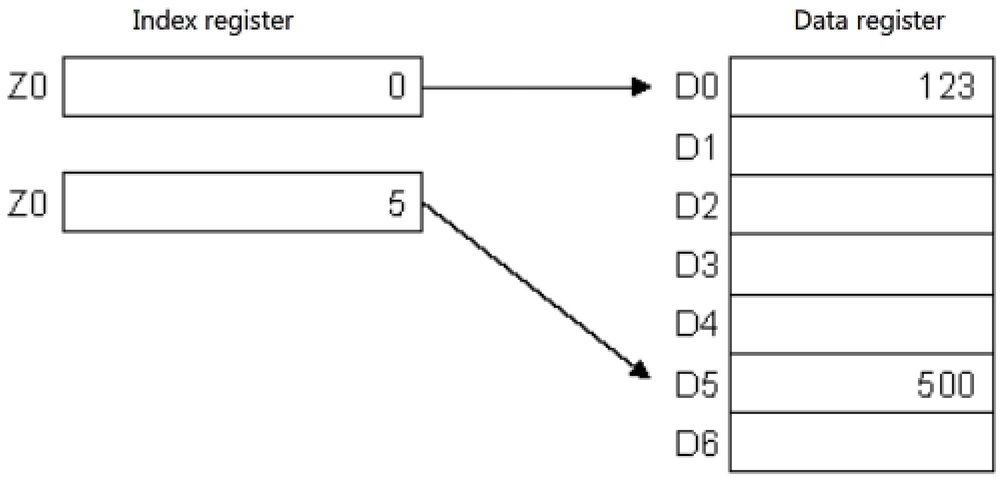
The index register (Z) is used in combination with another device to indirectly specify (modify) the device number of the control-target device. The index register is useful for simplifying programs because it can describe multiple devices in a batch.

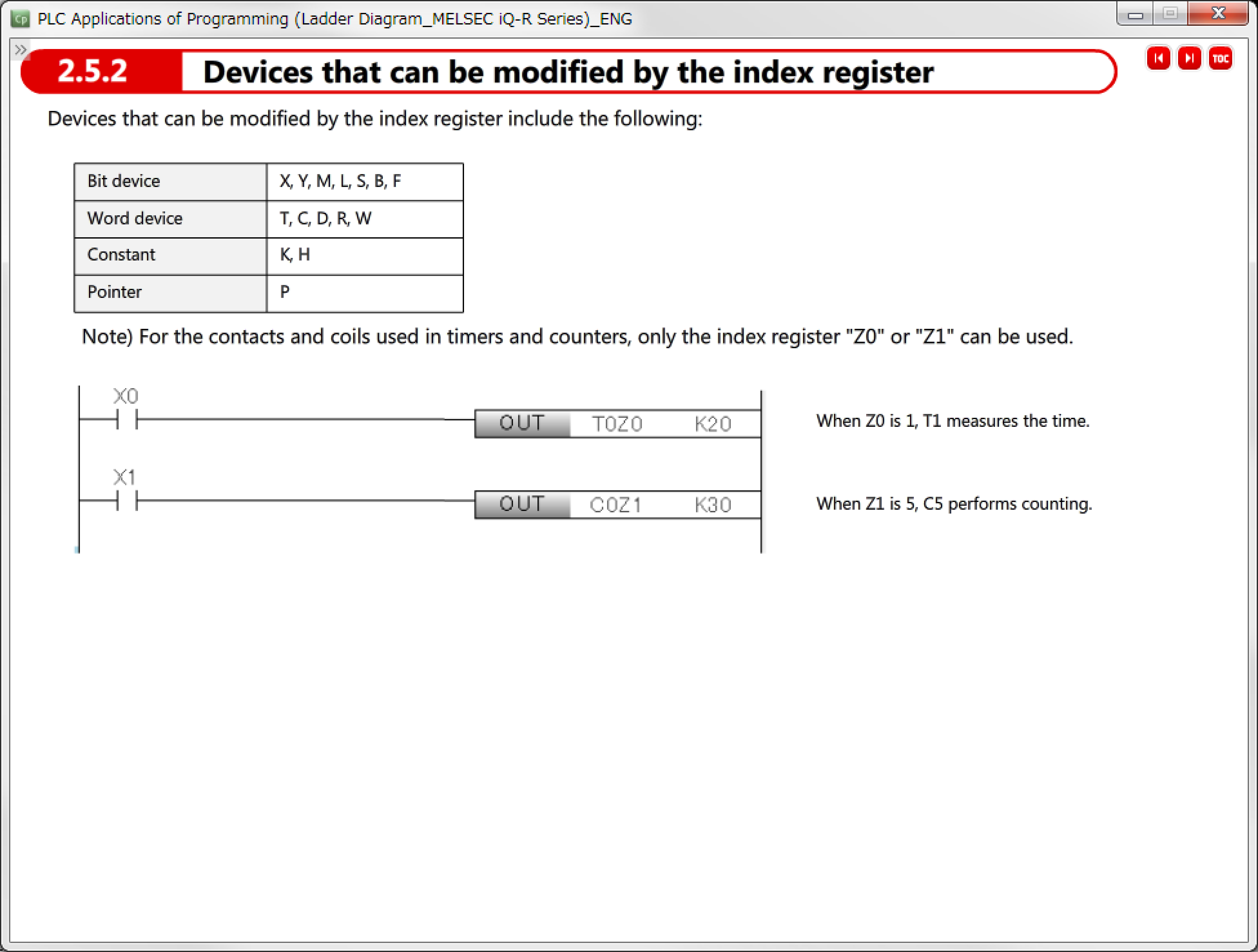
- The index register is described after a device symbol and a device number
- Actual control target device number = Device symbol (device number + index register)
- The number of device points for the index register is 20 points (Z0 to Z19) by default

2.5.1 Application example of the index register

When a device is described as "D0Z0", it means D(0+Z0).

Example) When Z0 is 0, the device number is D0.
When Z0 is 5, the device number is D5.





2.5.2 Devices that can be modified by the index register

Devices that can be modified by the index register include the following:

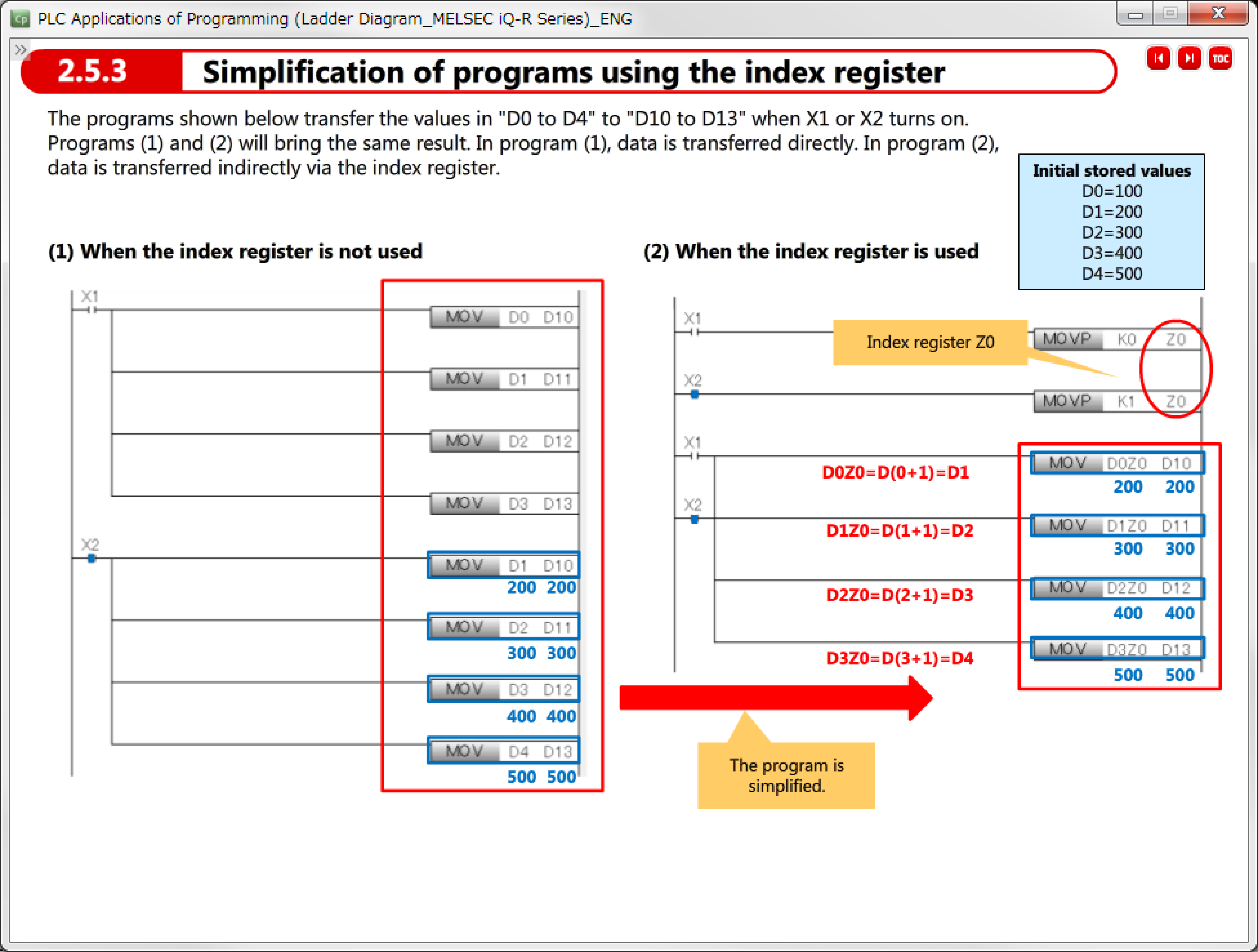
Bit device	X, Y, M, L, S, B, F
Word device	T, C, D, R, W
Constant	K, H
Pointer	P

Note) For the contacts and coils used in timers and counters, only the index register "Z0" or "Z1" can be used.

```
 X0
-| |------------------[OUT  T0Z0  K20]
 X1
-| |------------------[OUT  C0Z1  K30]
```

When Z0 is 1, T1 measures the time.

When Z1 is 5, C5 performs counting.



2.5.3 Simplification of programs using the index register

The programs shown below transfer the values in "D0 to D4" to "D10 to D13" when X1 or X2 turns on. Programs (1) and (2) will bring the same result. In program (1), data is transferred directly. In program (2), data is transferred indirectly via the index register.

Initial stored values
D0=100
D1=200
D2=300
D3=400
D4=500

(1) When the index register is not used

(2) When the index register is used

Index register Z0

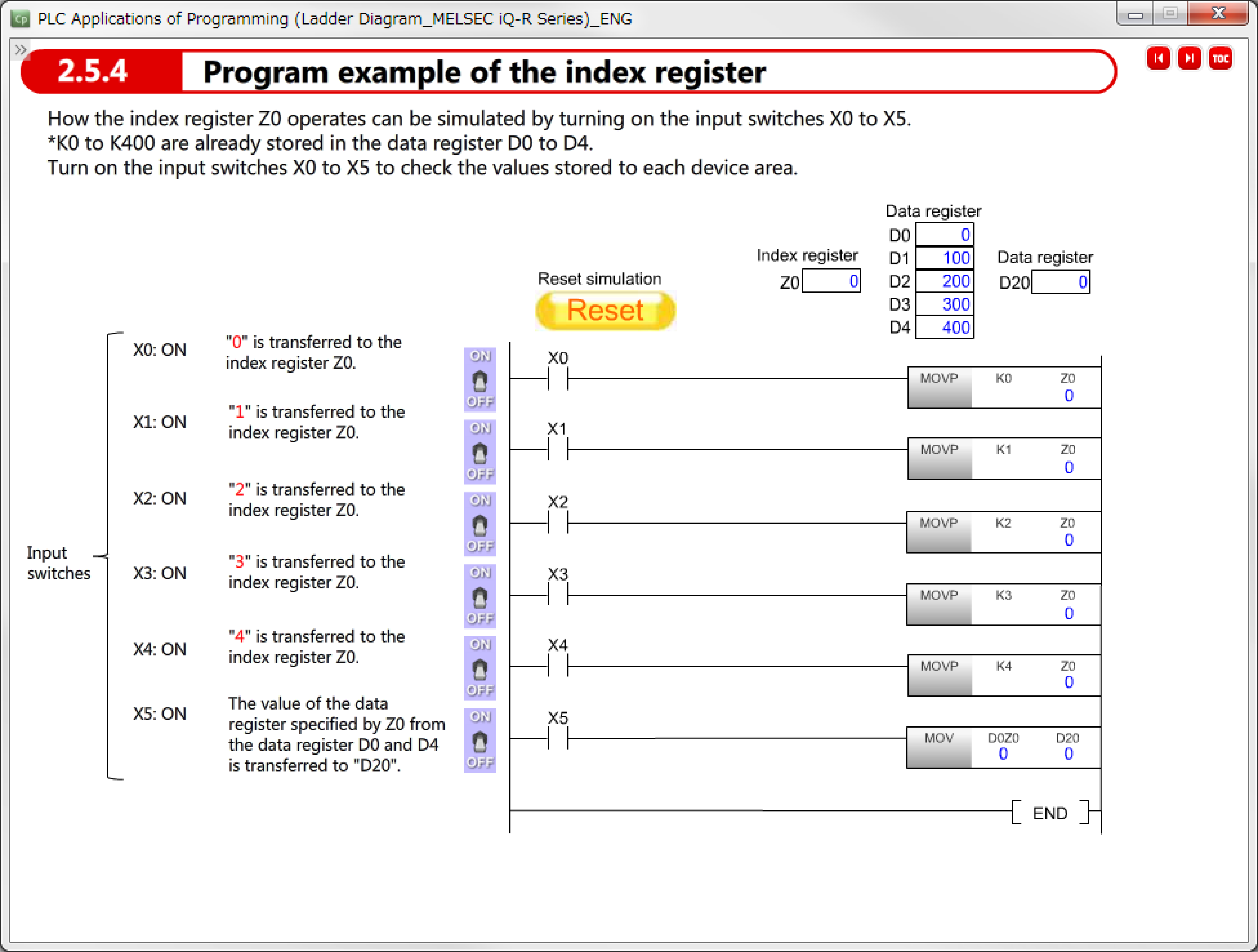
D0Z0=D(0+1)=D1

D1Z0=D(1+1)=D2

D2Z0=D(2+1)=D3

D3Z0=D(3+1)=D4

The program is simplified.



2.5.4 Program example of the index register

How the index register Z0 operates can be simulated by turning on the input switches X0 to X5.
*K0 to K400 are already stored in the data register D0 to D4.
Turn on the input switches X0 to X5 to check the values stored to each device area.

Input switches

X0: ON "0" is transferred to the index register Z0.

X1: ON "1" is transferred to the index register Z0.

X2: ON "2" is transferred to the index register Z0.

X3: ON "3" is transferred to the index register Z0.

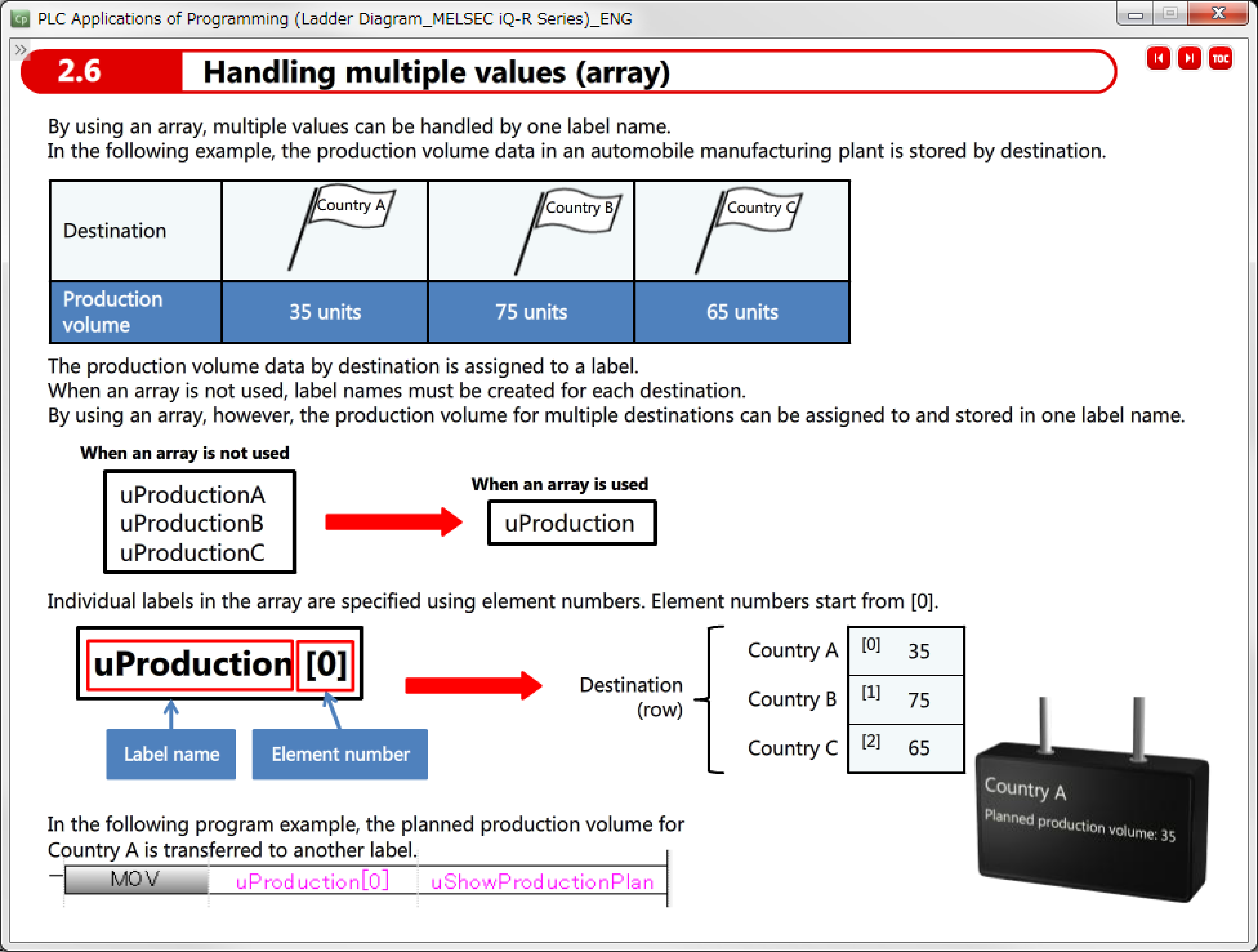
X4: ON "4" is transferred to the index register Z0.

X5: ON The value of the data register specified by Z0 from the data register D0 and D4 is transferred to "D20".

Data register	
D0	0
D1	100
D2	200
D3	300
D4	400

Index register Z0: 0

Data register D20: 0



PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC

2.6 Handling multiple values (array)

By using an array, multiple values can be handled by one label name.
In the following example, the production volume data in an automobile manufacturing plant is stored by destination.

Destination	Country A	Country B	Country C
Production volume	35 units	75 units	65 units

The production volume data by destination is assigned to a label.
When an array is not used, label names must be created for each destination.
By using an array, however, the production volume for multiple destinations can be assigned to and stored in one label name.

When an array is not used

uProductionA
uProductionB
uProductionC

When an array is used

uProduction

Individual labels in the array are specified using element numbers. Element numbers start from [0].

uProduction [0]

Label name

Element number

Destination (row)

Destination (row)	Element	Value
Country A	[0]	35
Country B	[1]	75
Country C	[2]	65

In the following program example, the planned production volume for Country A is transferred to another label.

```
MOV    uProduction[0]    uShowProductionPlan
```

2.7 Handling multiple values (structure)

By using a structure, multiple related labels can be handled by one label name.
In the following example, the status of an automobile production line is displayed on Andon (display board).

The following table lists the label names, values, and data types corresponding to the displayed items.

Item	Label name	Value	Data type of label
Model	sModel	'ST TRUCK'	String type
Operation status	bStatus	'in production'	Bit type
Today's target production volume	uPlanQty	'100' units	Integer type (word, unsigned)
Current production volume	uActualQty	'88' units	Integer type (word, unsigned)

Model
ST TRUCK
Status
in production
Target
100
Current
88

If a structure is not used for a factory having multiple production lines, the label name must be changed for each line.
The following are examples of label names with production line names added.

First production line

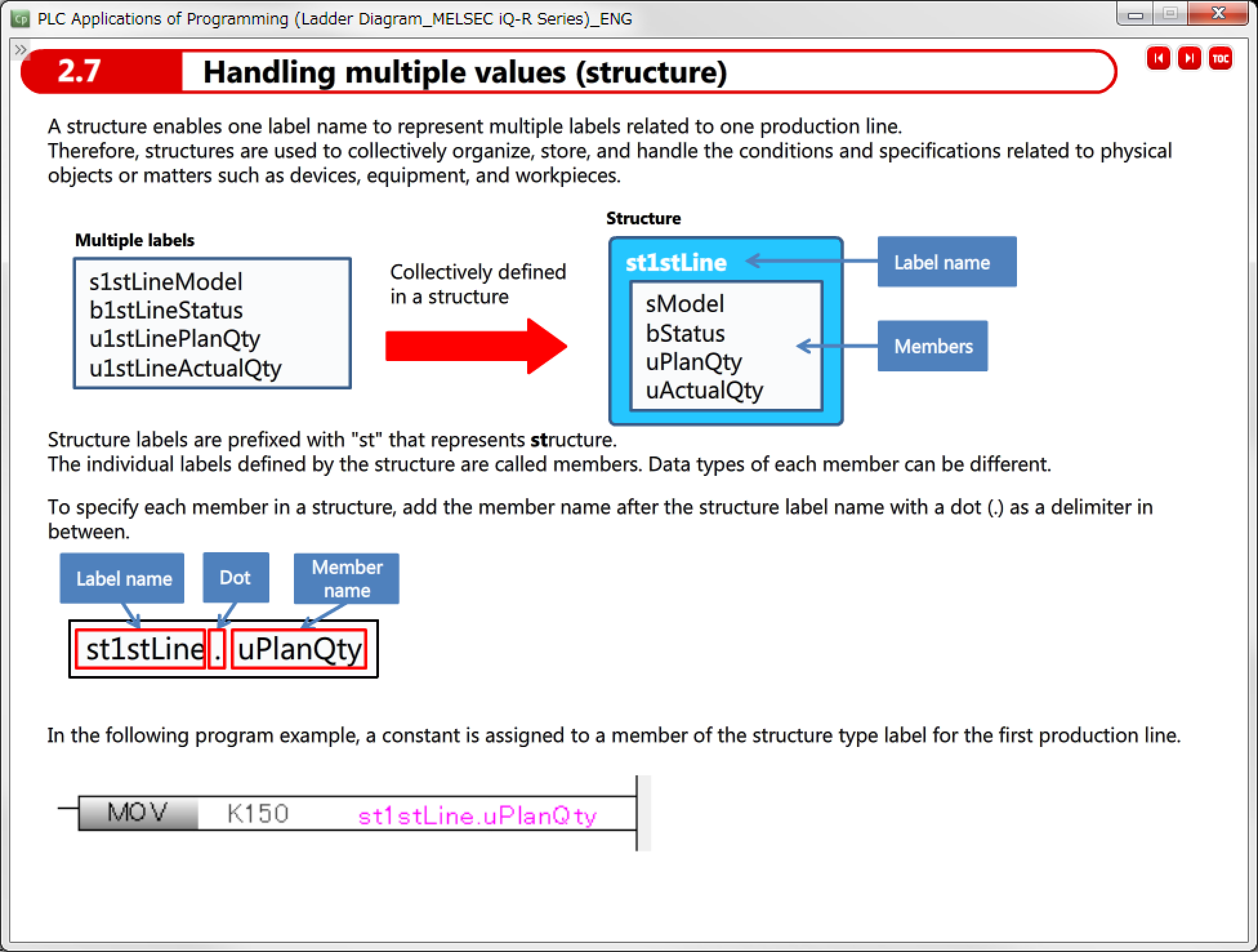
s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

Second production line

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

● ● ●

As the number of production lines increases, the number of labels to be handled increases. As a result, the program becomes longer and more difficult to read.



2.7 Handling multiple values (structure)

A structure enables one label name to represent multiple labels related to one production line.
Therefore, structures are used to collectively organize, store, and handle the conditions and specifications related to physical objects or matters such as devices, equipment, and workpieces.

Structure labels are prefixed with "st" that represents **st**ructure.
The individual labels defined by the structure are called members. Data types of each member can be different.

To specify each member in a structure, add the member name after the structure label name with a dot (.) as a delimiter in between.

In the following program example, a constant is assigned to a member of the structure type label for the first production line.

```
─┤ MOV │ K150 │ st1stLine.uPlanQty ├─
```

2.8 Holding the device status (latch)

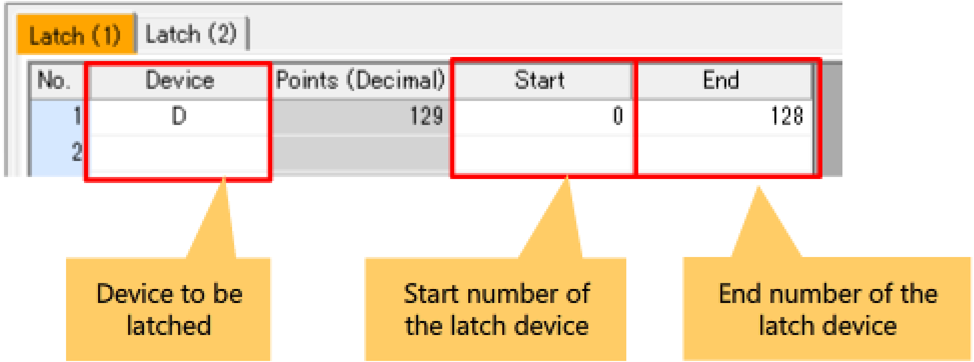
This section describes the latch function, which holds device values when the CPU module stops operation.
For example, even when a power failure exceeding the allowable momentary power failure time occurs, the programmable controller can restart sequence control using the data held at the operation stop.

If the latch function is not used, device values are cleared and reset to the initial values (off for bit devices and "0" for word devices) in the following events:

- Power-off
- Reset by the RUN/STOP/RESET switch
- Power failure exceeding the allowable momentary power failure time

2.8.1 Setting latch on devices

Set the latch range in the "Device Setting" window of CPU parameter.
The following is a latch setting example of the data register, D0 to D128.


Latch (1)
Latch (2)
Device to be latched
Start number of the latch device
End number of the latch device

No.	Device	Points (Decimal)	Start	End
1	D	129	0	128
2				

2.8.2 Latch types and clear methods

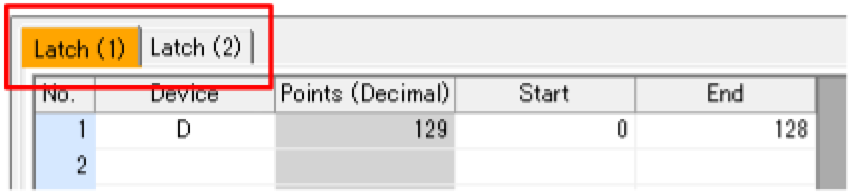
There are two types of latch (latch (1) and latch (2)) according to the clear methods.

- Latch (1)

  Latched data can be cleared by using the CPU memory operation function* of GX Works3.
  Use latch 1 when latched data needs to be cleared at the installation site.

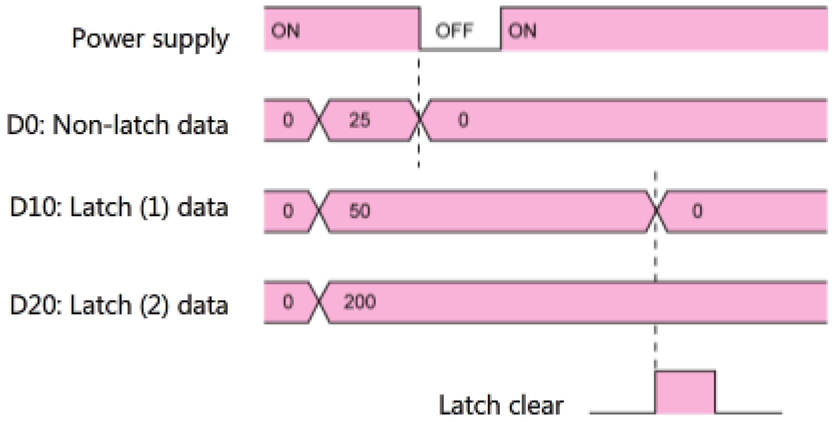
- Latch (2)

  Latched data can be cleared by using an instruction in the program.
  Use latch 2 when latched data is not cleared at the installation site.



No.	Device	Points (Decimal)	Start	End
1	D	129	0	128
2				

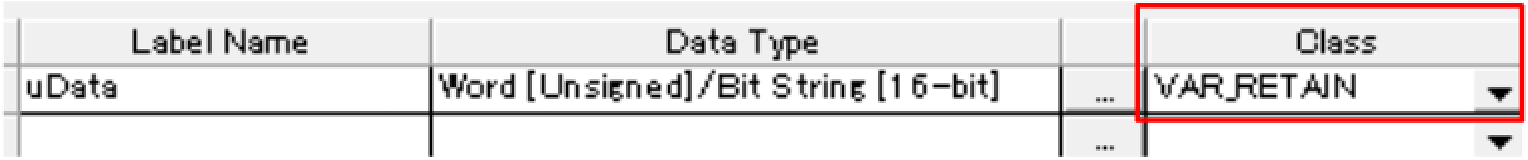
The following is the timing chart of latch clear.



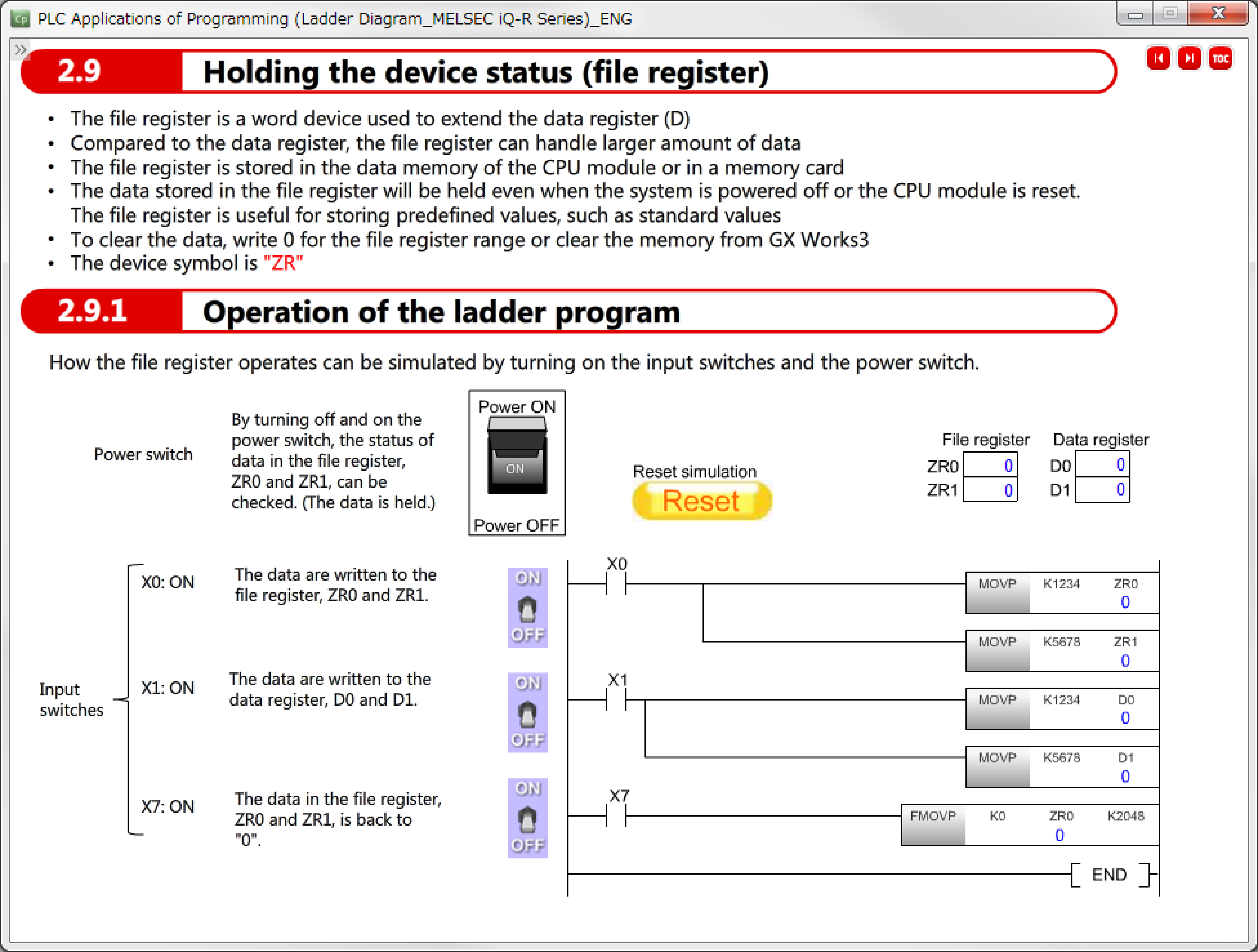
*Execute the function by selecting [Online] → [CPU Memory Operation] from the menu of GX Works3.

2.8.3 Setting latch on labels

To set a latch on a label, select a class name containing "RETAIN" on the window for setting labels.
For local labels, select "VAR_RETAIN".



Label Name	Data Type		Class
uData	Word [Unsigned]/Bit String [16-bit]	...	VAR_RETAIN
		...	



PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG

2.9 Holding the device status (file register)

- The file register is a word device used to extend the data register (D)
- Compared to the data register, the file register can handle larger amount of data
- The file register is stored in the data memory of the CPU module or in a memory card
- The data stored in the file register will be held even when the system is powered off or the CPU module is reset. The file register is useful for storing predefined values, such as standard values
- To clear the data, write 0 for the file register range or clear the memory from GX Works3
- The device symbol is "ZR"

2.9.1 Operation of the ladder program

How the file register operates can be simulated by turning on the input switches and the power switch.

Power switch

By turning off and on the power switch, the status of data in the file register, ZR0 and ZR1, can be checked. (The data is held.)

Reset simulation

File register		Data register	
ZR0	0	D0	0
ZR1	0	D1	0

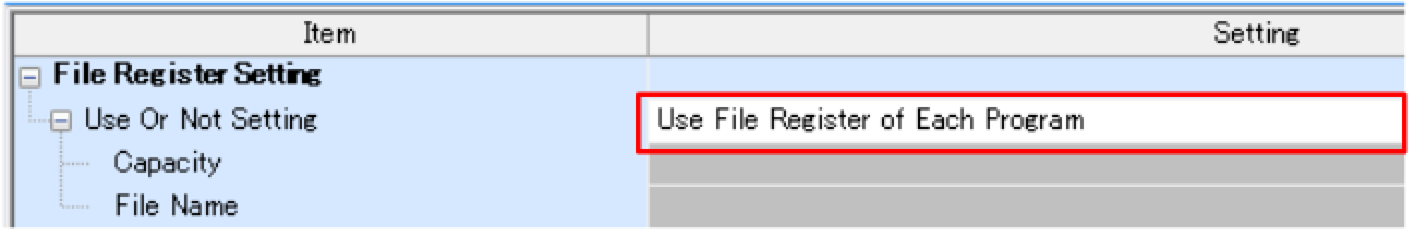
Input switches

- X0: ON — The data are written to the file register, ZR0 and ZR1.
- X1: ON — The data are written to the data register, D0 and D1.
- X7: ON — The data in the file register, ZR0 and ZR1, is back to "0".

```
X0
─┤├─┬──────────── MOVP   K1234  ZR0
    │                           0
    └──────────── MOVP   K5678  ZR1
                                0
X1
─┤├─┬──────────── MOVP   K1234  D0
    │                           0
    └──────────── MOVP   K5678  D1
                                0
X7
─┤├────────────── FMOVP  K0     ZR0   K2048
                                0
───────────────────────────────── [ END ]
```

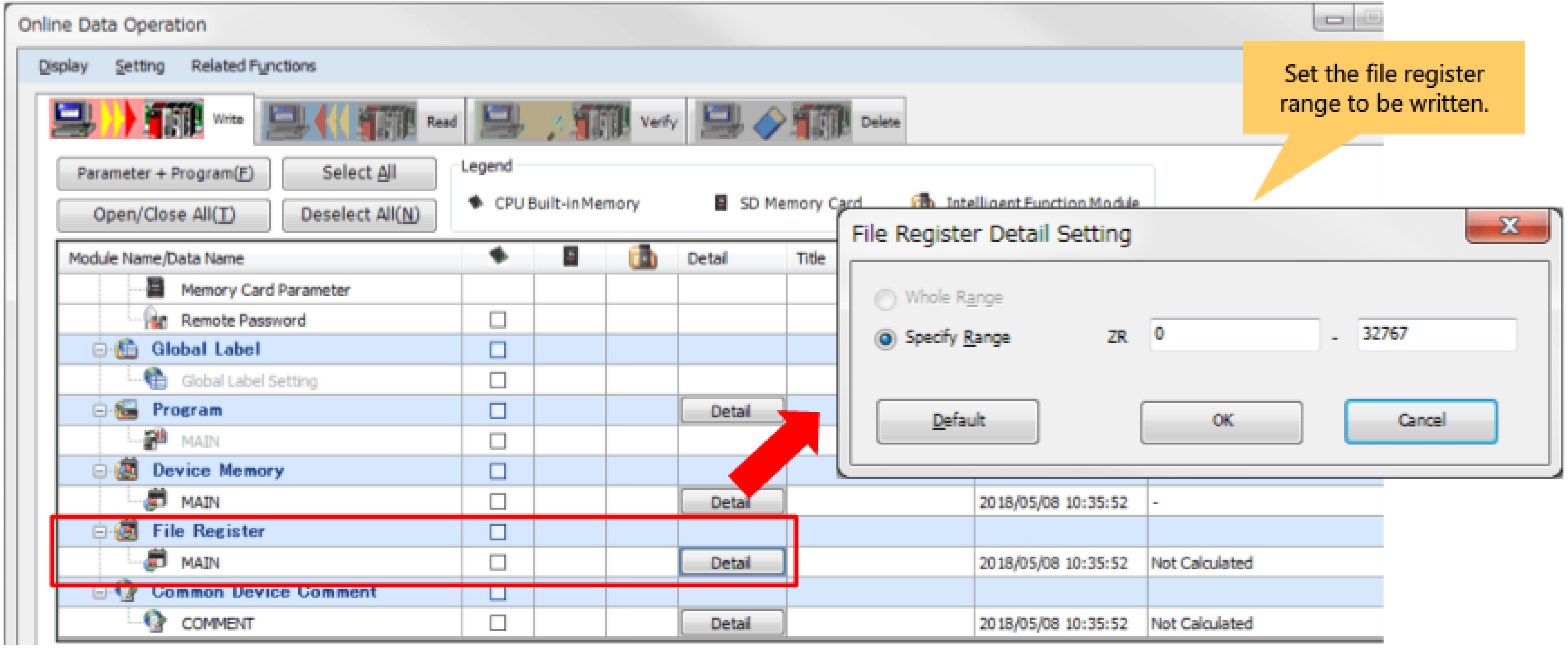
2.9.2 Setting for the file register

This section describes the setting that specifies a local file register as the storage destination of each program.
Select the "File Register Setting" in CPU parameter, and select "Use File Register of Each Program".



Item	Setting
⊟ **File Register Setting**	
⊟ Use Or Not Setting	Use File Register of Each Program
Capacity	
File Name	

To write data to the programmable controller, the file register setting must be written to each program.



Online Data Operation

Module Name/Data Name	◆			Detail	Title		
Memory Card Parameter							
Remote Password	□						
⊟ **Global Label**	□						
Global Label Setting	□						
⊟ **Program**	□			Detail			
MAIN	□						
⊟ **Device Memory**	□						
MAIN	□			Detail		2018/05/08 10:35:52	-
⊟ **File Register**	□						
MAIN	□			Detail		2018/05/08 10:35:52	Not Calculated
⊟ **Common Device Comment**	□						
COMMENT	□			Detail		2018/05/08 10:35:52	Not Calculated

File Register Detail Setting

○ Whole Range
● Specify Range ZR 0 - 32767

Set the file register range to be written.

2.10 Using devices with predetermined functions and operations

The special relay and the special register used in the CPU module have predetermined functions and operations.
The special relay (SM) is the internal relay used for bit information (on/off), and the special register (SD) is the internal register used for word information.

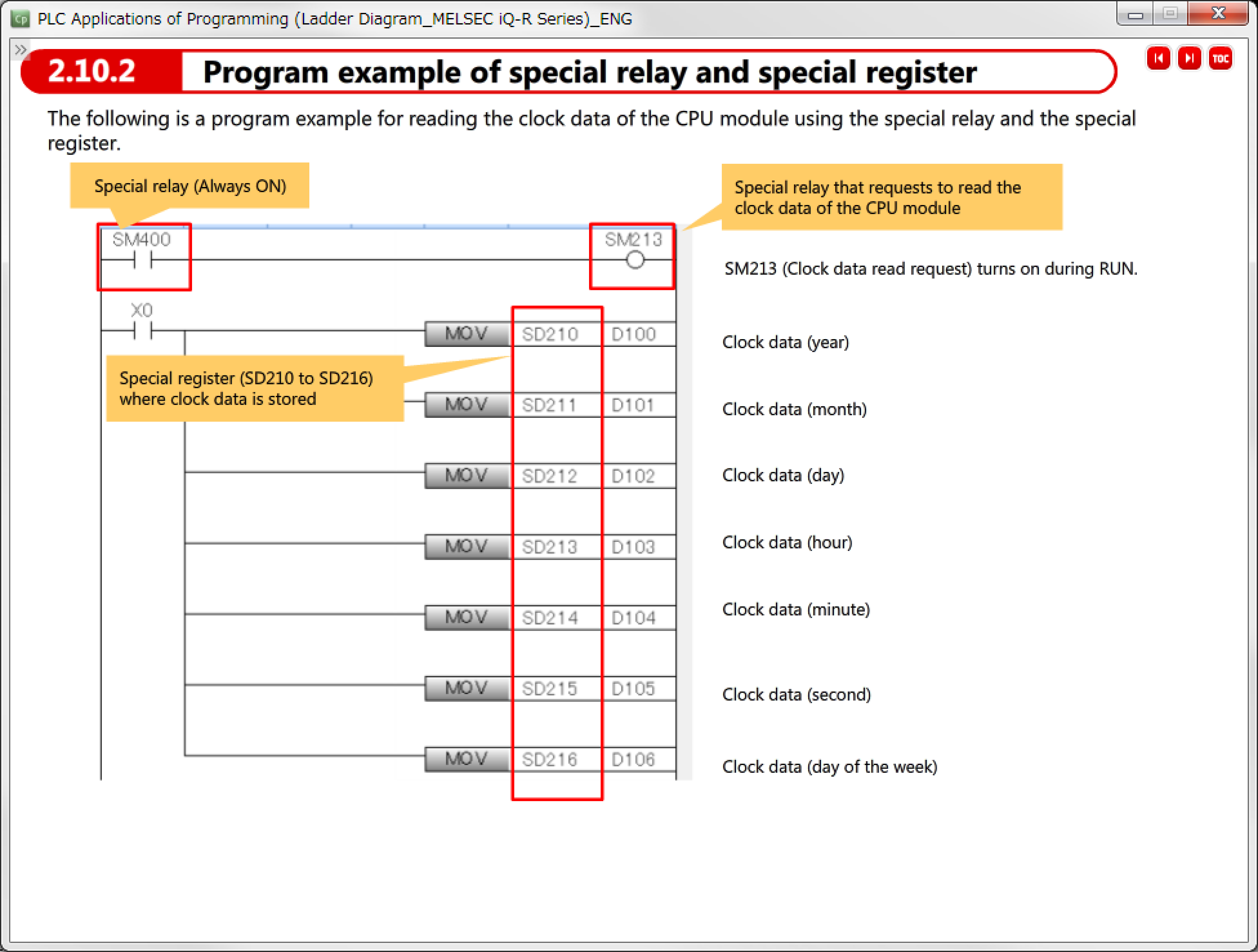
2.10.1 Types of special relay and special register

The special relay and the special register are categorized by their information types. The major types are listed below.
In user programs, the special relay and the special register are used as determination conditions for control.
They are also used for operation monitoring, which can be performed on the device monitor of GX Works3.

Diagnostic information
Diagnostic results of the CPU module
Diagnostic errors and error codes

System information
System information of the CPU module
Operating status of the CPU module, clock data, and other information

System clock
Clock signals and count values that are used as basic timing elements
Various clock signals (always ON, ON/OFF at specified intervals, and other signals)


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC

2.10.2 Program example of special relay and special register

The following is a program example for reading the clock data of the CPU module using the special relay and the special register.

Special relay (Always ON)

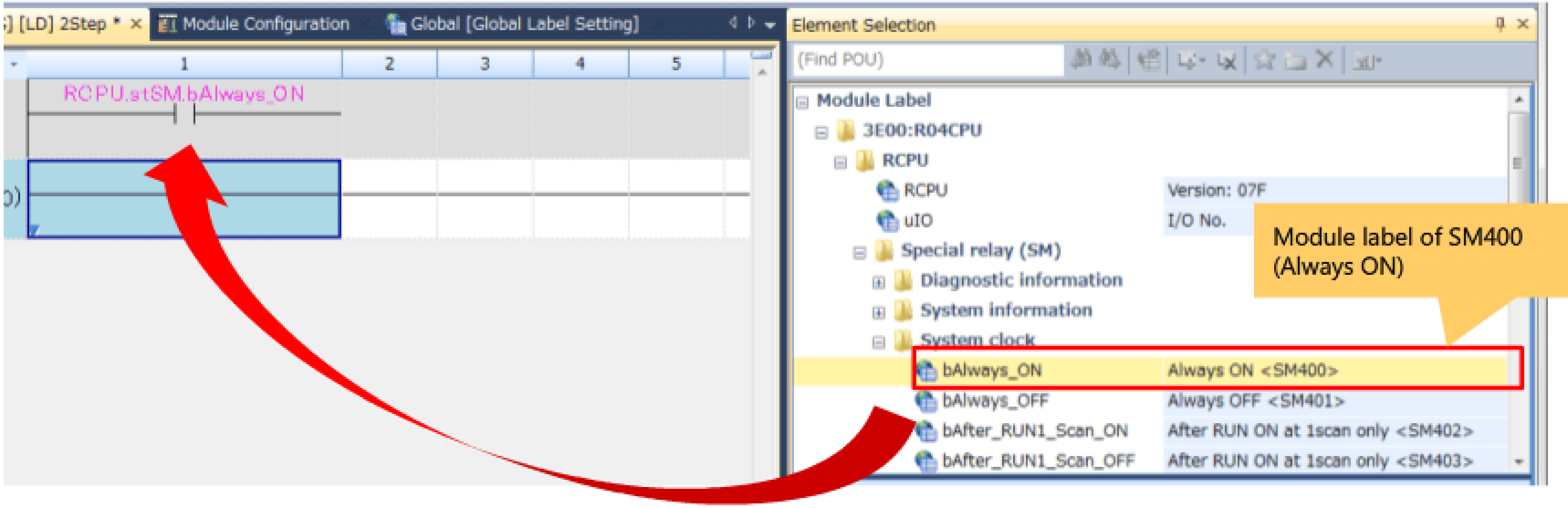
Special relay that requests to read the clock data of the CPU module

Special register (SD210 to SD216) where clock data is stored

```
SM400                                   SM213
--| |-----------------------------------( )--    SM213 (Clock data read request) turns on during RUN.
X0
--| |--+------------------[MOV  SD210  D100]    Clock data (year)
       +------------------[MOV  SD211  D101]    Clock data (month)
       +------------------[MOV  SD212  D102]    Clock data (day)
       +------------------[MOV  SD213  D103]    Clock data (hour)
       +------------------[MOV  SD214  D104]    Clock data (minute)
       +------------------[MOV  SD215  D105]    Clock data (second)
       +------------------[MOV  SD216  D106]    Clock data (day of the week)
```

2.10.3 Using labels for special relay and special register

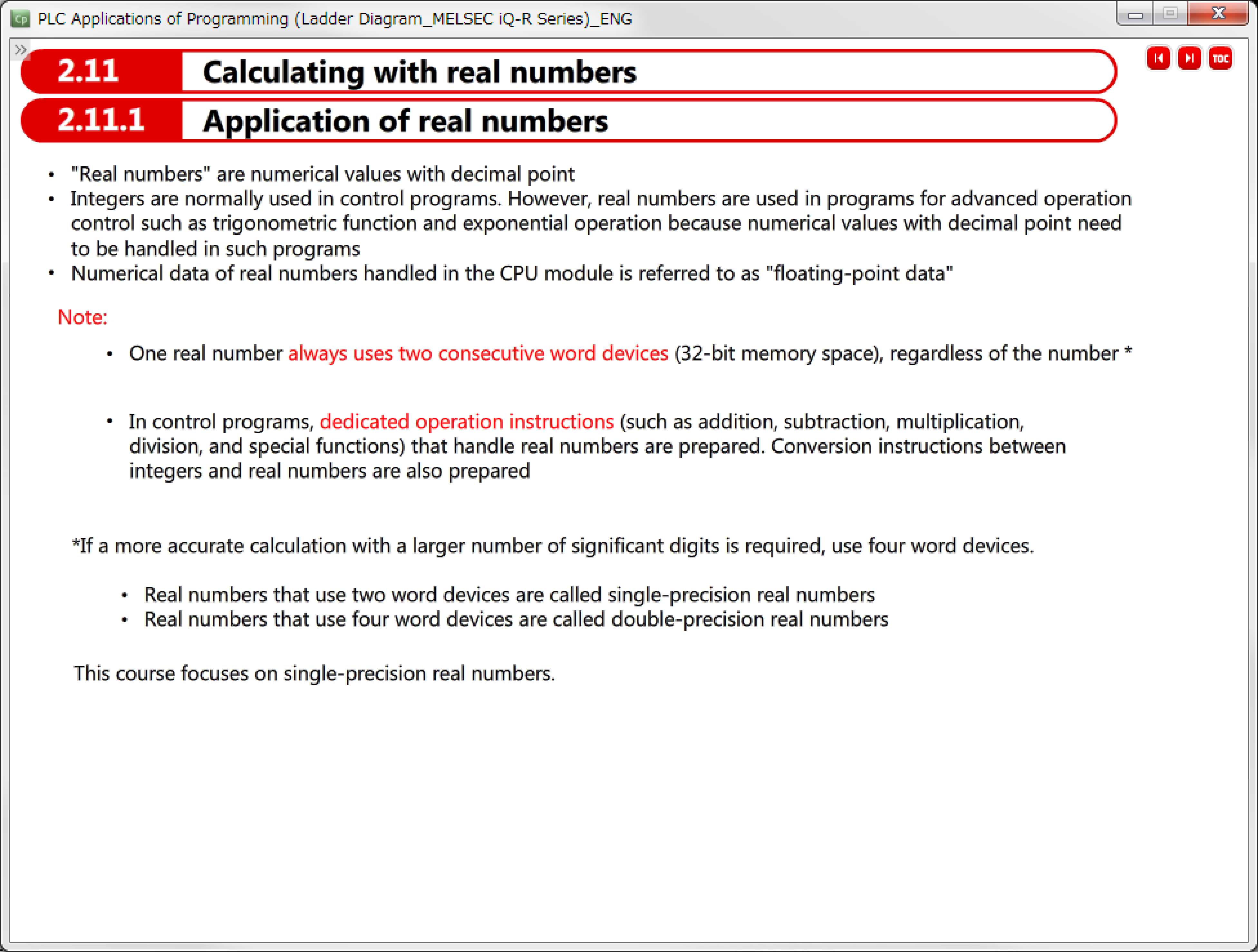
The special relay and the special register are prepared as module labels in the CPU module. They can be used simply by selecting and placing the relevant label names, without checking the device numbers in the manual.



- Module Label
  - 3E00:R04CPU
    - RCPU

RCPU	Version: 07F
uIO	I/O No.
Special relay (SM)	
Diagnostic information	
System information	
System clock	
bAlways_ON	Always ON <SM400>
bAlways_OFF	Always OFF <SM401>
bAfter_RUN1_Scan_ON	After RUN ON at 1scan only <SM402>
bAfter_RUN1_Scan_OFF	After RUN ON at 1scan only <SM403>

Module label of SM400 (Always ON)



2.11 Calculating with real numbers

2.11.1 Application of real numbers

- "Real numbers" are numerical values with decimal point
- Integers are normally used in control programs. However, real numbers are used in programs for advanced operation control such as trigonometric function and exponential operation because numerical values with decimal point need to be handled in such programs
- Numerical data of real numbers handled in the CPU module is referred to as "floating-point data"

Note:

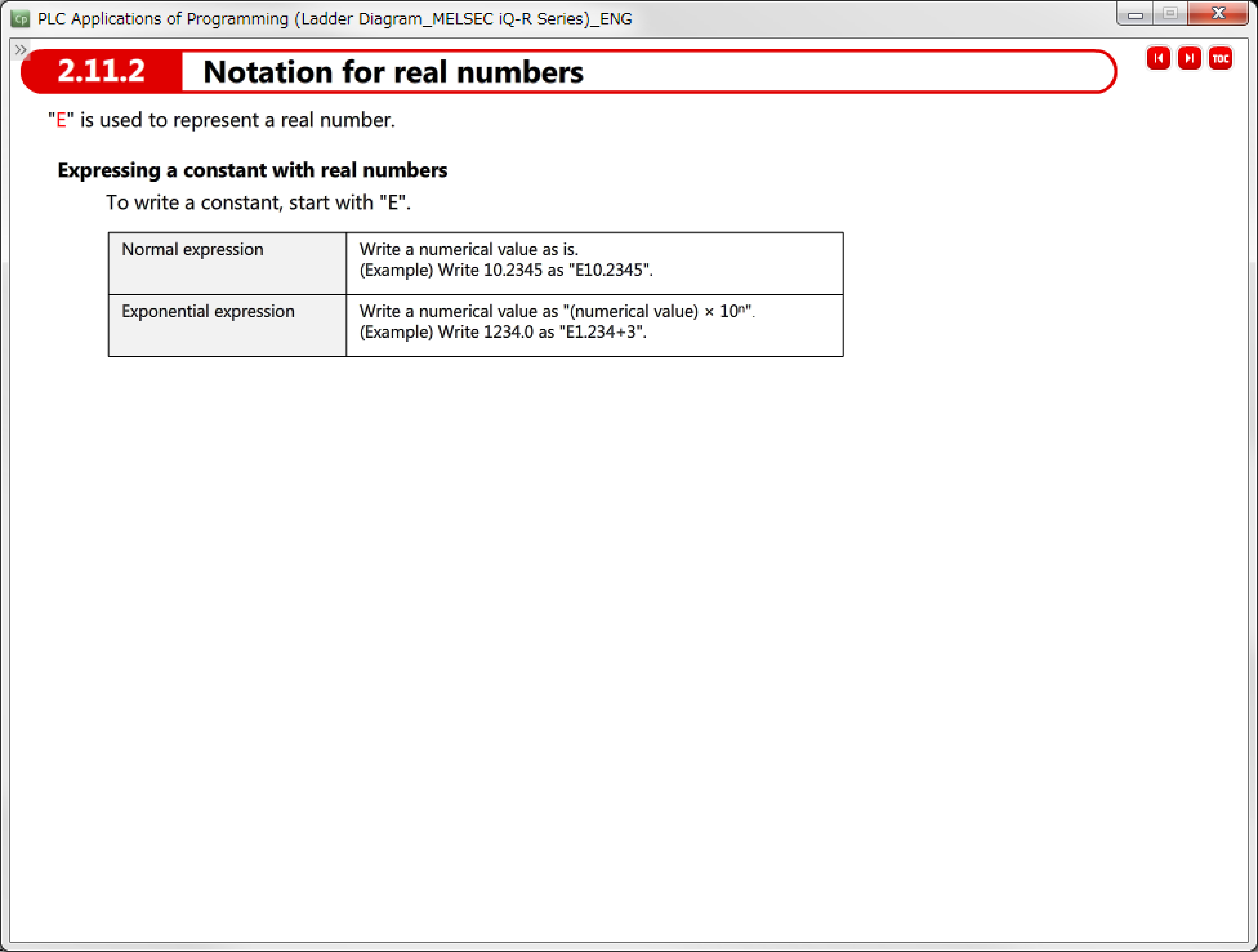
- One real number always uses two consecutive word devices (32-bit memory space), regardless of the number *

- In control programs, dedicated operation instructions (such as addition, subtraction, multiplication, division, and special functions) that handle real numbers are prepared. Conversion instructions between integers and real numbers are also prepared

*If a more accurate calculation with a larger number of significant digits is required, use four word devices.

- Real numbers that use two word devices are called single-precision real numbers
- Real numbers that use four word devices are called double-precision real numbers

This course focuses on single-precision real numbers.



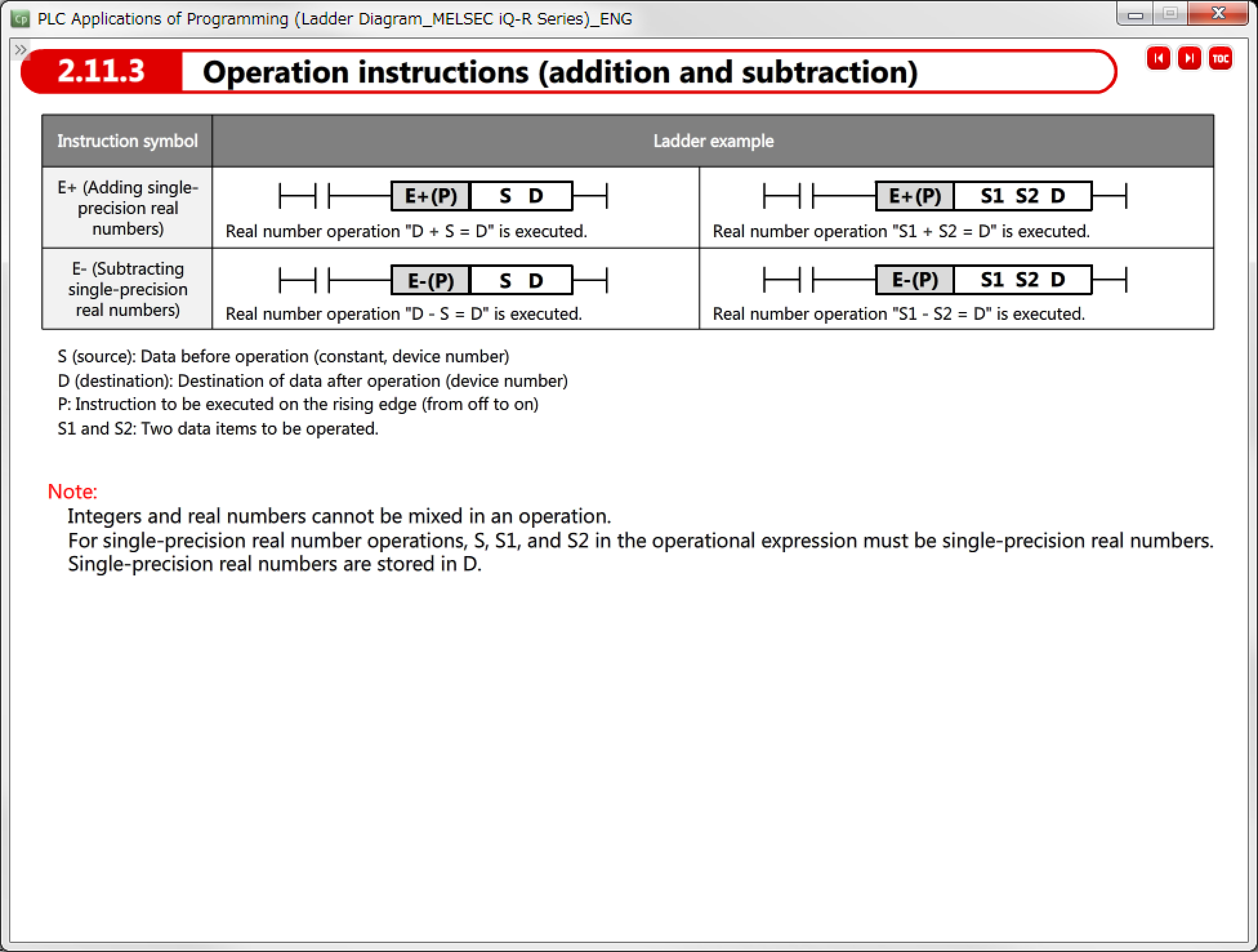
2.11.2 Notation for real numbers

"E" is used to represent a real number.

Expressing a constant with real numbers

To write a constant, start with "E".

Normal expression	Write a numerical value as is. (Example) Write 10.2345 as "E10.2345".
Exponential expression	Write a numerical value as "(numerical value) × 10^n". (Example) Write 1234.0 as "E1.234+3".



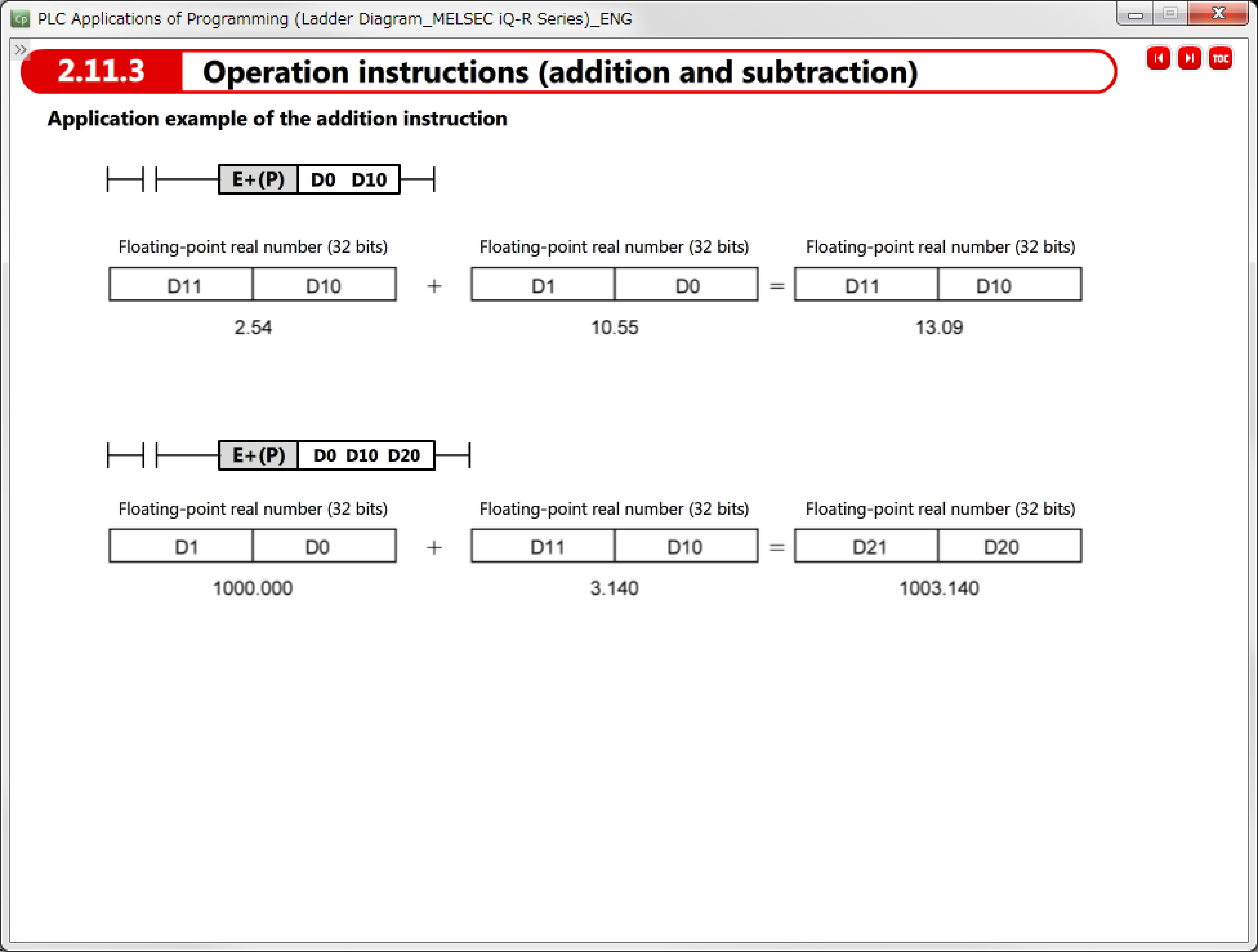
2.11.3 Operation instructions (addition and subtraction)

Instruction symbol	Ladder example	
E+ (Adding single-precision real numbers)	E+(P) S D Real number operation "D + S = D" is executed.	E+(P) S1 S2 D Real number operation "S1 + S2 = D" is executed.
E- (Subtracting single-precision real numbers)	E-(P) S D Real number operation "D - S = D" is executed.	E-(P) S1 S2 D Real number operation "S1 - S2 = D" is executed.

S (source): Data before operation (constant, device number)
D (destination): Destination of data after operation (device number)
P: Instruction to be executed on the rising edge (from off to on)
S1 and S2: Two data items to be operated.

Note:

Integers and real numbers cannot be mixed in an operation.
For single-precision real number operations, S, S1, and S2 in the operational expression must be single-precision real numbers.
Single-precision real numbers are stored in D.


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC

2.11.3 Operation instructions (addition and subtraction)

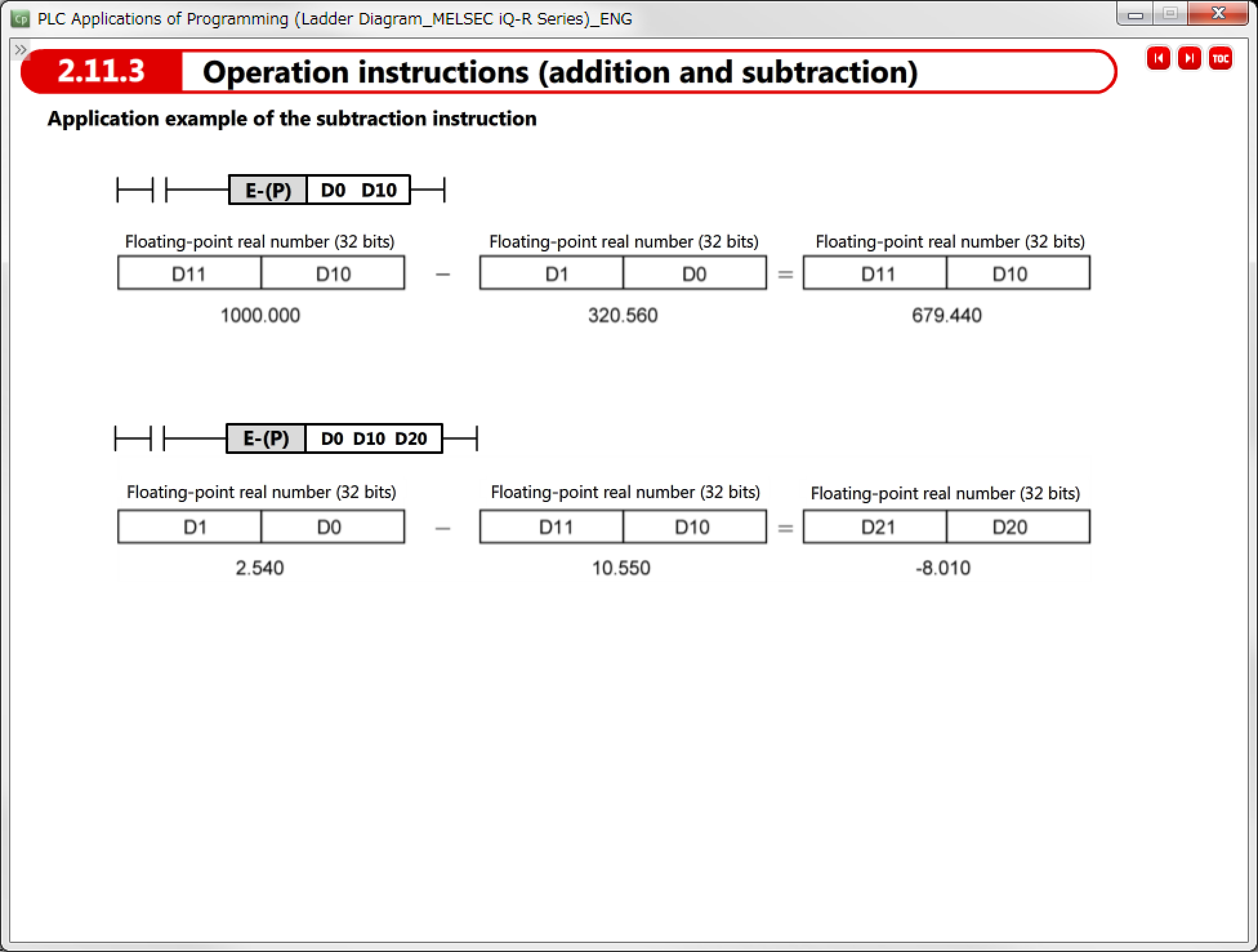
Application example of the addition instruction

E+(P) D0 D10

Floating-point real number (32 bits)			Floating-point real number (32 bits)			Floating-point real number (32 bits)	
D11	D10	+	D1	D0	=	D11	D10
2.54			10.55			13.09	

E+(P) D0 D10 D20

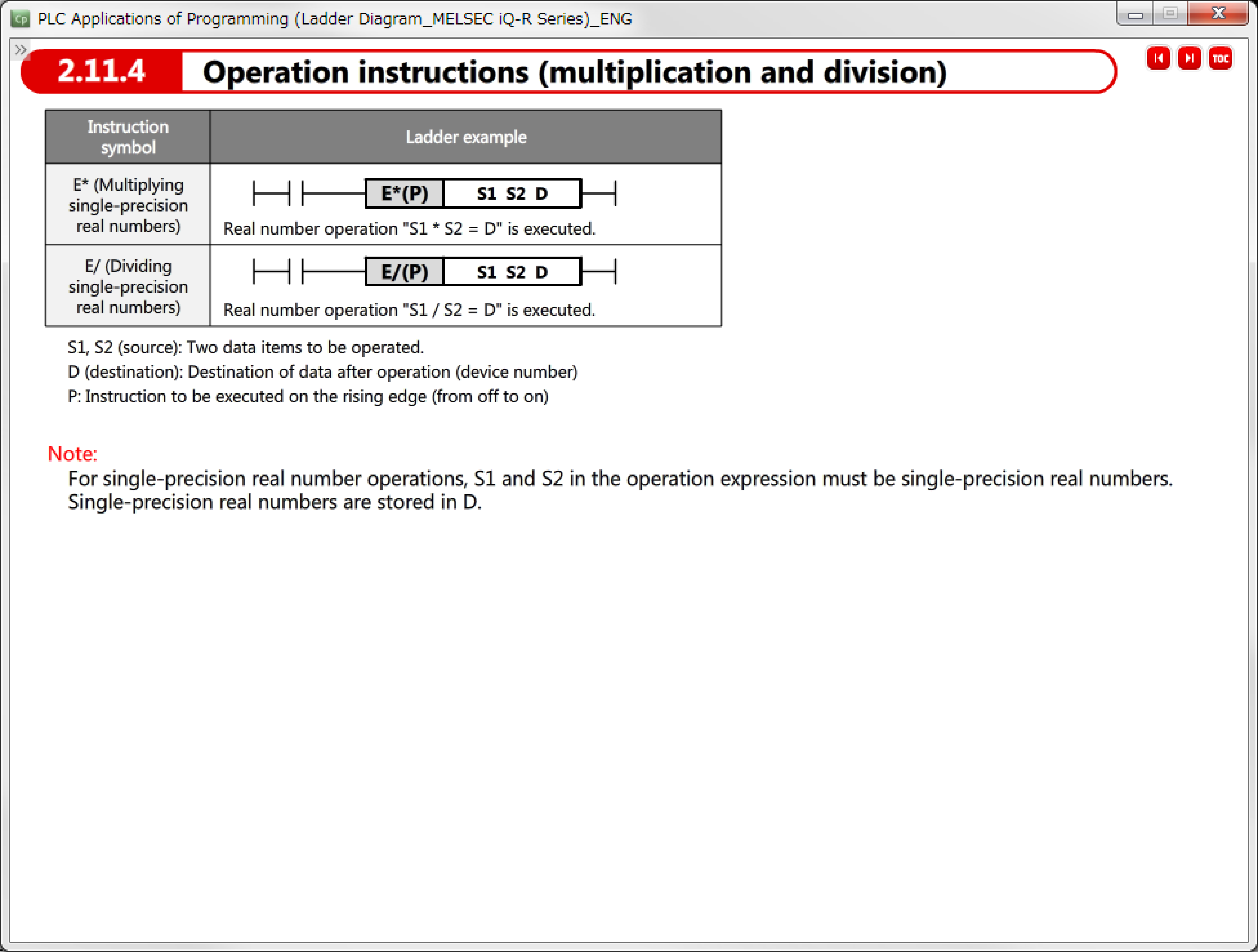
Floating-point real number (32 bits)			Floating-point real number (32 bits)			Floating-point real number (32 bits)	
D1	D0	+	D11	D10	=	D21	D20
1000.000			3.140			1003.140	



2.11.3 Operation instructions (addition and subtraction)

Application example of the subtraction instruction

E-(P)
D0 D10
Floating-point real number (32 bits)
D11
D10
1000.000
−
Floating-point real number (32 bits)
D1
D0
320.560
=
Floating-point real number (32 bits)
D11
D10
679.440
E-(P)
D0 D10 D20
Floating-point real number (32 bits)
D1
D0
2.540
−
Floating-point real number (32 bits)
D11
D10
10.550
=
Floating-point real number (32 bits)
D21
D20
-8.010
TOC



2.11.4 Operation instructions (multiplication and division)

Instruction symbol	Ladder example
E* (Multiplying single-precision real numbers)	E*(P) S1 S2 D Real number operation "S1 * S2 = D" is executed.
E/ (Dividing single-precision real numbers)	E/(P) S1 S2 D Real number operation "S1 / S2 = D" is executed.

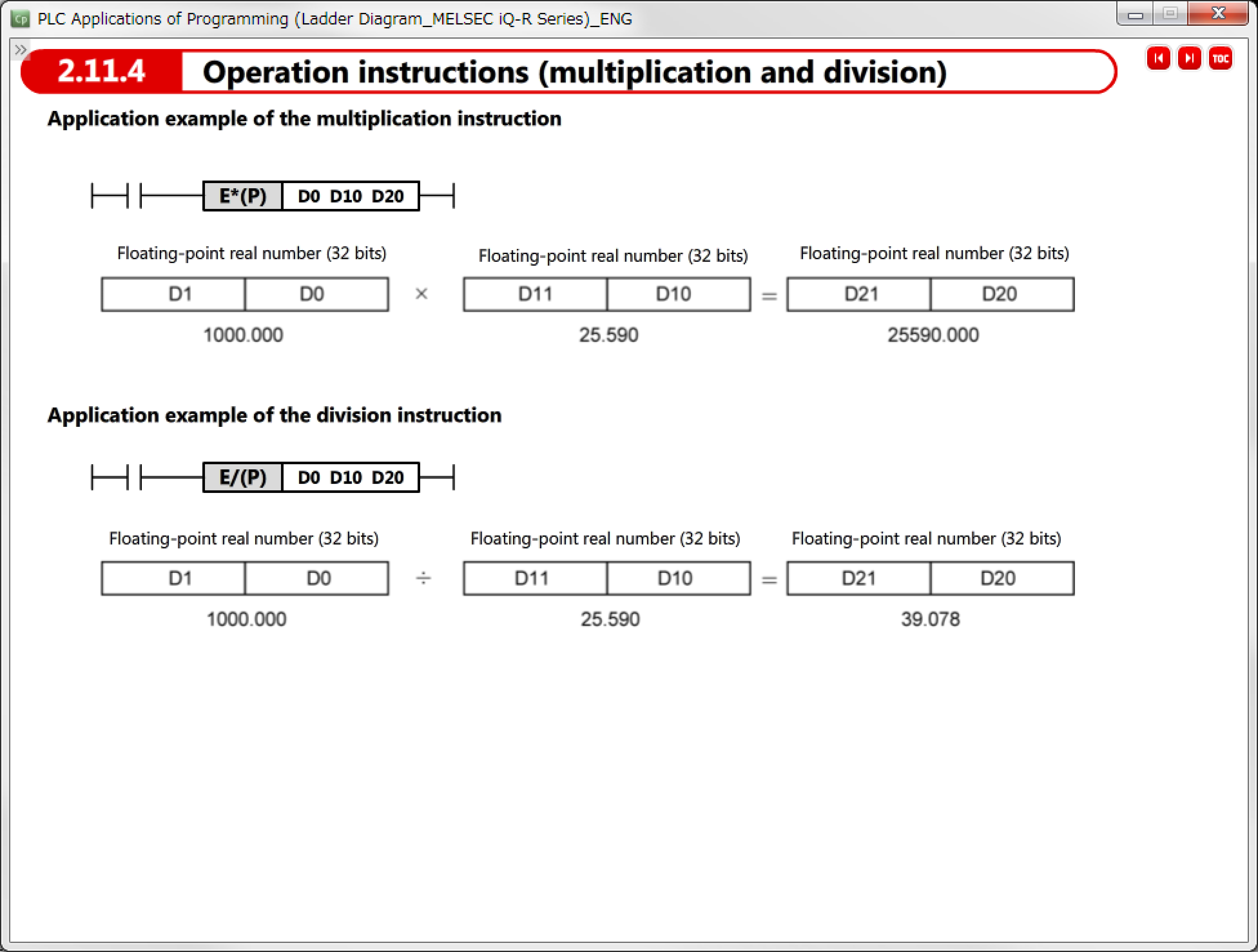
S1, S2 (source): Two data items to be operated.

D (destination): Destination of data after operation (device number)

P: Instruction to be executed on the rising edge (from off to on)

Note:

For single-precision real number operations, S1 and S2 in the operation expression must be single-precision real numbers. Single-precision real numbers are stored in D.



PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG

2.11.4 Operation instructions (multiplication and division)

Application example of the multiplication instruction

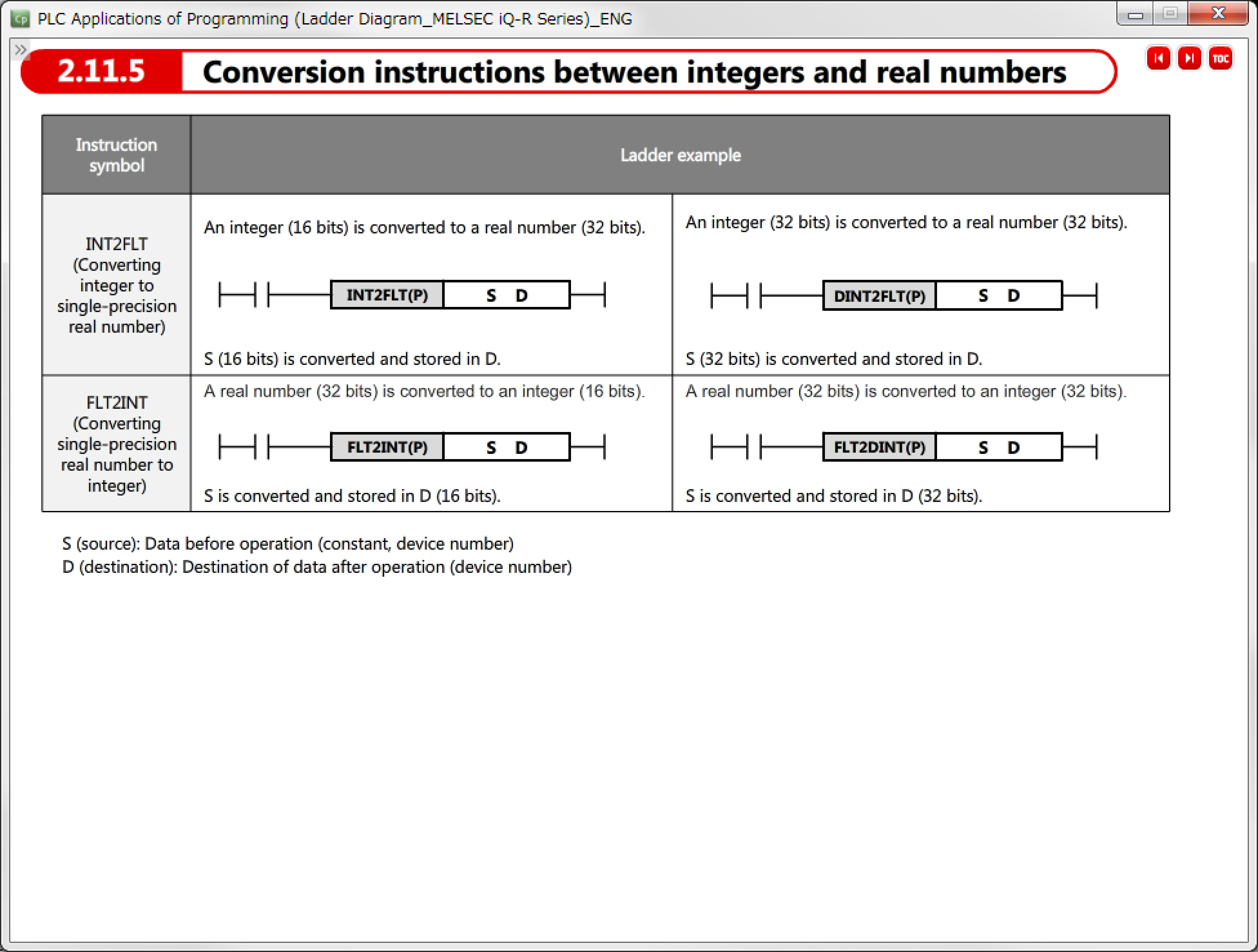
E*(P) D0 D10 D20

Floating-point real number (32 bits)			Floating-point real number (32 bits)			Floating-point real number (32 bits)	
D1	D0	×	D11	D10	=	D21	D20
1000.000			25.590			25590.000	

Application example of the division instruction

E/(P) D0 D10 D20

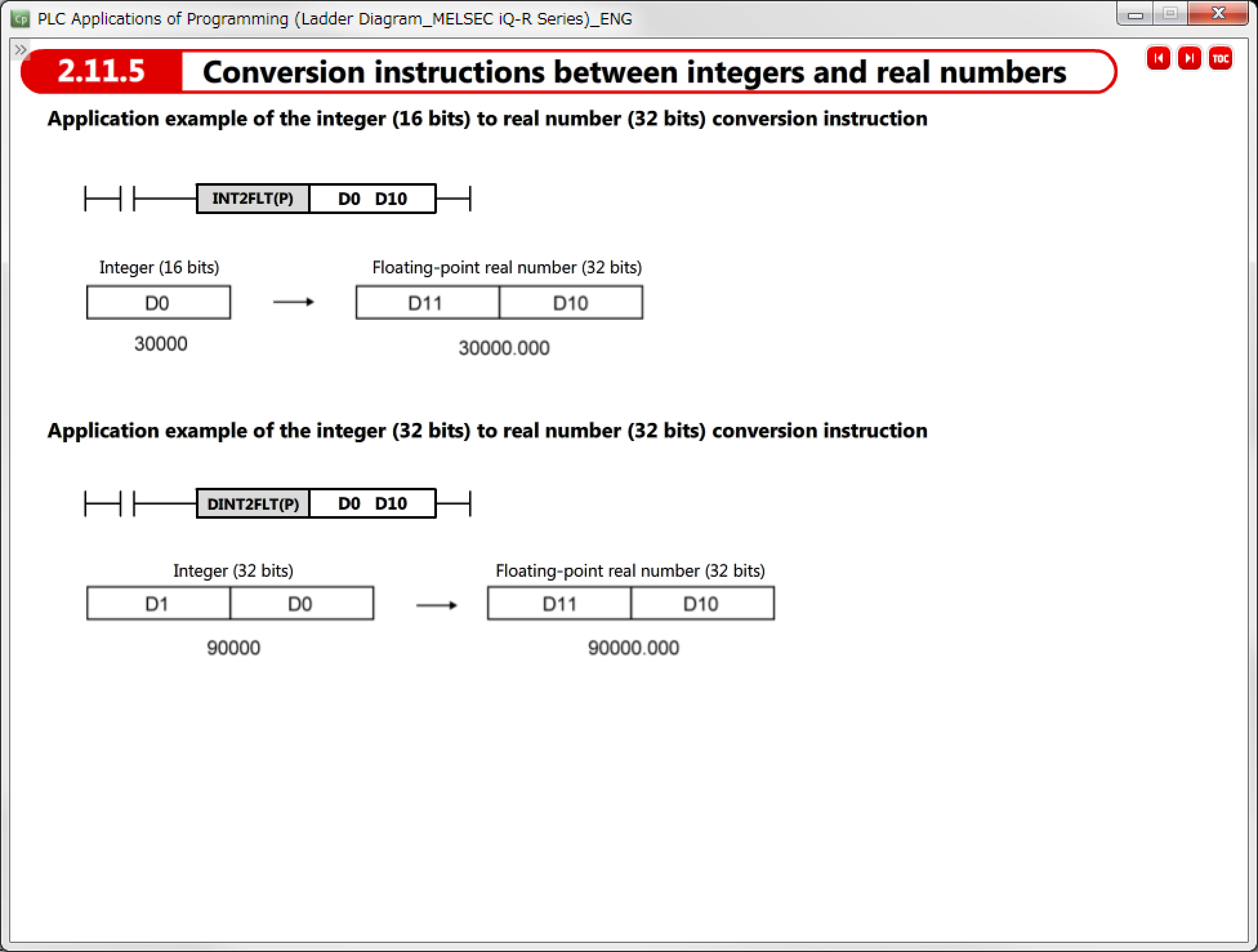
Floating-point real number (32 bits)			Floating-point real number (32 bits)			Floating-point real number (32 bits)	
D1	D0	÷	D11	D10	=	D21	D20
1000.000			25.590			39.078	



2.11.5 Conversion instructions between integers and real numbers

Instruction symbol	Ladder example	
INT2FLT (Converting integer to single-precision real number)	An integer (16 bits) is converted to a real number (32 bits). INT2FLT(P) S D S (16 bits) is converted and stored in D.	An integer (32 bits) is converted to a real number (32 bits). DINT2FLT(P) S D S (32 bits) is converted and stored in D.
FLT2INT (Converting single-precision real number to integer)	A real number (32 bits) is converted to an integer (16 bits). FLT2INT(P) S D S is converted and stored in D (16 bits).	A real number (32 bits) is converted to an integer (32 bits). FLT2DINT(P) S D S is converted and stored in D (32 bits).

S (source): Data before operation (constant, device number)
D (destination): Destination of data after operation (device number)


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC

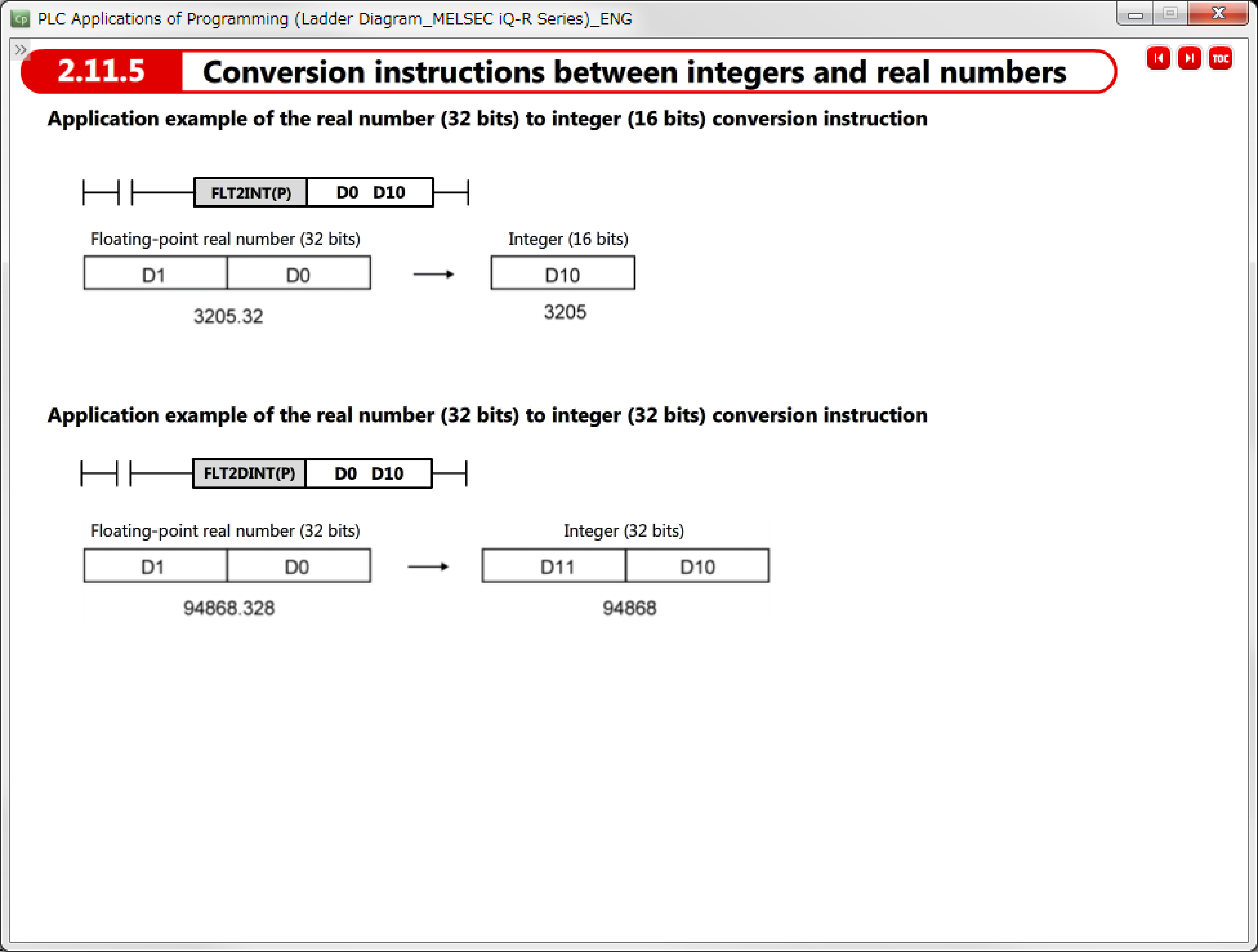
2.11.5 Conversion instructions between integers and real numbers

Application example of the integer (16 bits) to real number (32 bits) conversion instruction

INT2FLT(P)
D0 D10
Integer (16 bits)
D0
30000
Floating-point real number (32 bits)
D11
D10
30000.000

Application example of the integer (32 bits) to real number (32 bits) conversion instruction

DINT2FLT(P)
D0 D10
Integer (32 bits)
D1
D0
90000
Floating-point real number (32 bits)
D11
D10
90000.000



2.11.5 Conversion instructions between integers and real numbers

Application example of the real number (32 bits) to integer (16 bits) conversion instruction

FLT2INT(P) D0 D10

Floating-point real number (32 bits)			Integer (16 bits)
D1	D0	→	D10
3205.32			3205

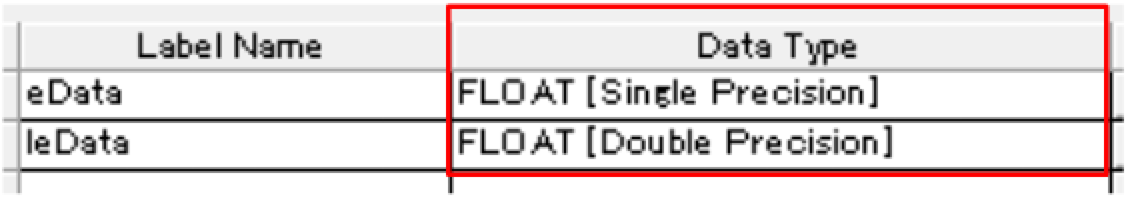
Application example of the real number (32 bits) to integer (32 bits) conversion instruction

FLT2DINT(P) D0 D10

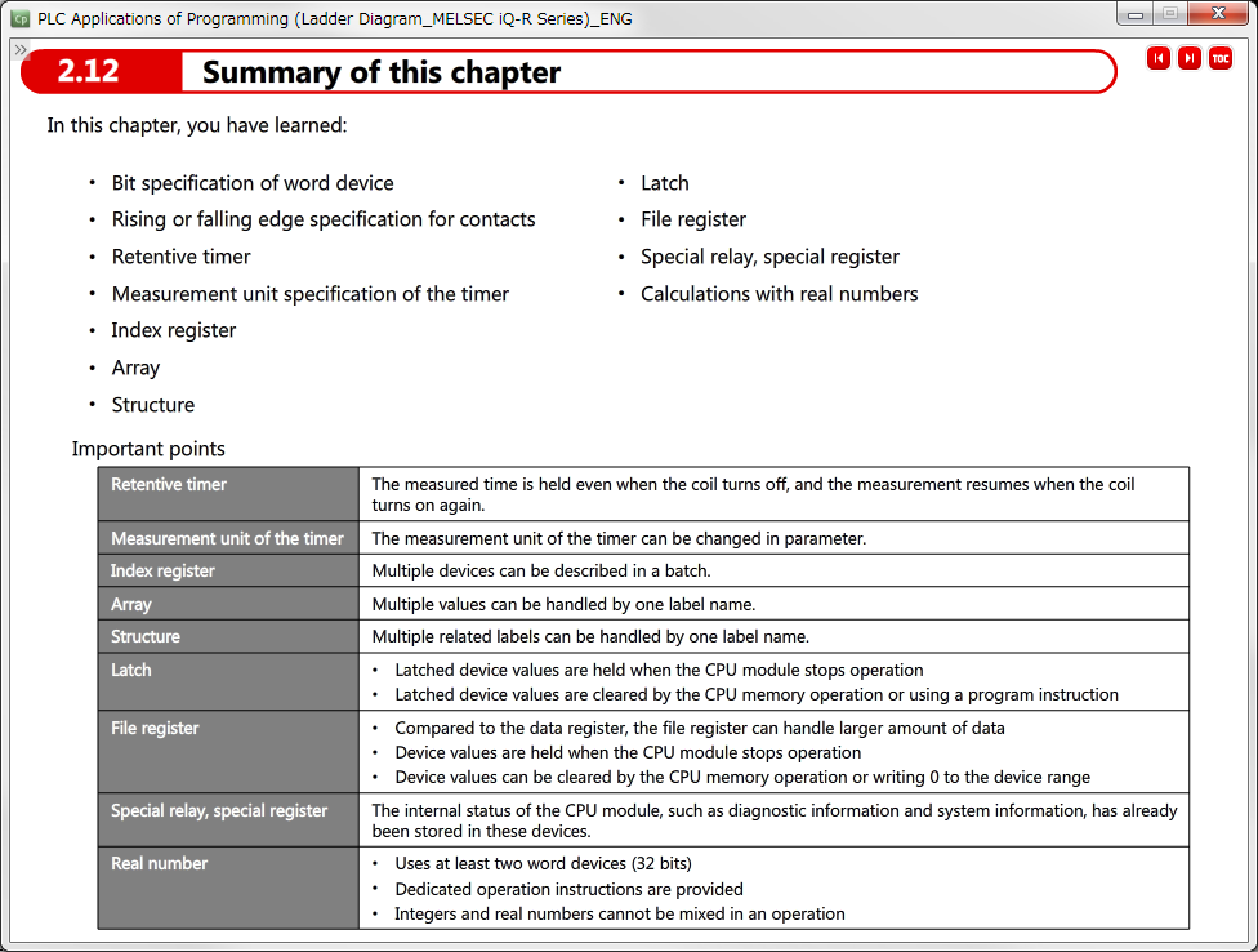
Floating-point real number (32 bits)			Integer (32 bits)	
D1	D0	→	D11	D10
94868.328			94868	

2.11.6 Using labels representing real numbers

To use labels for real numbers, set the data type to "Single Precision" or "Double Precisition" on the window for setting labels.



Label Name	Data Type
eData	FLOAT [Single Precision]
leData	FLOAT [Double Precision]


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC

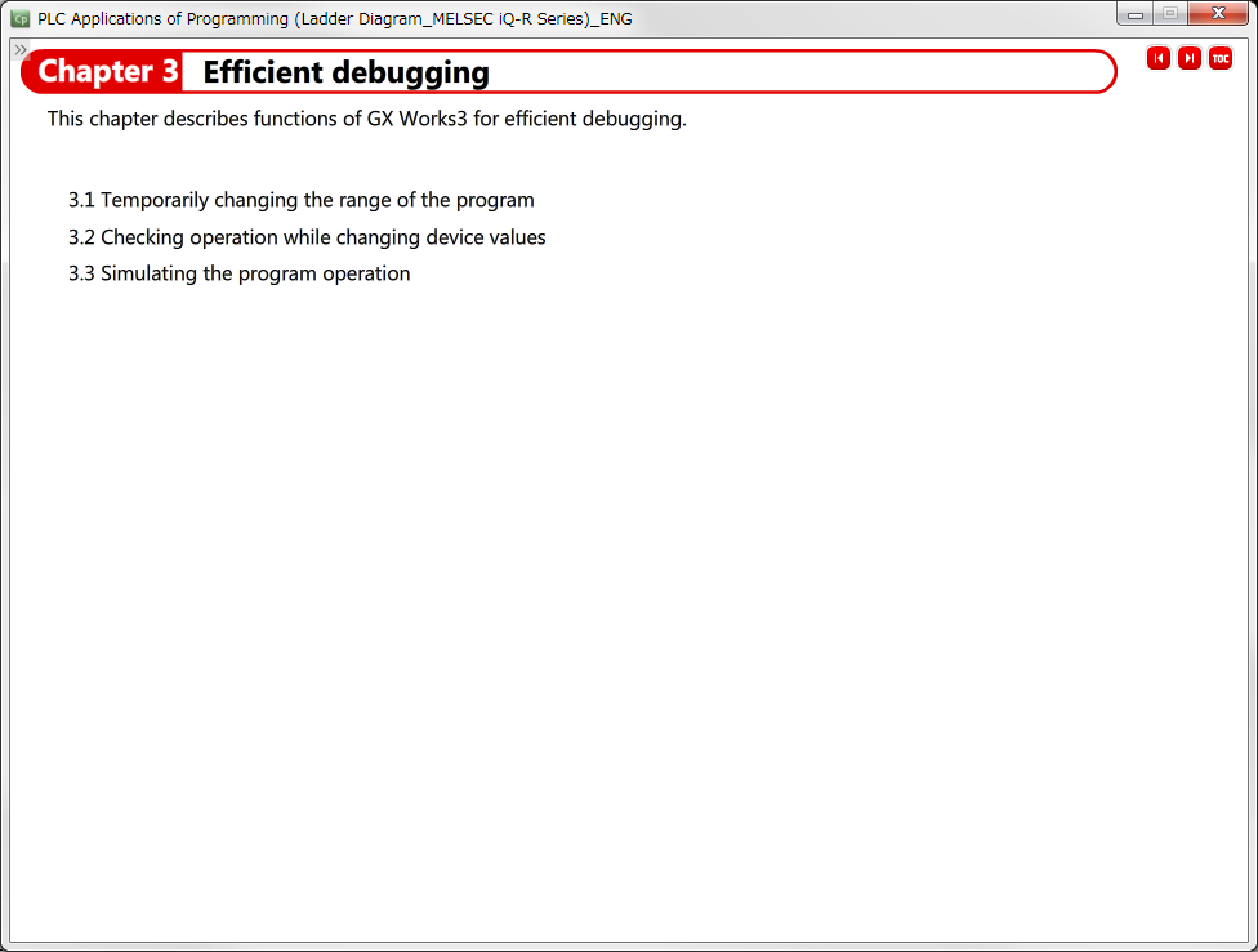
2.12 Summary of this chapter

In this chapter, you have learned:

- Bit specification of word device
- Rising or falling edge specification for contacts
- Retentive timer
- Measurement unit specification of the timer
- Index register
- Array
- Structure
- Latch
- File register
- Special relay, special register
- Calculations with real numbers

Important points

Retentive timer	The measured time is held even when the coil turns off, and the measurement resumes when the coil turns on again.
Measurement unit of the timer	The measurement unit of the timer can be changed in parameter.
Index register	Multiple devices can be described in a batch.
Array	Multiple values can be handled by one label name.
Structure	Multiple related labels can be handled by one label name.
Latch	• Latched device values are held when the CPU module stops operation • Latched device values are cleared by the CPU memory operation or using a program instruction
File register	• Compared to the data register, the file register can handle larger amount of data • Device values are held when the CPU module stops operation • Device values can be cleared by the CPU memory operation or writing 0 to the device range
Special relay, special register	The internal status of the CPU module, such as diagnostic information and system information, has already been stored in these devices.
Real number	• Uses at least two word devices (32 bits) • Dedicated operation instructions are provided • Integers and real numbers cannot be mixed in an operation



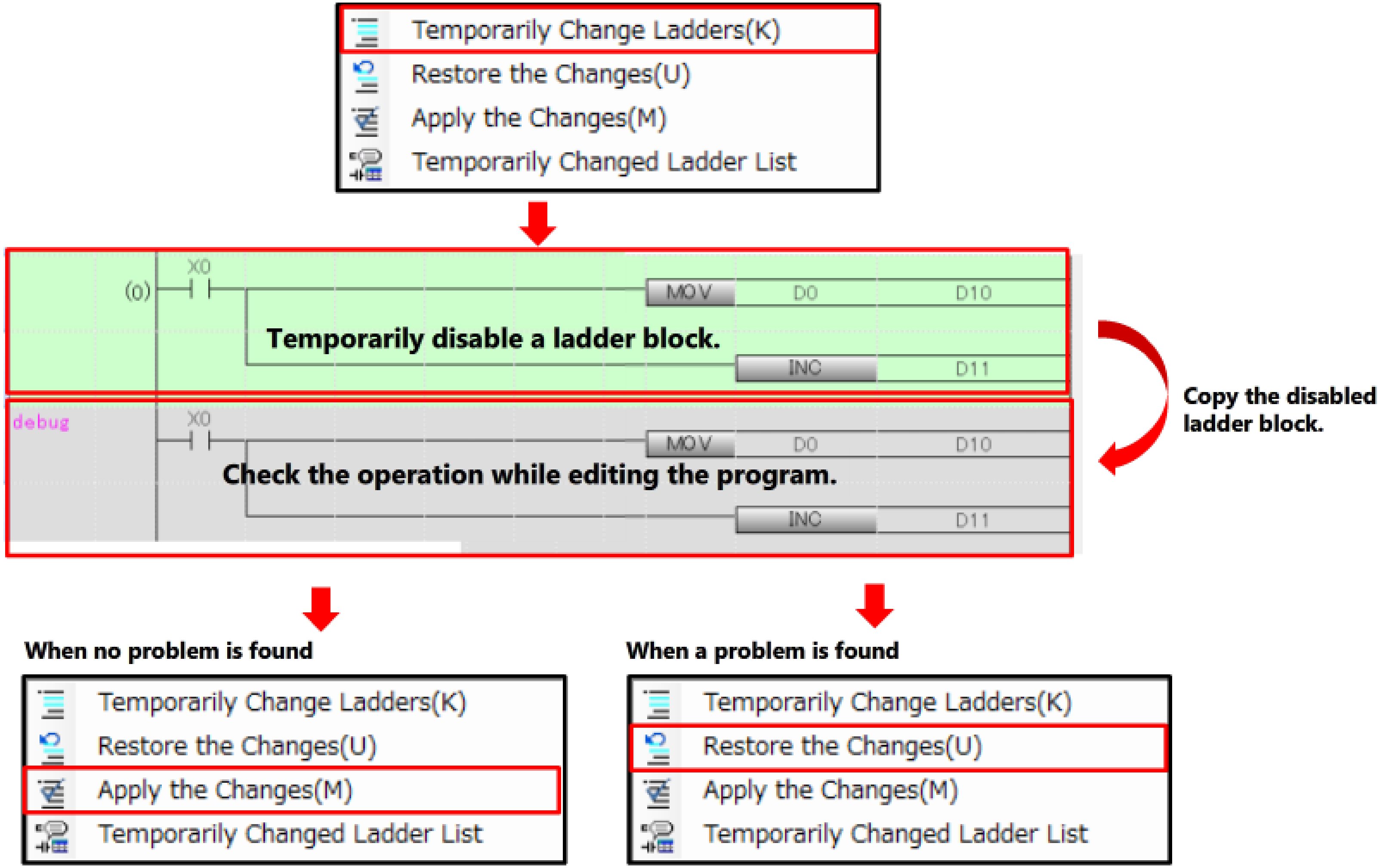
PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG

Chapter 3 Efficient debugging

This chapter describes functions of GX Works3 for efficient debugging.

3.1 Temporarily changing the range of the program

When editing a wide range of program for debugging, it is very difficult to undo all the changes.
By temporarily disabling the desired ladder block, users can use a copy of the program for debugging without changing the original.
(Temporarily change ladders function)
After changing the ladder block and checking the operation with this function, the changes are applied if there is no problem or the changes are restored if there is a problem.


Temporarily Change Ladders(K)
Restore the Changes(U)
Apply the Changes(M)
Temporarily Changed Ladder List
(0)
X0
MOV
D0
D10
INC
D11
debug
X0
MOV
D0
D10
INC
D11
Temporarily Change Ladders(K)
Restore the Changes(U)
Apply the Changes(M)
Temporarily Changed Ladder List
Temporarily Change Ladders(K)
Restore the Changes(U)
Apply the Changes(M)
Temporarily Changed Ladder List

Temporarily disable a ladder block.

Check the operation while editing the program.

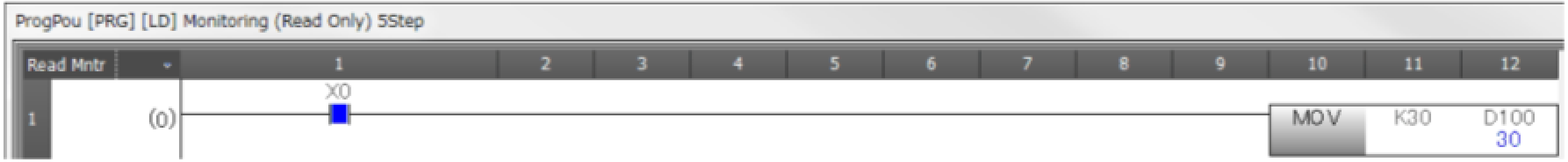
Copy the disabled ladder block.

When no problem is found

When a problem is found

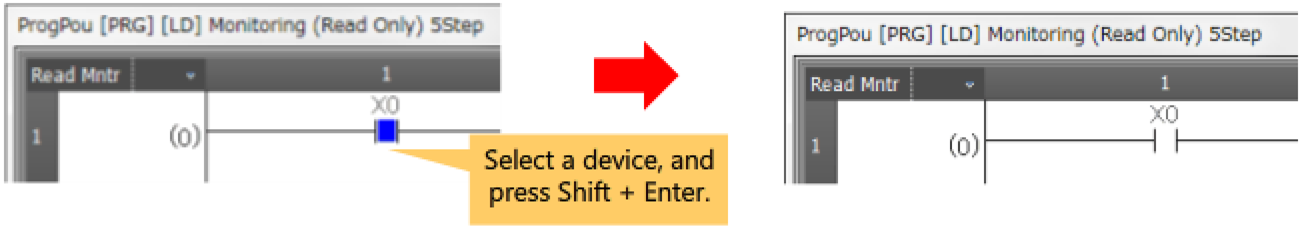
3.2 Checking operation while changing device values

While running a created program, the on/off status of bit devices and the values stored in word devices can be displayed in the program editor. (Monitor function)
With the monitor function, users can easily check the operating status of the program.



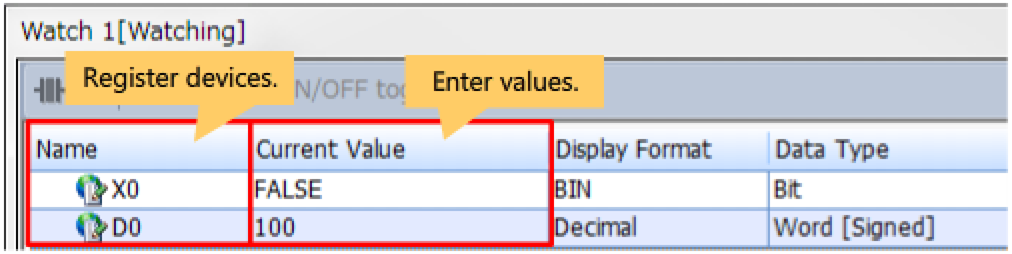
The current values of the device can be forcibly changed during monitoring. (Current value change)
With the current value change function, changes can be made without editing the entire program or running it on the actual system.

The status of bit devices can be switched on the program editor.


ProgPou [PRG] [LD] Monitoring (Read Only) 5Step
Read Mntr
1
1
(0)
X0
ProgPou [PRG] [LD] Monitoring (Read Only) 5Step
Read Mntr
1
1
(0)
X0

Select a device, and press Shift + Enter.

Using the "Watch" window, word devices to be monitored can be registered and their current values can be changed.



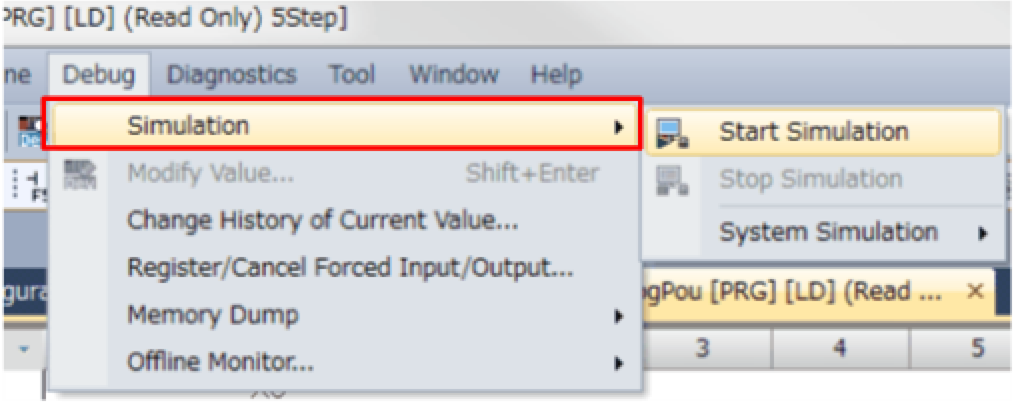
Register devices.

Enter values.

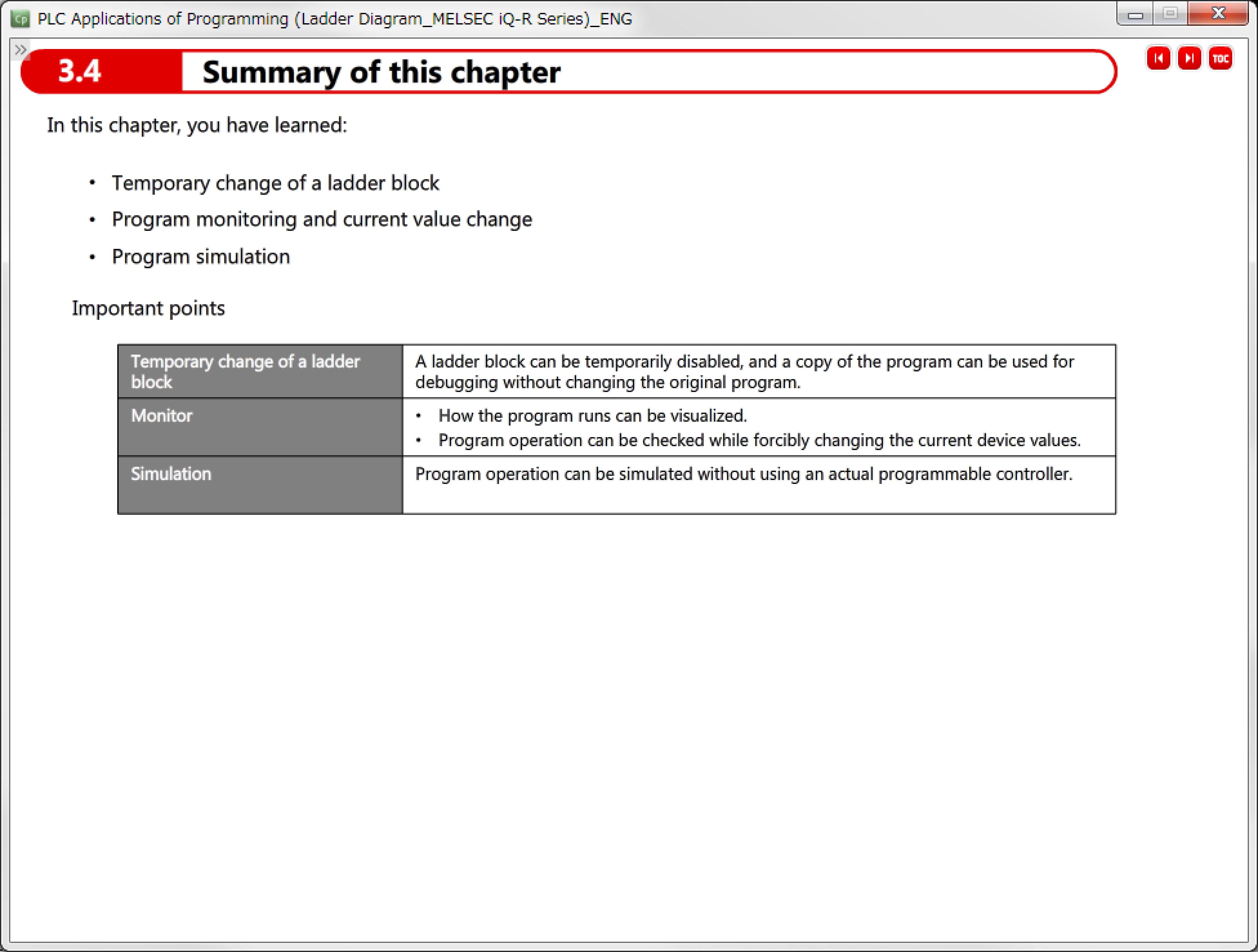
Name	Current Value	Display Format	Data Type
X0	FALSE	BIN	Bit
D0	100	Decimal	Word [Signed]

3.3 Simulating the program operation

If a newly created program is run on the actual system, an unexpected error may occur.
The program operation can be simulated without using an actual programmable controller. (Simulation function)


PRG] [LD] (Read Only) 5Step]
ne
Debug
Diagnostics
Tool
Window
Help
Simulation
Modify Value... Shift+Enter
Change History of Current Value...
Register/Cancel Forced Input/Output...
Memory Dump
Offline Monitor...
Start Simulation
Stop Simulation
System Simulation
gPou [PRG] [LD] (Read ...
3
4
5

With the simulation function, the program operation can be checked as if the program is running on an actual programmable controller.



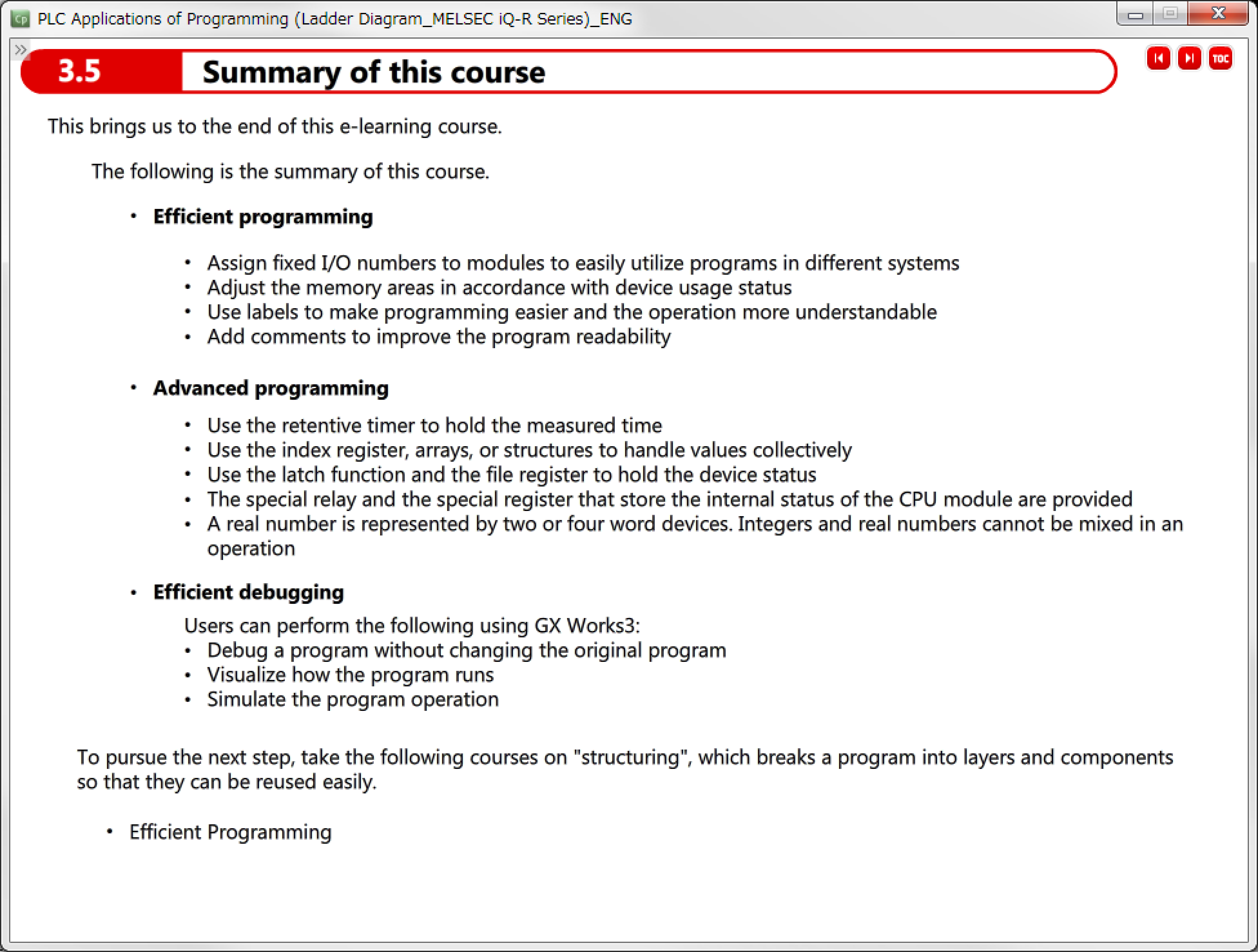
3.4 Summary of this chapter

In this chapter, you have learned:

- Temporary change of a ladder block
- Program monitoring and current value change
- Program simulation

Important points

Temporary change of a ladder block	A ladder block can be temporarily disabled, and a copy of the program can be used for debugging without changing the original program.
Monitor	• How the program runs can be visualized. • Program operation can be checked while forcibly changing the current device values.
Simulation	Program operation can be simulated without using an actual programmable controller.



3.5 Summary of this course

This brings us to the end of this e-learning course.

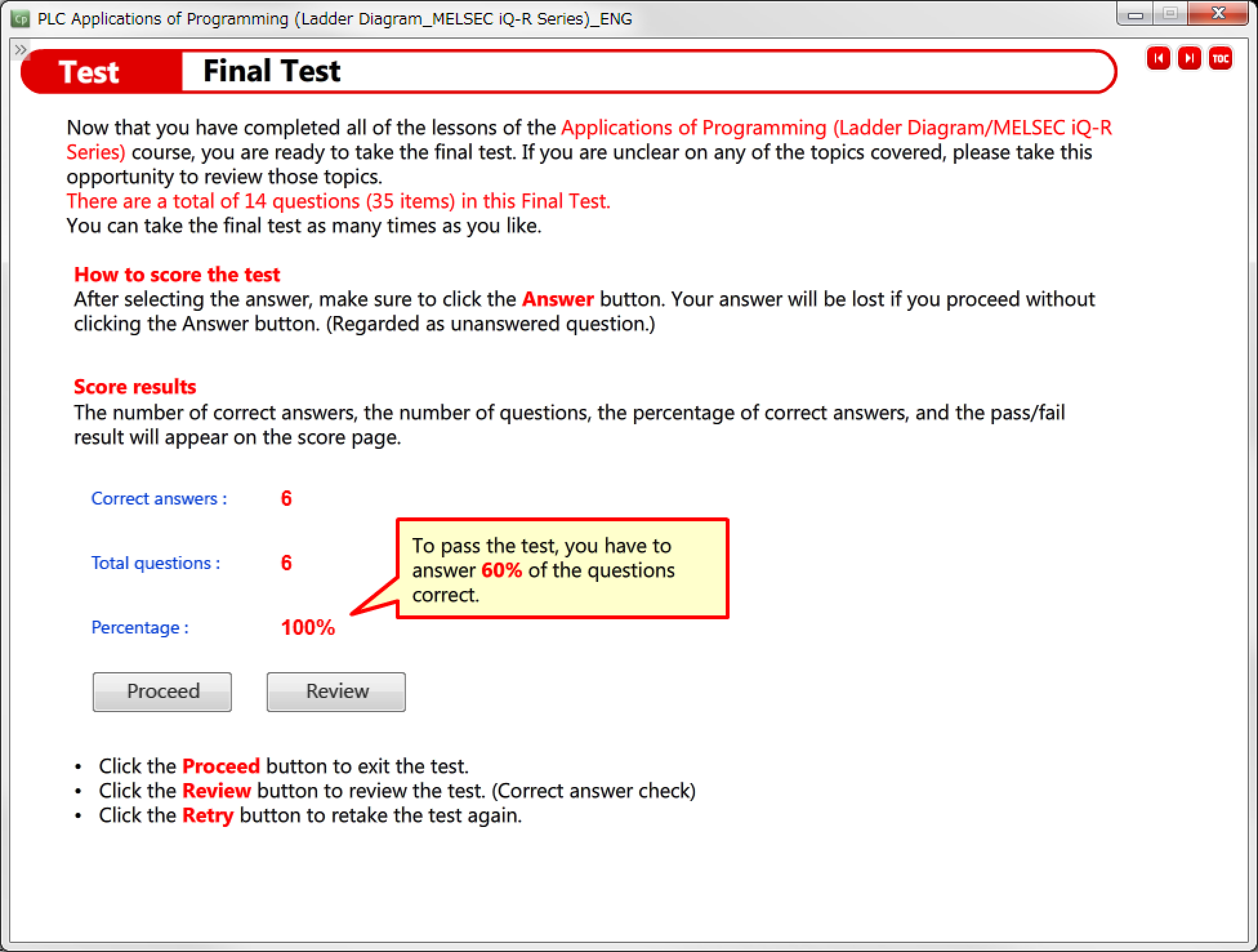
The following is the summary of this course.

- **Efficient programming**
  - Assign fixed I/O numbers to modules to easily utilize programs in different systems
  - Adjust the memory areas in accordance with device usage status
  - Use labels to make programming easier and the operation more understandable
  - Add comments to improve the program readability
- **Advanced programming**
  - Use the retentive timer to hold the measured time
  - Use the index register, arrays, or structures to handle values collectively
  - Use the latch function and the file register to hold the device status
  - The special relay and the special register that store the internal status of the CPU module are provided
  - A real number is represented by two or four word devices. Integers and real numbers cannot be mixed in an operation
- **Efficient debugging**

  Users can perform the following using GX Works3:
  - Debug a program without changing the original program
  - Visualize how the program runs
  - Simulate the program operation

To pursue the next step, take the following courses on "structuring", which breaks a program into layers and components so that they can be reused easily.

- Efficient Programming


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC

Test Final Test

Now that you have completed all of the lessons of the Applications of Programming (Ladder Diagram/MELSEC iQ-R Series) course, you are ready to take the final test. If you are unclear on any of the topics covered, please take this opportunity to review those topics.

There are a total of 14 questions (35 items) in this Final Test.

You can take the final test as many times as you like.

How to score the test

After selecting the answer, make sure to click the **Answer** button. Your answer will be lost if you proceed without clicking the Answer button. (Regarded as unanswered question.)

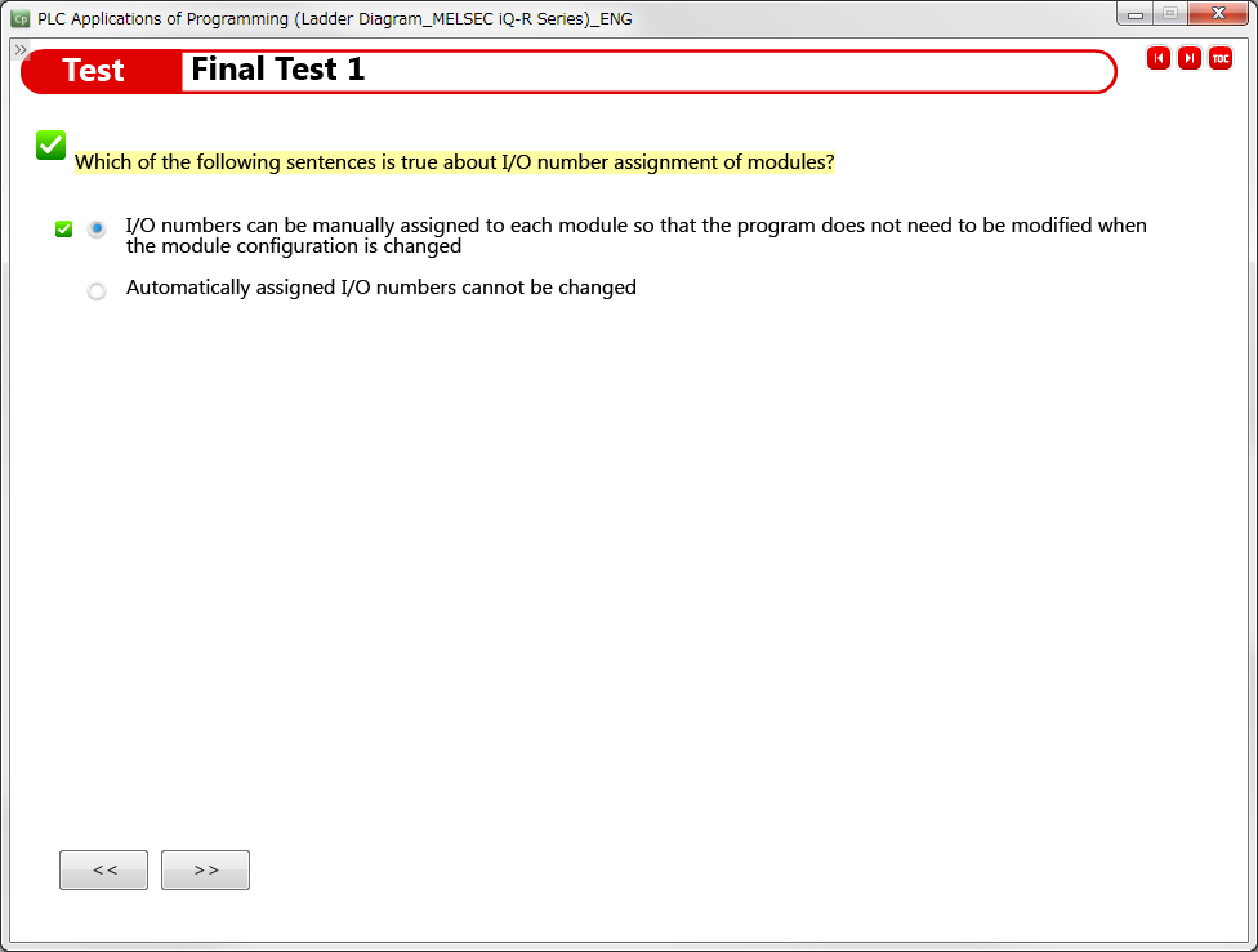
Score results

The number of correct answers, the number of questions, the percentage of correct answers, and the pass/fail result will appear on the score page.

Correct answers :	6
Total questions :	6
Percentage :	100%

To pass the test, you have to answer **60%** of the questions correct.

- Click the **Proceed** button to exit the test.
- Click the **Review** button to review the test. (Correct answer check)
- Click the **Retry** button to retake the test again.

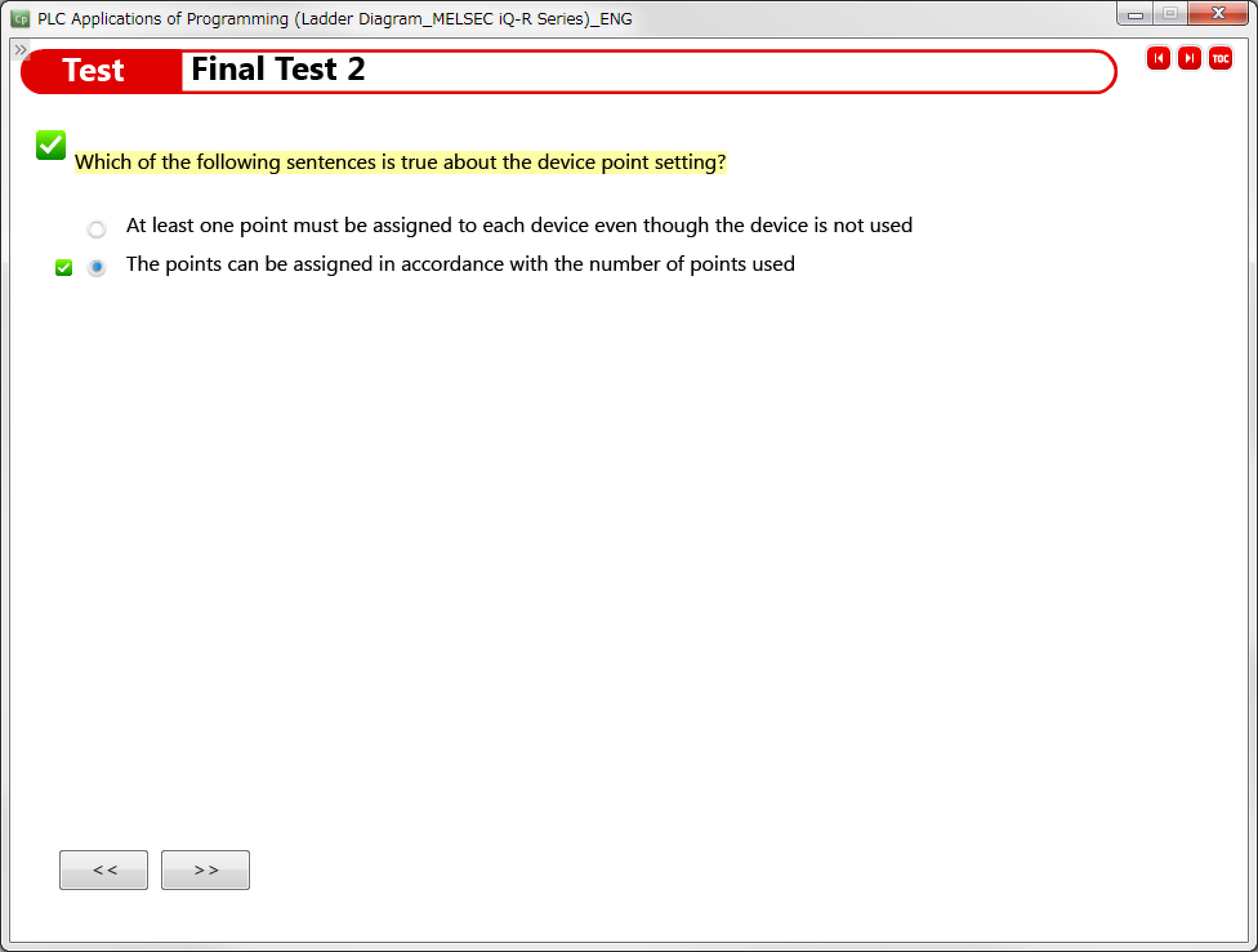

PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
Test
TOC
<<
>>

Final Test 1

✓ Which of the following sentences is true about I/O number assignment of modules?

✓ ● I/O numbers can be manually assigned to each module so that the program does not need to be modified when the module configuration is changed

○ Automatically assigned I/O numbers cannot be changed


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC
<<
>>

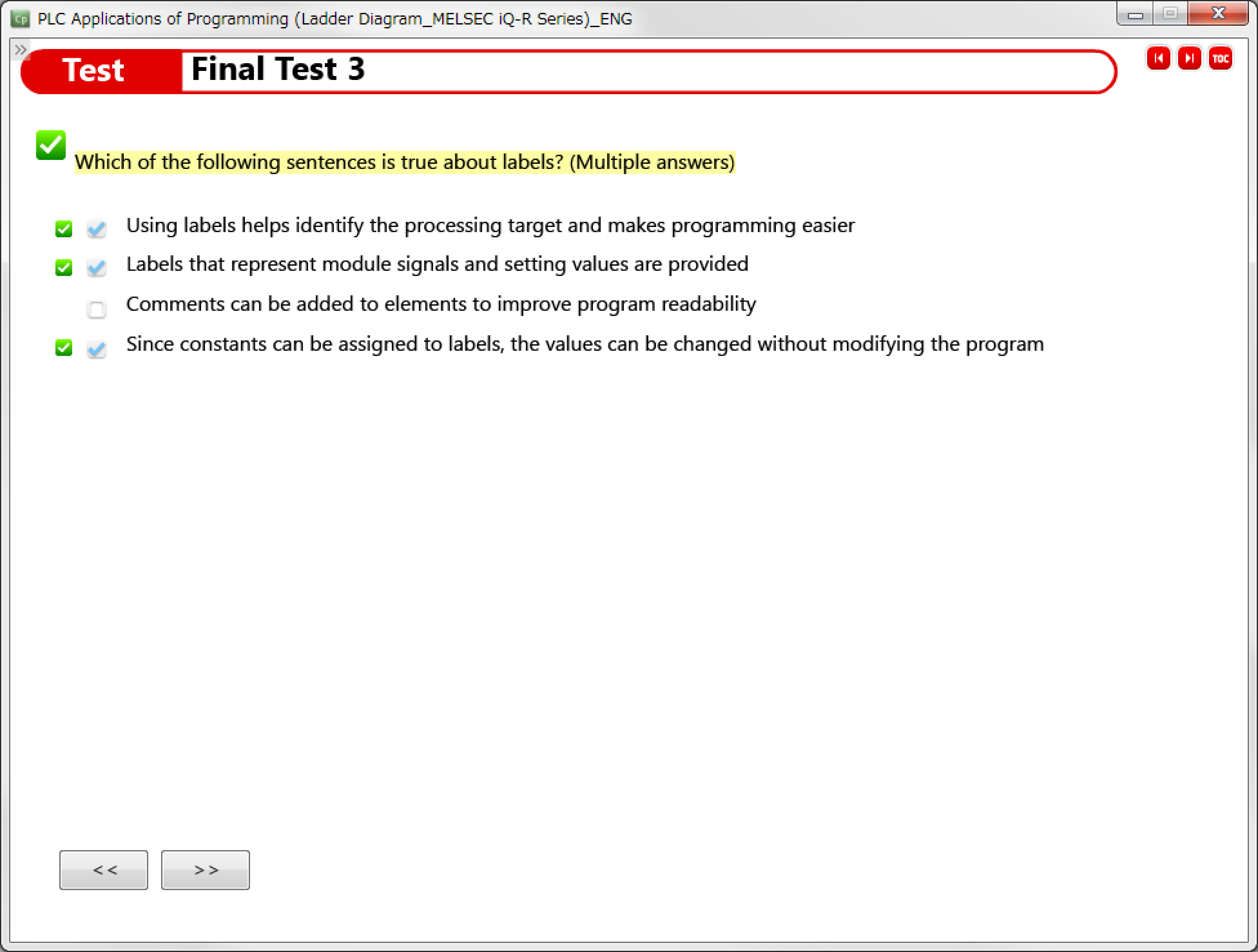
Test

Final Test 2

✓ Which of the following sentences is true about the device point setting?

○ At least one point must be assigned to each device even though the device is not used

✓ ● The points can be assigned in accordance with the number of points used

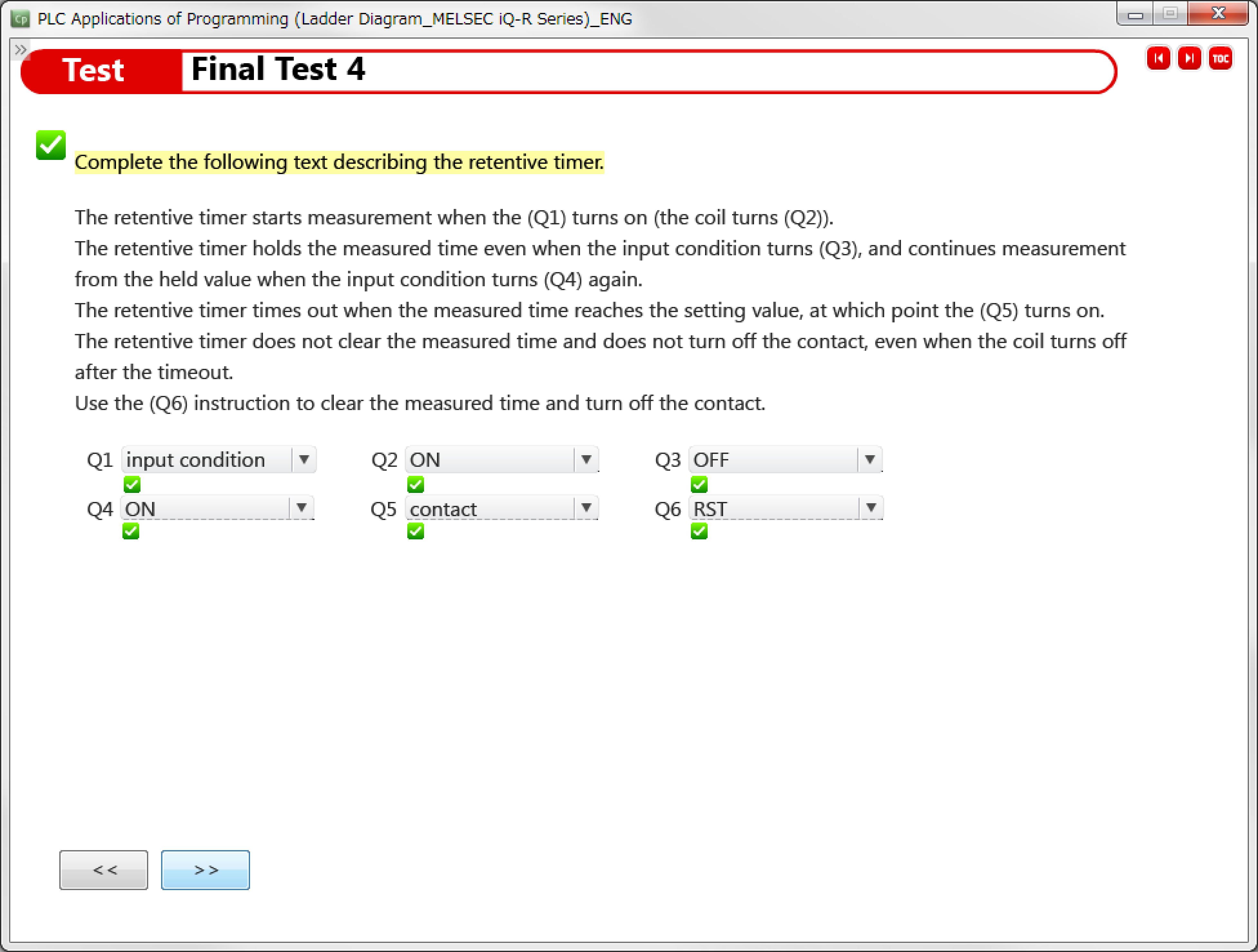


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC
<<
>>

Test Final Test 3

☑ Which of the following sentences is true about labels? (Multiple answers)

- ☑ ☑ Using labels helps identify the processing target and makes programming easier
- ☑ ☑ Labels that represent module signals and setting values are provided
- ☐ Comments can be added to elements to improve program readability
- ☑ ☑ Since constants can be assigned to labels, the values can be changed without modifying the program

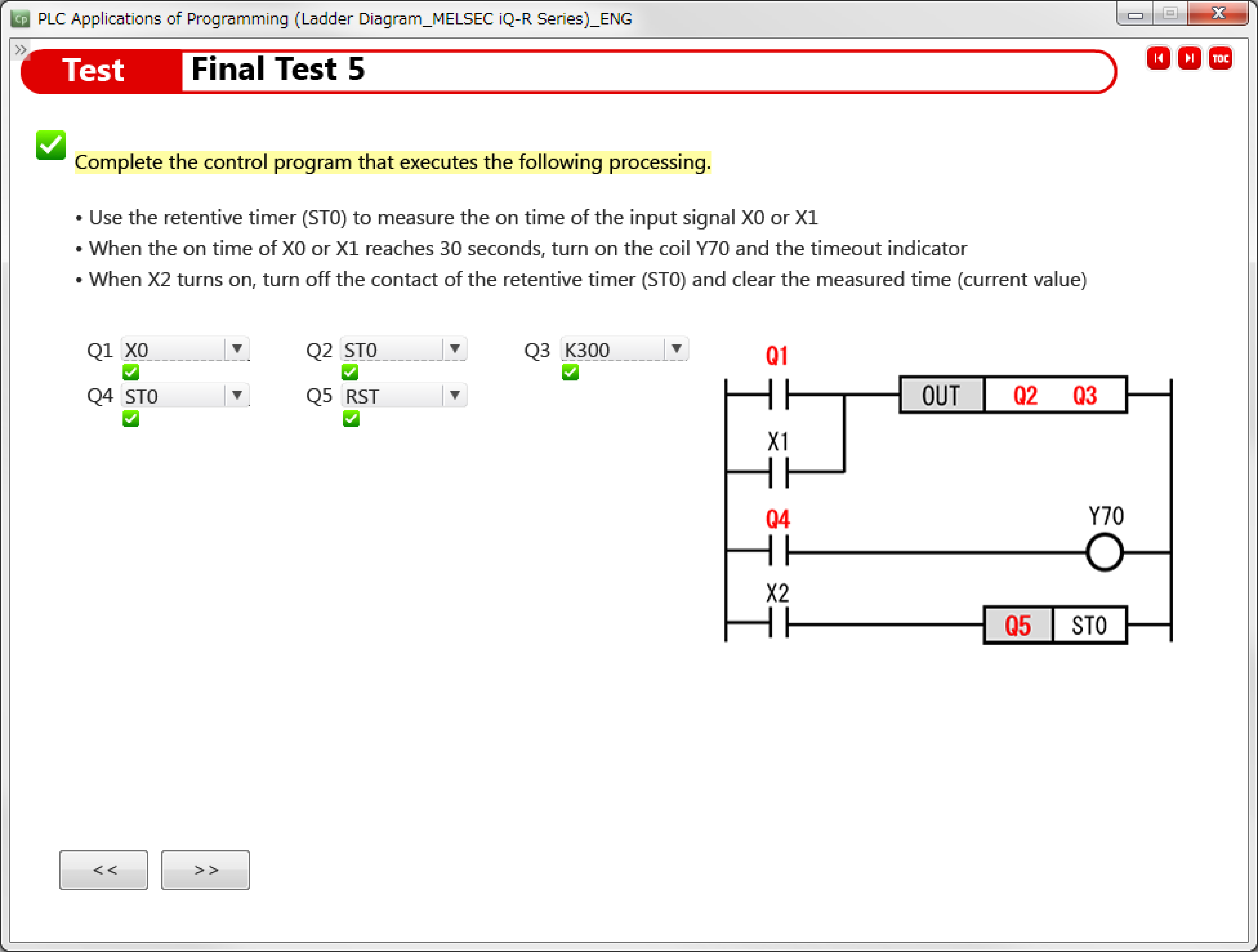

PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC
<<
>>

Test Final Test 4

✓ Complete the following text describing the retentive timer.

The retentive timer starts measurement when the (Q1) turns on (the coil turns (Q2)).
The retentive timer holds the measured time even when the input condition turns (Q3), and continues measurement from the held value when the input condition turns (Q4) again.
The retentive timer times out when the measured time reaches the setting value, at which point the (Q5) turns on.
The retentive timer does not clear the measured time and does not turn off the contact, even when the coil turns off after the timeout.
Use the (Q6) instruction to clear the measured time and turn off the contact.

Q1 input condition ✓
Q2 ON ✓
Q3 OFF ✓
Q4 ON ✓
Q5 contact ✓
Q6 RST ✓



PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG

Test Final Test 5

✓ Complete the control program that executes the following processing.

- Use the retentive timer (ST0) to measure the on time of the input signal X0 or X1
- When the on time of X0 or X1 reaches 30 seconds, turn on the coil Y70 and the timeout indicator
- When X2 turns on, turn off the contact of the retentive timer (ST0) and clear the measured time (current value)

Q1 X0 ✓

Q2 ST0 ✓

Q3 K300 ✓

Q4 ST0 ✓

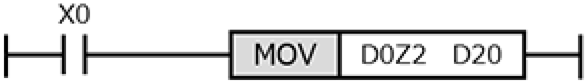
Q5 RST ✓

PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG

Test Final Test 6

✓ Select the value stored in the data register D20 when X0 turns on under each of the following conditions in the control program below.

Q1) When the value stored in Z2 is "0"
Q2) When the value stored in Z2 is "1"
Q3) When the value stored in Z2 is "2"
Q4) When the value stored in Z2 is "3"



Values stored in the data register

D0	100
D1	200
D2	400
D3	500

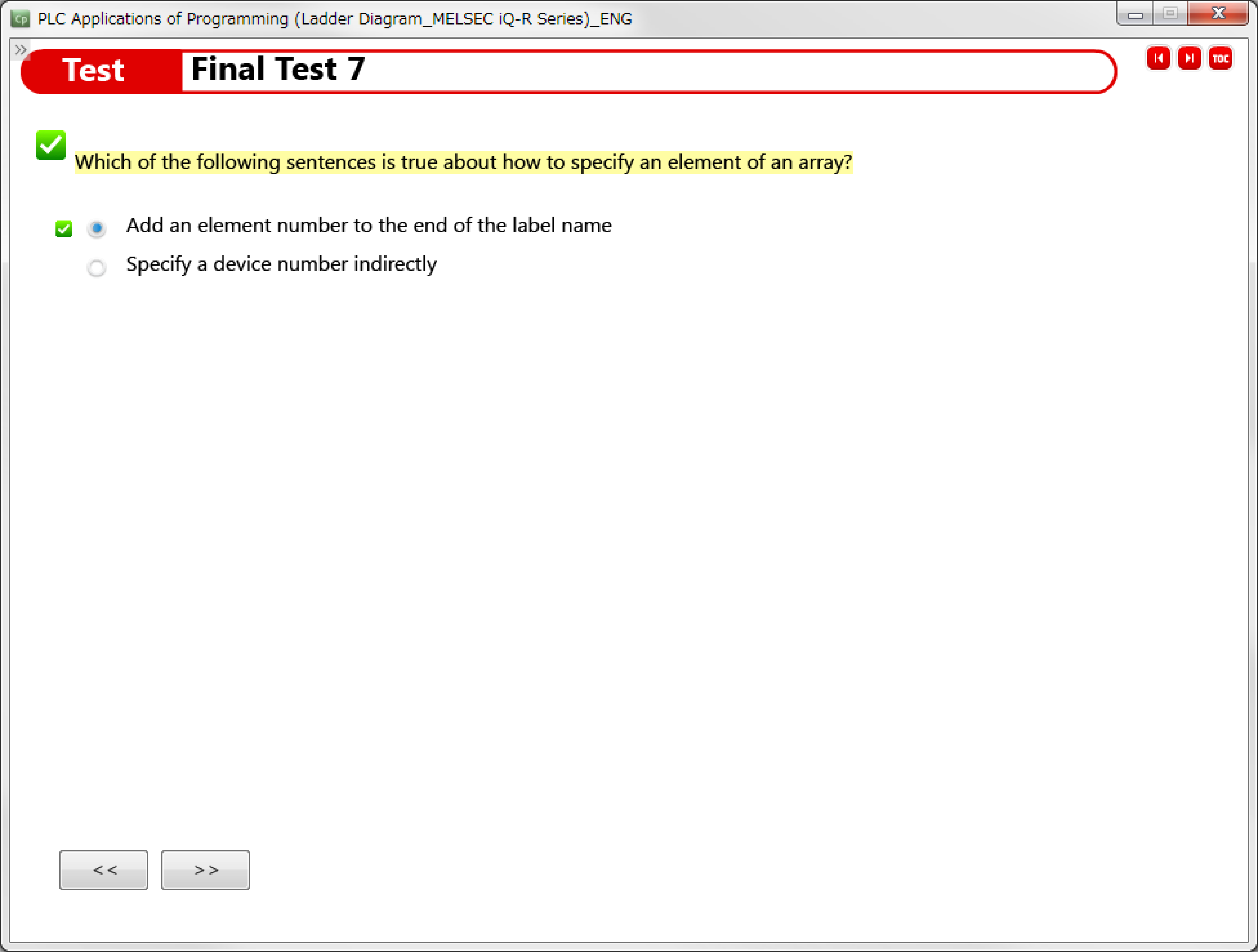
Q1 100 ▼ ✓

Q2 200 ▼ ✓

Q3 400 ▼ ✓

Q4 500 ▼ ✓

<< >>


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC
<<
>>

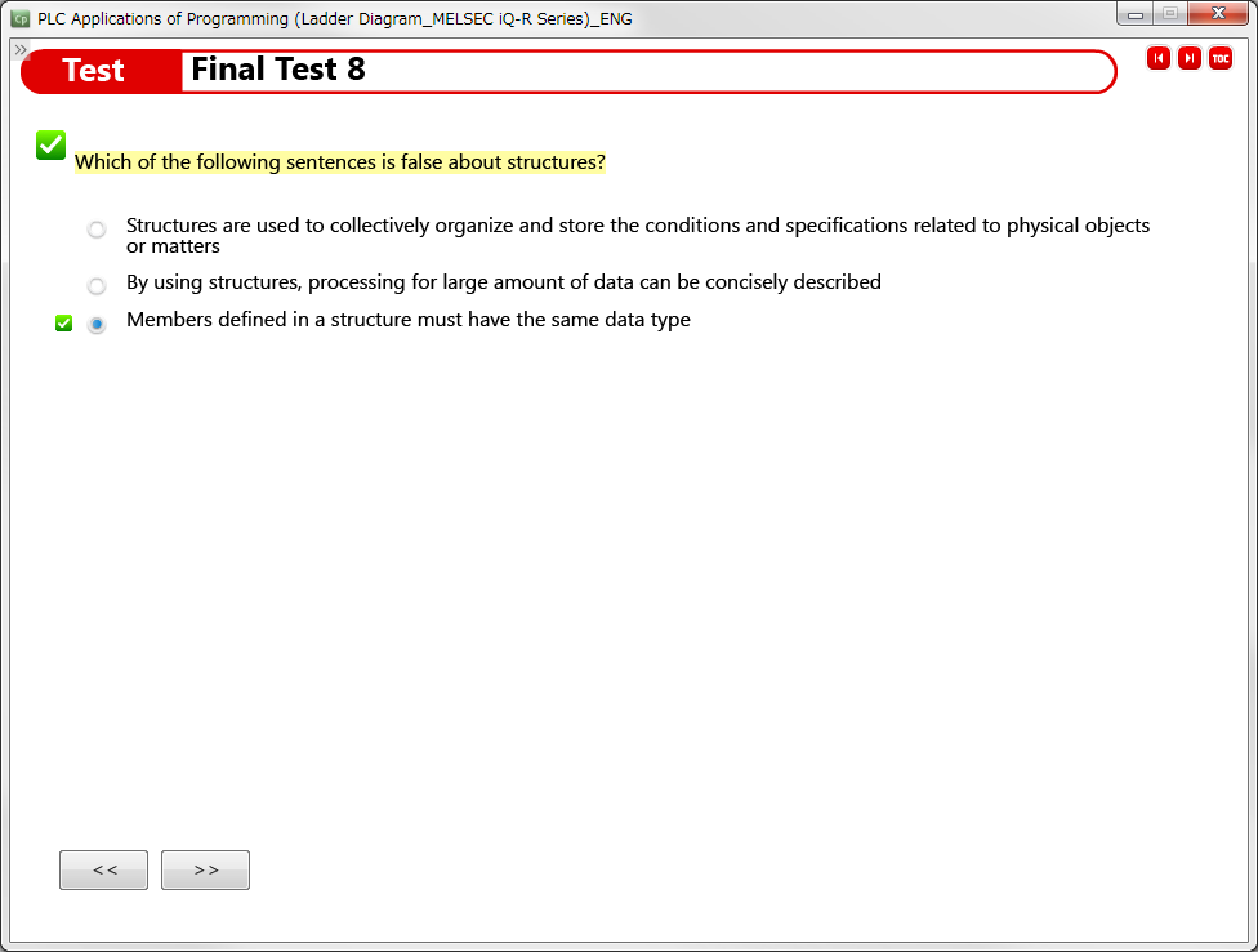
Test

Final Test 7

✓ Which of the following sentences is true about how to specify an element of an array?

✓ ● Add an element number to the end of the label name

○ Specify a device number indirectly


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC
<<
>>

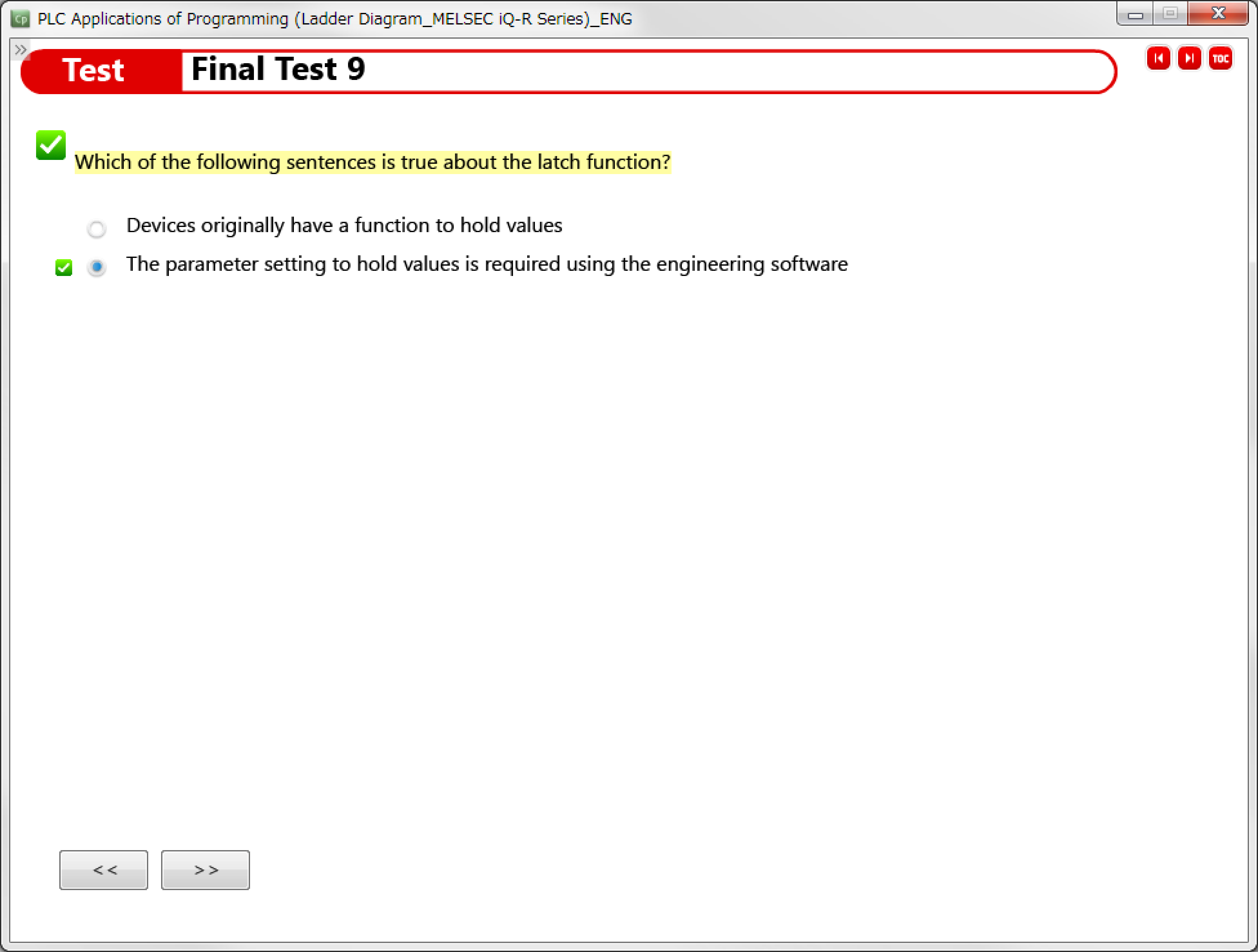
Test Final Test 8

✓ Which of the following sentences is false about structures?

○ Structures are used to collectively organize and store the conditions and specifications related to physical objects or matters

○ By using structures, processing for large amount of data can be concisely described

✓ ● Members defined in a structure must have the same data type



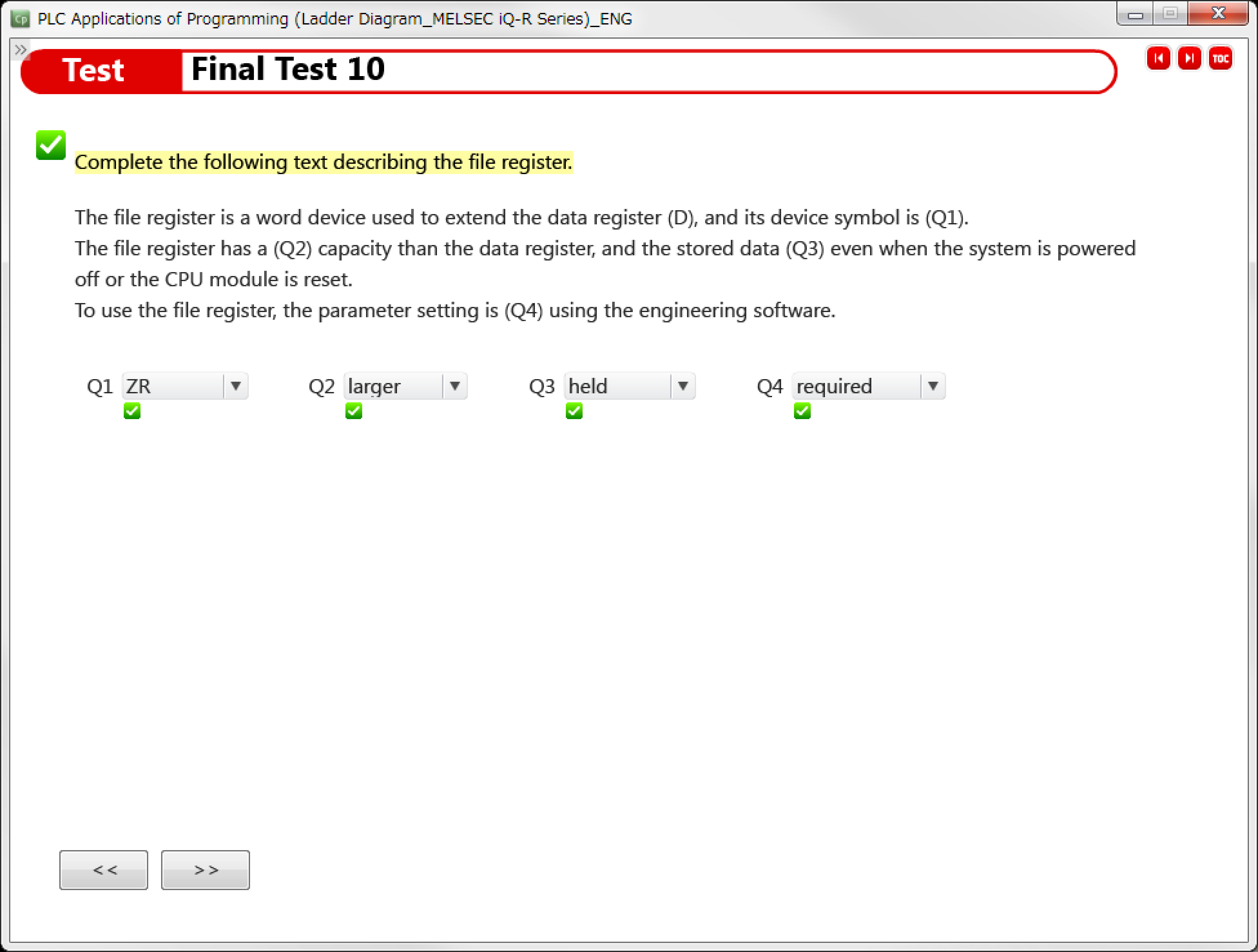
PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG

Test Final Test 9

✓ Which of the following sentences is true about the latch function?

○ Devices originally have a function to hold values

✓ ● The parameter setting to hold values is required using the engineering software



PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG

Test Final Test 10

✓ Complete the following text describing the file register.

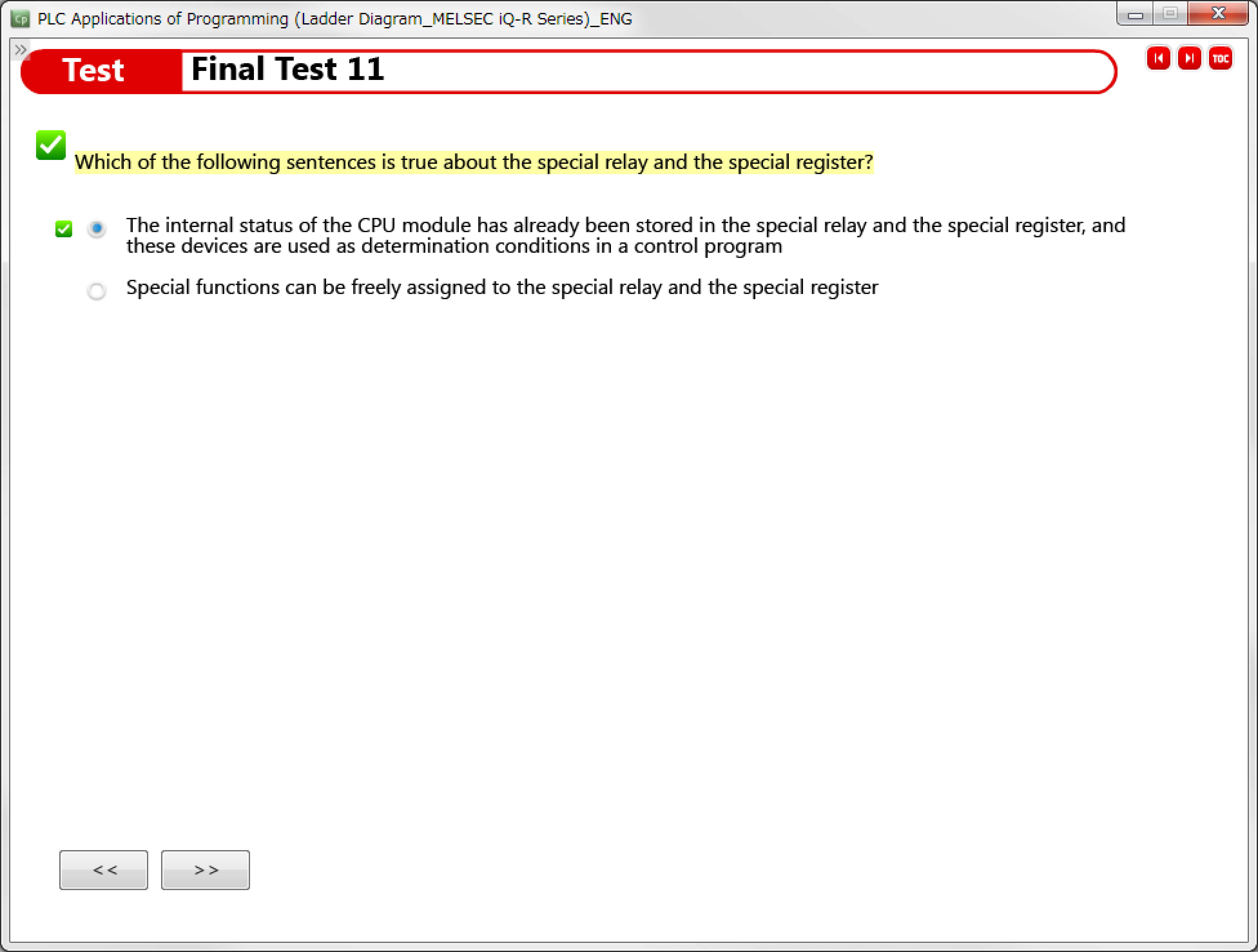
The file register is a word device used to extend the data register (D), and its device symbol is (Q1).
The file register has a (Q2) capacity than the data register, and the stored data (Q3) even when the system is powered off or the CPU module is reset.
To use the file register, the parameter setting is (Q4) using the engineering software.

Q1 ZR ▼ ✓

Q2 larger ▼ ✓

Q3 held ▼ ✓

Q4 required ▼ ✓



PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG

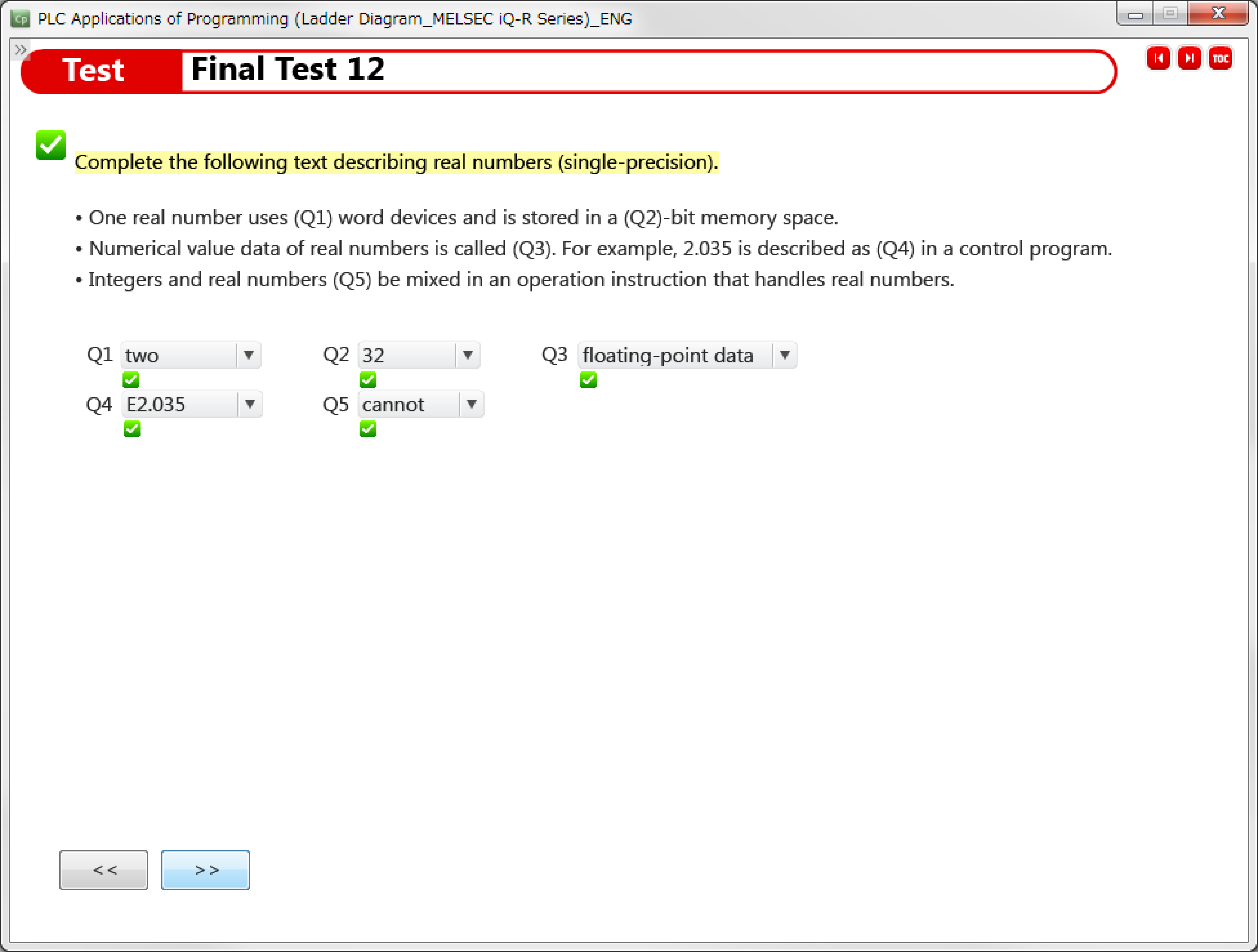
Test

Final Test 11

✓ Which of the following sentences is true about the special relay and the special register?

✓ ● The internal status of the CPU module has already been stored in the special relay and the special register, and these devices are used as determination conditions in a control program

○ Special functions can be freely assigned to the special relay and the special register

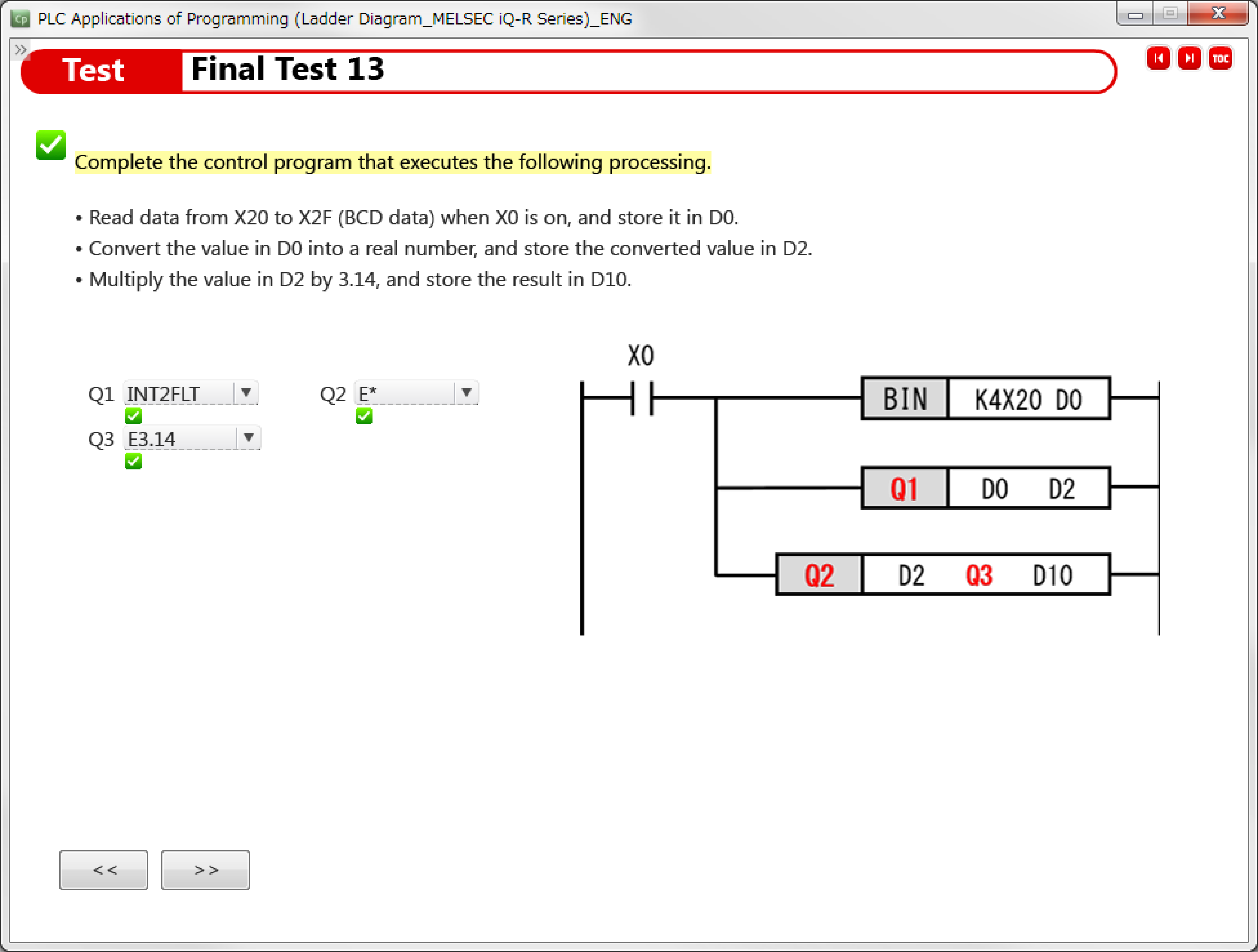

PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC
< <
> >

Test Final Test 12

✓ Complete the following text describing real numbers (single-precision).

- One real number uses (Q1) word devices and is stored in a (Q2)-bit memory space.
- Numerical value data of real numbers is called (Q3). For example, 2.035 is described as (Q4) in a control program.
- Integers and real numbers (Q5) be mixed in an operation instruction that handles real numbers.

Q1 two ▼ ✓
Q2 32 ▼ ✓
Q3 floating-point data ▼ ✓
Q4 E2.035 ▼ ✓
Q5 cannot ▼ ✓



PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG

Test Final Test 13

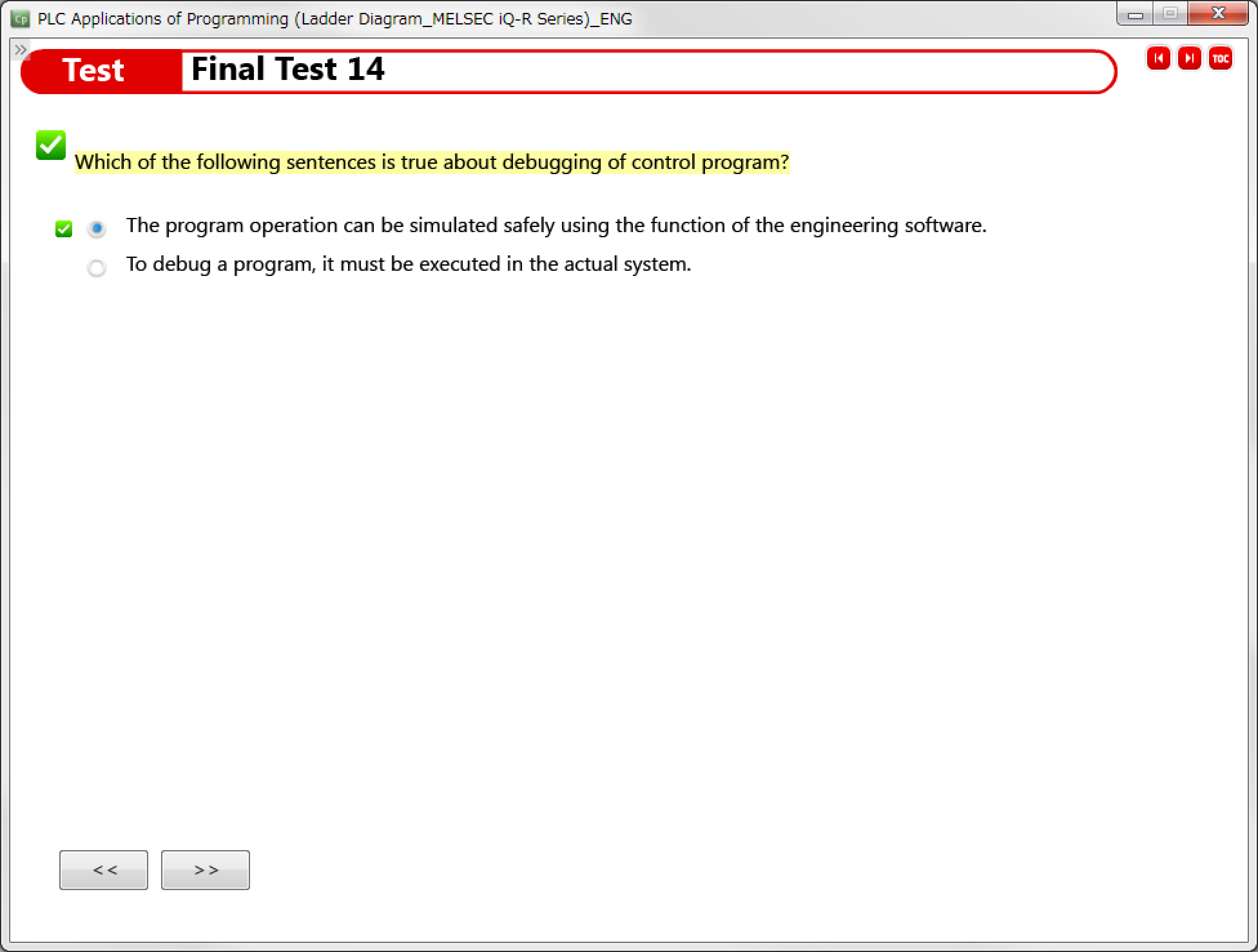
✓ Complete the control program that executes the following processing.

- Read data from X20 to X2F (BCD data) when X0 is on, and store it in D0.
- Convert the value in D0 into a real number, and store the converted value in D2.
- Multiply the value in D2 by 3.14, and store the result in D10.

Q1 INT2FLT ▼ ✓

Q2 E* ▼ ✓

Q3 E3.14 ▼ ✓

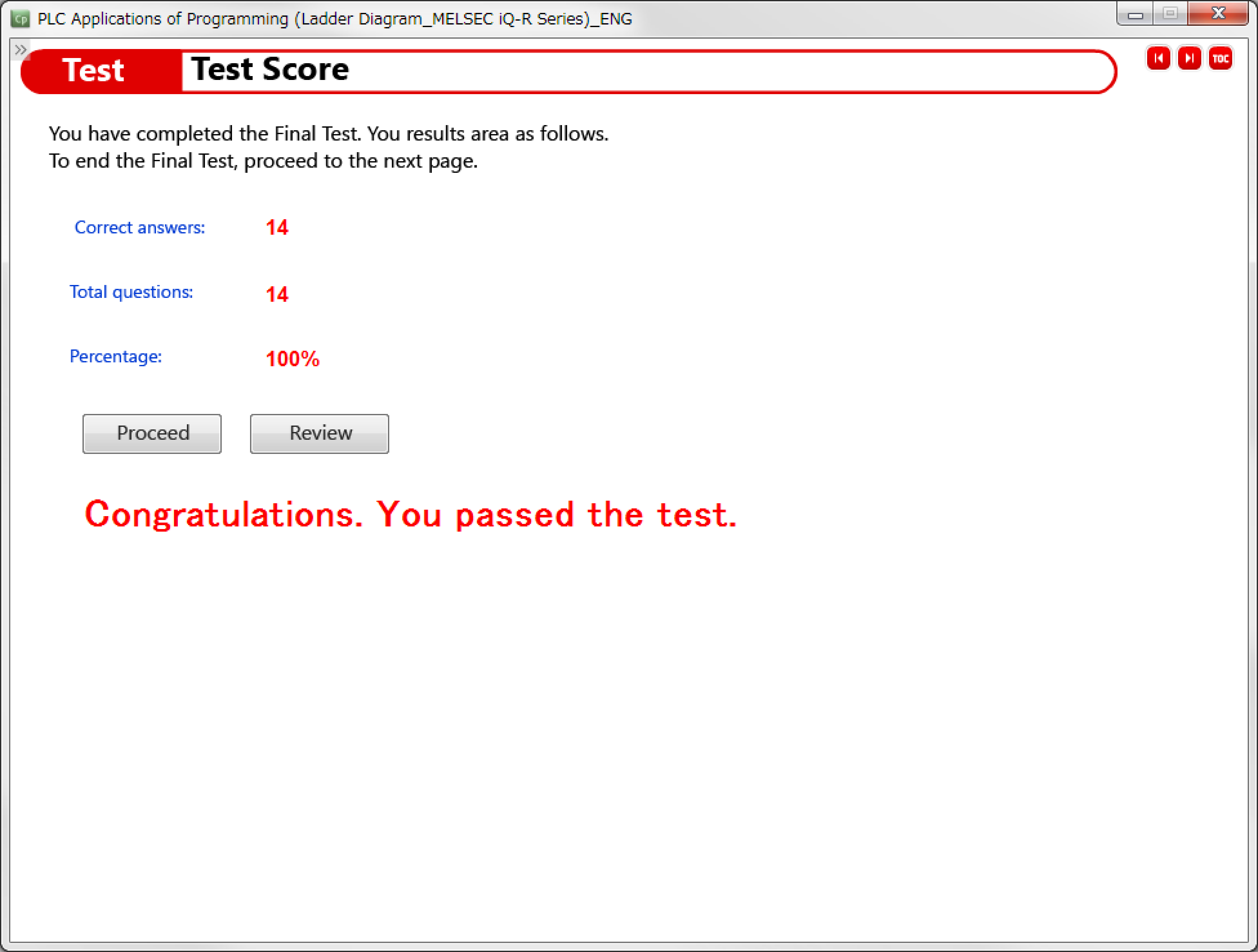

PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC
<<
>>

Test Final Test 14

✓ Which of the following sentences is true about debugging of control program?

✓ ● The program operation can be simulated safely using the function of the engineering software.

○ To debug a program, it must be executed in the actual system.


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC
Proceed
Review

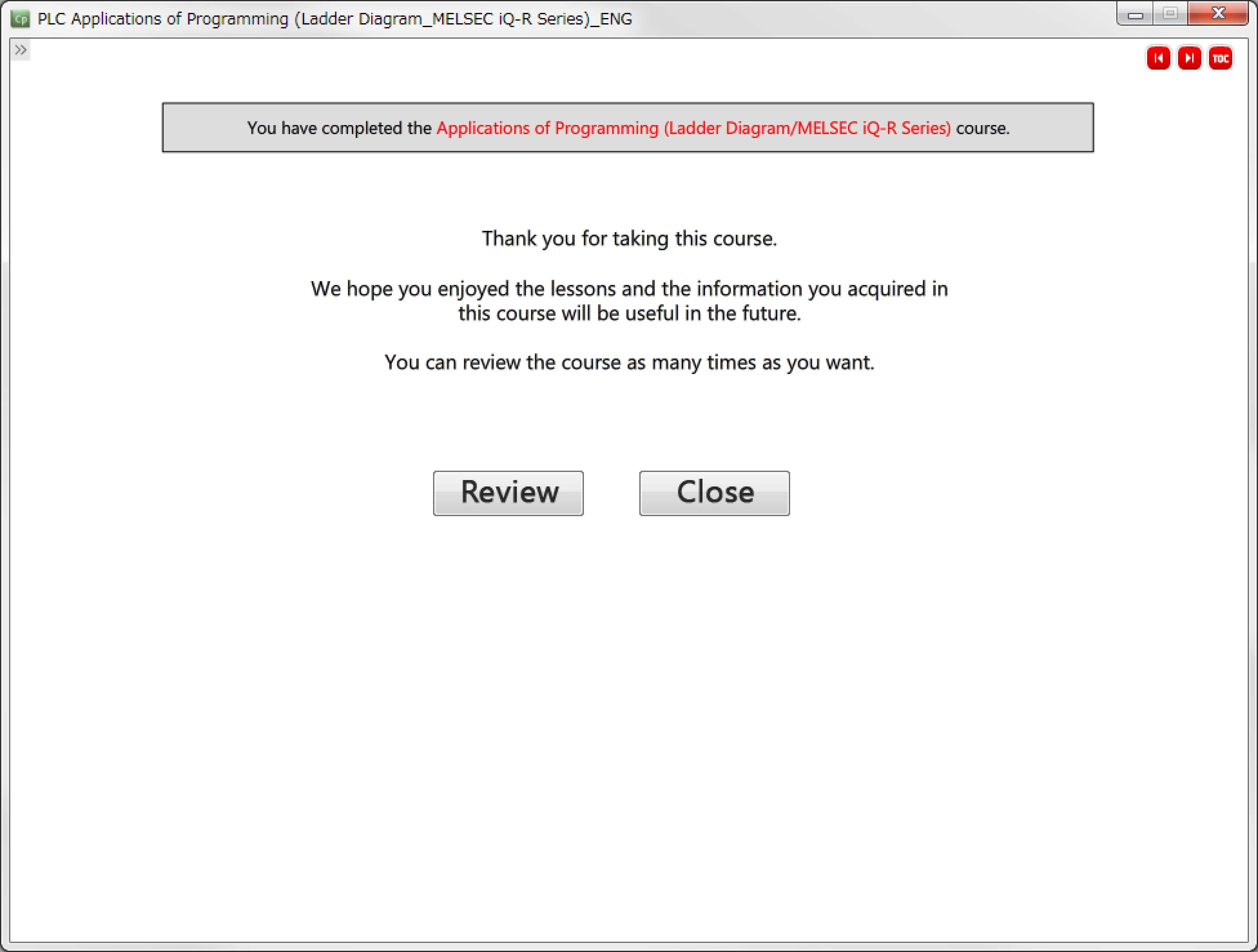
Test

Test Score

You have completed the Final Test. You results area as follows.
To end the Final Test, proceed to the next page.

Correct answers:	14
Total questions:	14
Percentage:	100%

Congratulations. You passed the test.


PLC Applications of Programming (Ladder Diagram_MELSEC iQ-R Series)_ENG
TOC
Review
Close

You have completed the Applications of Programming (Ladder Diagram/MELSEC iQ-R Series) course.

Thank you for taking this course.

We hope you enjoyed the lessons and the information you acquired in this course will be useful in the future.

You can review the course as many times as you want.